POLYPUBLIE

Algorithme de Branch and Price pour la confection d'horaires d'infirmières

Jean-Philippe Doyon

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

Date	2004
Type	Mémoire ou thèse / Dissertation or Thesis Mémoire de maîtrise/Master's thesis
URL de PolyPublie PolyPublie URL	https://publications.polymtl.ca/7311/
Conditions d'utilisation Terms of use	© Tous droits réservés / All rights reserved
Référence Citation	Doyon, J.-P. (2004). Algorithme de Branch and Price pour la confection d'horaires d'infirmières [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7311/

Détails de la thèse
Thesis details

Directeurs de recherche Advisors	Brigitte Jaumard
Programme Program	Non spécifié

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

UNIVERSITÉ DE MONTRÉAL

ALGORITHME DE BRANCH AND PRICE POUR LA CONFECTION D'HORAIRES D'INFIRMIÈRES

JEAN-PHILIPPE DOYON
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)
AVRIL 2004

© Jean-Philippe Doyon, 2004.

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisisitons et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-90868-2
Our file Notre référence
ISBN: 0-612-90868-2

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ALGORITHME DE BRANCH AND PRICE POUR LA CONFECTION D'HORAIRES D'INFIRMIÈRES

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

M. SAVARD Gilles, Ph.D., président
Mme. JAUMARD Brigitte, T. Doct., T. Hab., membre et directrice de recherche
M. DALLE Daniel, Ph.D., membre

RÉSUMÉ

Générer des horaires d'infirmières à l'aide de ressources informatiques est très intéressant pour les gestionnaires d'hôpitaux. En effet, l'utilisation d'un logiciel pour accomplir cette tâche permet non seulement d'économiser du temps et de l'argent, mais aussi de respecter les règles de la convention collective des infirmières, les contraintes de quotas et de satisfaire plus équitablement les préférences des infirmières.

Ce mémoire de maîtrise propose une méthode exacte pour résoudre le problème de confection d'horaires d'infirmières. La méthode utilisée est celle de la génération de colonnes couplée avec un algorithme de branch-and-bound afin de trouver une solution optimale à notre problème. Deux travaux antérieurs au présent mémoire ont déjà étudié la modélisation mathématique du problème et la composante nommée "problème auxiliaire" de notre méthode. Dans ce travail, nous étudions la composante nommée "problème maître" en plus d'approfondir l'interaction entre ces deux composantes de notre algorithme.

Mots clefs : horaire d'infirmière, génération de colonnes, valeur optimale, coût réduit, Simplexe, relaxation continue, algorithme de plus courts chemins.

ABSTRACT

Nurse scheduling generation using computer technology is very interesting for hospital managers. Using a program to perform this task enables to spend less time and less money, as well as to respect all the convention rules and quota constraints, and to suit in an equivalent way the nurses' preferences.

This master's thesis presents an exact method to solve the nurse scheduling problem. Our technique is based on the column generation method paired with a branch and bound algorithm with the final goal of finding an optimal solution. Two previous works to this one have already studied the mathematical model and the component named "auxiliary problem" of our method. In this work, we present the component named "master problem" as well as study the interaction between these two components of our algorithm.

Key Words: nurse schedule, column generation, optimal value, reduced cost, linear relaxation, shortest path algorithm.

TABLE DES MATIÈRES

RÉSUMÉ iv

ABSTRACT v

TABLE DES MATIÈRES vi

LISTE DES TABLEAUX xii

LISTE DES FIGURES xiv

LISTE DES ALGORITHMES xvi

LISTE DES SIGLES ET ABRÉVIATIONS xvii

LISTE DES ANNEXES xxiii

INTRODUCTION 1

CHAPITRE 1 : PROBLÉMATIQUE ET MÉTHODE UTILISÉE . 5

1.1 : Présentation du problème de confection d'horaires d'infirmières . . . 6

1.1.1 : L'objectif 7

1.1.2 : Les contraintes 7

1.2 : Revue des méthodes existantes 10

1.2.1 : Travaux précurseurs de notre méthode exacte 11

1.2.2 : Méthodes heuristiques 12

1.3 : Description et modélisation du problème NSP1 13

1.3.1 : Description du problème 13

1.3.2 : Présentation d'un exemple 16

1.3.3 : Modélisation mathématique 21

1.4 : Méthode retenue 26

1.4.1 : Trois groupes d'algorithmes 26

1.4.2 : Le branch and price 28

1.5 : Description et modélisation du problème NSP2 29

1.5.1 : Description du problème 29

1.5.2 : Présentation d'un exemple 31

1.5.3 : Modélisation mathématique 35

CHAPITRE 2 : L'ALGORITHME DE BRANCH AND PRICE . . 38

2.1 : Introduction au branch and price 38

2.2 : Présentation du problème (R) 41

2.3 : Méthode de génération de colonnes en 2 phases 43

2.3.1 : L'algorithme GC 44

2.3.2 : L'algorithme Pricing 49

2.3.3 : Description de la première phase GC_{ϕ_1} 53

2.3.4 : Condition de réalisabilité du problème (R^u) 56

2.3.5 : Description de la seconde phase GC_{ϕ_2} 57

2.3.6 : Organigramme et algorithme de la méthode de génération de colonnes en 2 phases 58

2.4 : Méthodes de séparation 59

2.4.1 : Séparation disjointe et complète 61

2.4.2 : Règles de branchement Ryan-Foster et binaire 63

2.4.3 : Deux classes de séparation possibles 67

2.4.4 : Technique pour obtenir une séparation efficace 69

2.5 : Exploration de l'arbre de recherche 70

2.5.1 : Introduction aux parcours DFS et BSFS 70

2.5.2 : Les algorithmes DFS et BSFS 72

2.6 : Exhaustivité du branch and price 77

2.6.1 : Arbre de recherche exhaustif 77

2.6.2 : Parcours implicitement exhaustif 79

CHAPITRE 3 : LES RÈGLES DE BRANCHEMENT 81

3.1 : Règle de branchement Ryan-Foster 81

3.1.1 : Condition nécessaire et définition 82

3.1.2 : Conditions pour obtenir un branchement Ryan-Foster efficace 84

3.2 : Règle de branchement binaire 87

3.2.1 : Condition nécessaire et définition 87

3.2.2 : Observations importantes 88

3.2.3 : Nouvelles contraintes obtenues par déductions logiques . . . 90

3.3 : Description du concept RFRC 96

3.3.1 : Présentation d'exemples 97

3.3.2 : Représentation binaire d'une combinaison 99

3.3.3 : Description de l'algorithme 100

CHAPITRE 4 : LE PROBLÈME AUXILIAIRE 104

4.1 : Description du problème auxiliaire 104

4.1.1 : Le graphe G_k 105

4.1.2 : Description de PA_{ϕ_1} 108

4.1.3 : Description de PA_{ϕ_2} 115

4.2 : Méthode pour le calcul de $|S_k^{cu}|$ 123

4.2.1 : Description de PA_{ϕ_3} 123

4.2.2 : Utilisation de PA_{ϕ_3} 126

4.3 : Modification du graphe G_k pour le problème NSP2 128

4.4 : Alternative au concept RFRC 130

CHAPITRE 5 : RÉSULTATS DE CALCUL 131

5.1 : Ressources informatiques utilisées 131

5.2 : Présentation des jeux de données 132

5.2.1 : Données du centre des naissances 133

5.2.2 : Données de l'urgence 133

5.2.3 : Données de l'urgence de Santa Gabrini (médecins) 135

5.3 : Résultats et analyse 137

5.3.1 : Données du centre des naissances 139

5.3.2 : Données de l'urgence 141

5.3.3 : Données pour le problème des médecins 144

CONCLUSION 147

BIBLIOGRAPHIE 150

ANNEXES 153

LISTE DES TABLEAUX

Tableau 1.1 : Les contraintes de quotas définies sur l'horizon. 18

Tableau 1.2 : Les horaires possibles pour les infirmières k_1, k_2 et k_3. . . . 20

Tableau 1.3 : Comptabilisation des quotas pour l'horaire ($k_1 : s_1$, $k_2 : s_1$, $k_3 : s_1$) de valeur 2. 20

Tableau 1.4 : Comptabilisation des quotas pour l'horaire ($k_1 : s_1$, $k_2 : s_1$, $k_3 : s_2$) de valeur 1. 21

Tableau 1.5 : Les contraintes de quotas définies sur l'horizon. 33

Tableau 1.6 : Les horaires possibles pour les infirmières k_1, k_2 et k_3. . . . 34

Tableau 1.7 : Comptabilisation des quotas pour l'horaire ($k_1 : s_1$, $k_2 : s_2$, $k_3 : s_2$) de valeur 1. 34

Tableau 1.8 : Comptabilisation des quotas pour l'horaire ($k_1 : s_2$, $k_2 : s_1$, $k_3 : s_1$) de valeur 3. 34

Tableau 1.9 : Comptabilisation des quotas pour l'horaire ($k_1 : s_2$, $k_2 : s_2$, $k_3 : s_2$) de valeur 3. 35

Tableau 3.1 : Définition des 8 combinaisons selon C_k^u. 97

Tableau 3.2 : Représentation binaire de la combinaison $c = 4$. 99

Tableau 5.1 : Données générales du centre des naissances. 133

Tableau 5.2 : Description des contraintes de quotas du centre des naissances. 134

Tableau 5.3 : Données générales de l'urgence. 134

Tableau 5.4 : Description des contraintes de quotas de l'urgence. 134

Tableau 5.5 : Données générales du problème de médecin. 136

Tableau 5.6 : Description des contraintes de quotas du problème de médecin. 136

Tableau 5.7 : Comparaison de la meilleure solution trouvée et du temps total de chacune des 4 versions de Iris. 139

Tableau 5.8 : Données des arbres obtenus par les 4 versions de Iris. . . . 140

Tableau 5.9 : La première et les 4 dernières solutions obtenues par la version BSFS-1 de Iris. 142

Tableau 5.10 :Données de l'arbre obtenu par la version BSFS-1 de Iris. . 143

Tableau 5.11 :Solutions obtenues par la version BSFS-1 de Iris. 144

Tableau 5.12 :Données de l'arbre obtenu par la version DSF-1 de Iris. . . 146

Tableau 5.13 :Solutions obtenues par la version DFS-1 de Iris. 146

LISTE DES FIGURES

Figure 1.1 : Exemple de relation quart-période. 18

Figure 1.2 : Formulation du problème NSP1 en un PLNE noté (P). . . . 25

Figure 1.3 : Formulation du problème NSP2 en un PLNE. 37

Figure 2.1 : Exemple d'arbre de recherche. 42

Figure 2.2 : Relaxation continue (R) du problème (P). 43

Figure 2.3 : Relaxation continue (R^{ua}) du problème (P^u) au noeud u de l'arbre de recherche. 55

Figure 2.4 : Organigramme de la méthode de génération de colonnes en deux phases pour la résolution de (R^u). 60

Figure 2.5 : Exemple d'une paire de variables s_1 et $s_2 \in S_k^{un}$ respectant les conditions nécessaires à la définition du branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$. 64

Figure 2.6 : Exemple d'une paire de variables s_1 et $s_2 \in S_k^{un}$ respectant les conditions nécessaires à la définition du branchement binaire $(k, d_1, t_1) \in \mathcal{D}_{\mathcal{T}_k}$. 65

Figure 2.7 : Le PLNE (P^u) au noeud u de l'arbre de recherche. 66

Figure 2.8 : Exemple de parcours DFS. 73

Figure 2.9 : Exemple de parcours BSFS où $z_1^* > z_6^*$, $z_7^* > z_8^*$ et $z_9^* > z_{10}^*$. 74

Figure 3.1 : Représentation matricielle des horaires d'une infirmière $k \in \mathcal{K}$. 85

Figure 4.1 : Le graphe $G_k = (V_k, E_k)$ associé à l'infirmière $k \in \mathcal{K}$. 108

Figure 4.2 : Représentation de l'ensemble d'état D_j d'un noeud $v_j \in V_k$. . 113

Figure 4.3 : Représentation du graphe d'états $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$ de l'infirmière $k \in \mathcal{K}$. 115

Figure 4.4 : Le graphe $G_k = (V_k, E_k)$ pour le problème NSP2, où $q = |\mathcal{L}_{dp}|$ pour $(d, p) \in \mathcal{D}_{\mathcal{P}}$. 129

Figure A.1 : Relation entre les quarts et les périodes du centre des naissances. 153

Figure A.2 : Relation entre les quarts et les périodes de l'urgence. 153

Figure B.1 : Arbre obtenu par l'application de la version BSFS-1 de Iris aux données de l'urgence du CUSM. 154

Figure B.2 : Arbre obtenu par l'application de la version BSFS-1 de Iris aux données du centre des naissances du CUSM. 155

LISTE DES ALGORITHMES

Algorithme 2.1 : Méthode de génération de colonnes pour résoudre le problème linéaire $(\dot{R}^u)$, qui correspond à (R^u) ou à (R^{ua}) du problème (P^u) au noeud u. 48

Algorithme 2.2 : L'algorithme Pricing où i est l'itération courante de GC. 53

Algorithme 2.3 : Méthode de génération de colonnes en deux phases pour la résolution de (R^u) au noeud u (v est le noeud père). 59

Algorithme 2.4 : Algorithme récursif du parcours DFS pour la résolution du PLNE (P^u) au noeud u. 75

Algorithme 2.5 : Algorithme récursif du parcours BSFS pour la résolution du PLNE (P^u) au noeud u. 76

Algorithme 3.1 : Méthode *calculer* $\mathcal{I}_k^u$ pour le problème (P^u) et pour une infirmière $k \in \mathcal{K}$. 103

Algorithme 4.1 : Description formelle de PA_{ϕ_2} appliqué au graphe $\hat{G}_{ki}^{cu}$. . 124

Algorithme 4.2 : Description formelle de PA_{ϕ_3} appliqué au graphe $\hat{G}_{k}^{cu}$. . 127

LISTE DES SIGLES ET ABRÉVIATIONS

Notations relatives à la formalisation du problème (R^{ua}) :

GC	Technique de génération de colonne.
GC_{ϕ_1}	Première phase de la méthode de génération de colonnes en 2 phases.
(R^{ua})	Relaxation continue résultant de l'ajout de variables artificielles à (R^u).
a_k	Variable artificielle créée pour l'infirmière $k \in \mathcal{K}$.
a_L	Variable artificielle créée pour la contrainte de quotas (Q_L).
n	Nombre d'itérations lors de l'application de GC au problème (R^{ua}).
z^*_{ua}	Valeur optimale du problème (R^{ua}).
(R^{una})	Dernier problème réduit (de (R^{ua})) considéré par GC.
z^n_{ua}	Valeur optimale du problème réduit (R^{una}).
x^n_{ua}	Solution de valeur optimale z^n_{ua} du problème réduit (R^{una}).

Notations relatives aux données du problème :

$\mathcal{K}$	$= \{ k_1, k_2, ..., k_{\lvert\mathcal{K}\rvert} \}$, l'ensemble des infirmières.
$\mathcal{D}$	$= \{ d_1, d_2, ..., d_{\lvert\mathcal{D}\rvert} \}$, l'ensemble des jours de l'horizon courant, où d_1 et $d_{\lvert\mathcal{D}\rvert}$ sont respectivement le premier et le dernier jour de l'horizon.
$\mathcal{T}$	$= \{ t_1, t_2, ..., t_{\lvert\mathcal{T}\rvert} \}$, l'ensemble des quarts de travail.

$\mathcal{P}$ $= \{ p_1, p_2, ..., p_{|\mathcal{P}|} \}$, l'ensemble des périodes qui forment la partition d'une journée de l'horizon.

$\mathcal{D}_\mathcal{P}$ $= \{ (d,p) : d \in \mathcal{D}, p \in \mathcal{P} \}$, l'ensemble des paires jour - période.

Λ $= \{ \ell_1, \ell_2, \ldots, \ell_{|\Lambda|} \}$, l'ensemble des compétences des infirmières.

$\mathcal{T}_k$ $\subseteq \mathcal{T}$, l'ensemble des quarts de travail possibles pour $k \in \mathcal{K}$.

$\mathcal{D}_{\mathcal{T}_k}$ $= \{ (d,t) : d \in \mathcal{D}, t \in \mathcal{T}_k \}$, l'ensemble des affectations possibles pour $k \in \mathcal{K}$.

(Q_L) Contrainte de quotas de type A définie sur $(d,p) \in \mathcal{D}_\mathcal{P}$ et $L \in \mathcal{L}_{dp}$ et associée au triplet (q_L, q_L^-, q_L^+) (respectivement quota cible, déficit et surplus maximaux acceptés).

$\mathcal{L}_{dp}$ $= \{L_1, L_2, \ldots, L_{|\mathcal{L}_{dp}|}\}$, chaque $L \in \mathcal{L}_{dp}$ est tel que $L \subseteq \Lambda$ et est associé à une contrainte (Q_L) définie sur $(d,p) \in \mathcal{D}_\mathcal{P}$.

Λ_k $\subseteq \Lambda$, l'ensemble des compétences de l'infirmière $k \in \mathcal{K}$.

$\mathcal{O}_k$ et $\mathcal{F}_k$ $\subseteq \mathcal{D}_{\mathcal{T}_k}$, respectivement l'ensemble des affectations obligatoires et interdites à l'infirmière $k \in \mathcal{K}$.

Notations relatives à la formalisation du PLNE (P) :

(P) Formulation du problème NSP1 sous forme d'un Programme Linéaire en Nombres Entiers (PLNE).

S L'ensemble des solutions (entières) du PLNE (P).

S_k L'ensemble des horaires valides de l'infirmière $k \in \mathcal{K}$ pour le PLNE (P).

z^* Meilleure valeur connue du PLNE (P). Initialement, $z^* = +\infty$.

p_{ks} Coefficient indiquant le poids de l'horaire $s \in S_k$ de l'infirmière $k \in \mathcal{K}$.

p^- (p^+) Pénalité accordée par unité de déficit (surplus) avec le quota requis de chaque contrainte de quotas.

α_{tp} Constante égale à 1 si le quart $t \in \mathcal{T}$ couvre la période $p \in \mathcal{P}$ et à 0 sinon.

b_{kL} Constante égale à 1 si l'infirmière $k \in \mathcal{K}$ peut apporter sa contribution à une contrainte de quotas de type A définie sur $L \subseteq \Lambda$ et à 0 sinon.

x_{ks} Variable de décision égale à 1 si l'horaire $s \in S_k$ est affecté à l'infirmière $k \in \mathcal{K}$ dans l'horizon courant et à 0 sinon.

a_{ksdt} Variable égale à 1 indiquant si l'affectation $(d,t) \in \mathcal{D}_{T_k}$ a été affectée à l'infirmière $k \in \mathcal{K}$ dans l'horaire $s \in S_k$ et à 0 sinon.

s_L^- (s_L^+) Variables d'écart de déficit (surplus) associée à la contrainte de quotas (Q_L), où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_{\mathcal{P}}$.

λ_L Variable duale associée à la contrainte de quotas (Q_L), où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_{\mathcal{P}}$.

μ_k Variable duale associée à la contrainte de partionnement de l'infirmière $k \in \mathcal{K}$.

c_{ks} Coût réduit de l'horaire $s \in S_k$ de l'infirmière $k \in \mathcal{K}$.

Notations relatives aux méthodes de branchement (Ryan-Foster et binaire) et à la modélisation du PLNE (P^u) :

u^0 Premier noeud de l'arbre de recherche.

(P^u) PLNE du noeud u de l'arbre de recherche.

S^u $\subseteq S$, l'ensemble des solutions (entières) du PLNE (P^u).

S_k^u $\subseteq S_k$, l'ensemble des horaires valides de l'infirmière $k \in \mathcal{K}$ pour le PLNE (P^u).

RFRC	Ryan-Foster Respect Combination.
$\mathcal{D}^2_{\mathcal{I}_k}$	$= \{ (d_i, t_i, d_j, t_j) : (d_i, t_i),\ (d_j, t_j) \in \mathcal{D}_{\mathcal{I}_k},\ (d_i, t_i) \neq (d_j, t_j) \}$.
$\mathcal{C}^u_{k(one)}$, $\mathcal{C}^u_{k(zero)}$	$\subseteq \mathcal{D}_{\mathcal{I}_k}$ composés des contraintes de branchement binaire définies pour $k \in \mathcal{K}$ pour le PLNE (P^u) : chaque $(d,t) \in \mathcal{C}^u_{k(one)}$ est obligatoire et chaque $(d,t) \in \mathcal{C}^u_{k(zero)}$ est interdite.
$\mathcal{C}^u_{k=}$, $\mathcal{C}^u_{k\neq}$	$\subseteq \mathcal{D}^2_{\mathcal{I}_k}$ composés des contraintes de branchement Ryan-Foster définies pour $k \in \mathcal{K}$ pour le PLNE (P^u) : $\forall (d_i, t_i, d_j, t_j) \in \mathcal{C}^u_{k=}$ $a_{ksd_it_i} = a_{ksd_jt_j}$ et $\forall (d_i, t_i, d_j, t_j) \in \mathcal{C}^u_{k\neq}$ $a_{ksd_it_i} \neq a_{ksd_jt_j}$.
$\mathcal{C}^u_k$	$= \mathcal{C}^u_{k=} \bigcup \mathcal{C}^u_{k\neq}$, l'ensemble de toutes les contraintes de branchement Ryan-Foster définies pour $k \in \mathcal{K}$ pour le PLNE (P^u).
$\mathcal{I}^u_k$	$\subseteq \{0,\ 1,\ \ldots,\ 2^{\lvert\mathcal{C}^u_k\rvert} - 1\}$, l'ensemble des RFRC possibles pour l'infirmière $k \in \mathcal{K}$ pour le PLNE (P^u).
$\mathcal{K}_{\mathcal{I}^u_k}$	$= \{(k,c) : k \in \mathcal{K},\ c \in \mathcal{I}^u_k\}$. Une RFRC $c \in \mathcal{I}^u_k$ d'une infirmière $k \in \mathcal{K}$ au PLNE (P^u) est notée par $(k,c) \in \mathcal{K}_{\mathcal{I}^u_k}$.
$\mathcal{K}_{\mathcal{D}_{\mathcal{I}_k}}$	$= \{ (k,d,t) : k \in \mathcal{K},\ (d,t) \in \mathcal{D}_{\mathcal{I}_k}\}$. Un branchement binaire est noté par $(k,d,t) \in \mathcal{K}_{\mathcal{D}_{\mathcal{I}_k}}$.
$\mathcal{K}_{\mathcal{D}^2_{\mathcal{I}_k}}$	$= \{ (k, d_i, t_i, d_j, t_j) : k \in \mathcal{K},\ (d_i, t_i, d_j, t_j) \in \mathcal{D}^2_{\mathcal{I}_k}\}$. Un branchement Ryan-Foster est noté par $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{I}_k}}$.
$\mathcal{O}^{cu}_k$ et $\mathcal{F}^{cu}_k$	Respectivement l'ensemble des affectations obligatoires et interdites définissant $(k,c) \in \mathcal{K}_{\mathcal{I}^u_k}$.
S^{cu}_k	$\subseteq S^u_k$, l'ensemble des horaires respectant la combinaison $(k,c) \in \mathcal{K}_{\mathcal{I}^u_k}$ du PLNE (P^u).

Notations relatives à la phase 2 :

GC_{ϕ_2}	Deuxième phase de la méthode de génération de colonnes en 2 phases.
(R^u)	Relaxation continue (programme linéaire) du PLNE (P^u).
n	Nombre d'itérations lors de l'application de GC au problème (R^u).
z_u^*	Valeur optimale du problème (R^u). Initialement, $z_u^* = +\infty$.
(R^{ui})	Le problème réduit de (R^u) à l'itération i de GC.
S_k^{ui}	$\subseteq S_k^u$, l'ensemble des horaires de $k \in \mathcal{K}$ inclus dans (R^{ui}).
z_u^i	Valeur optimale du problème (R^{ui}).
x_u^i	Solution primale-duale réalisable (optimale) de valeur z_u^i du problème (R^{ui}).
(R^{un})	Dernier problème réduit considéré par GC au problème (R^u).
z_u^n	Valeur optimale du problème (R^{un}). Utilisé par le test $z_u^n \stackrel{?}{=} z_u^*$.
x_u^n	Solution optimale de valeur z_u^n du problème (R^{un}).

Notations relatives au problème auxiliaire et à une infirmière $k \in \mathcal{K}$:

G_k	$= (V_k, E_k)$, le graphe de l'infirmière k, où $V_k = \{v_0, v_1, \ldots, v_n\}$ et $E_k \subseteq \{(v_i, v_j) : v_i, v_j \in V_k,\ v_i \neq v_j\}$.
v_0 et v_n	Respectivement le noeud source et le noeud puits du graphe G_k.
D_i	Ensemble des états réalisables au noeud $v_i \in V_k$.
$R(x_j)$	Ensemble des états qui ont donné naissance à l'état $x_j \in D_j$ au noeud $v_j \in V_k$.

PA_{ϕ_1}	La phase 1, du problème auxiliaire, appliquée au graphe G_k pour définir le graphe d'états $\hat{G}_k$.
$\hat{G}_k$	$= (\hat{V}_k, \hat{E}_k)$, le graphe d'états obtenu par $PA_{\phi_1}(G_k)$.
$\hat{V}_k$	$= \{v_i \cdot x_i : v_i \in V_k,\ x_i \in D_i\}$ est l'ensemble des noeuds d'état de $\hat{G}_k$.
$\hat{E}_k$	$= \{(v_i \cdot x_i,\ v_j \cdot x_j) : v_i,\ v_j \in V_k,\ x_i \in D_i,\ x_j \in D_j,\ x_i \in R(x_j)\}$ est l'ensemble des arcs d'états de $\hat{G}_k$.
c_{ij}	Coût associé à l'arc $(v_i, v_j) \in E_k$ et aux arcs d'état $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$.
$\hat{G}_{ki}^{cu}$	Graphe d'états représentant la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ du PLNE (P^u) et les variables duales de l'itération courante i de GC.
PA_{ϕ_2}	La phase 2, du problème auxiliaire, appliquée au graphe $\hat{G}_{ki}^{cu}$ pour obtenir un ensemble, noté $PA_{\phi_2}(\hat{G}_{ki}^{cu})$, de variables $s \in S_k^{cu}$ hors base et de coût réduit négatif $(c_{ks} < 0)$.
$P_k^{cu}(v_j \cdot x_j)$	Pour $v_j \cdot x_j \in \hat{V}_k$, l'ensemble des sous-chemins $v_0 \cdot x_0 \rightsquigarrow v_j \cdot x_j$ qui sont réalisables pour la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ du PLNE (P^u).
P_k^{cu}	L'ensemble de tous les chemins $v_0 \cdot x_0 \rightsquigarrow v_n \cdot x_n$, où $v_n \cdot x_n \in \hat{V}_k$ est un puits quelconque, qui sont réalisables pour la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ du PLNE (P^u).

LISTE DES ANNEXES

ANNEXE A : RELATION ENTRE LES QUARTS ET LES PÉRIODES DES FICHIERS DU CUSM 153

ANNEXE B : ARBRES DE RECHERCHE OBTENUS PAR IRIS 154

INTRODUCTION

La confection automatisée d'horaires de personnel est un sujet très actuel et de nombreux chercheurs (euses) étudient ce problème. Que le personnel considéré soit du domaine hospitalier, ferroviaire ou aérien, etc., les secteurs d'application sont nombreux. Dans le présent mémoire, nous considérons le problème de confection d'horaires d'infirmières, abrégé par NSP (Nurse Scheduling Problem). D'ailleurs, il est important de préciser que l'utilisation d'un logiciel pour la résolution du NSP permet d'économiser du temps et de l'argent à l'hôpital. En effet, cette tâche est habituellement accomplie par l'infirmière-chef de l'unité considérée et nécessite environ l'équivalent d'une semaine de travail pour chaque horizon (4 à 6 semaines suivant les hôpitaux).

La méthode proposée dans ce mémoire occupe une place importante dans le domaine de la recherche opérationnelle et plus précisément, dans celui de la programmation mathématique. En effet, la technique que nous avons choisie est celle de la méthode de génération de colonnes couplée avec un algorithme de branch-and-bound afin d'obtenir une méthode exacte, soit l'algorithme de séparation et d'évaluation progressive (branch and price). L'algorithme est décomposé en deux niveaux : le problème maître et le problème auxiliaire. Le premier problème utilise la méthode de génération de colonnes pour résoudre la relaxation continue d'un programme linéaire en nombres entiers afin d'obtenir une borne inférieure sur la valeur optimale de ce dernier. Le deuxième problème est un algorithme de plus court chemin dans un graphe d'états et a la tâche de générer de nouvelles variables de coût réduit négatif lors de l'application de la méthode de génération de colonnes.

Le projet a débuté en 1997 avec la thèse de doctorat de Tsevi Vovor [1] et a été

suivi, l'année suivante, par l'article de Jaumard, Semet et Vovor [2]. Leurs travaux portaient essentiellement sur le développement de l'algorithme de plus court chemin avec contraintes de ressources, sur la formulation du NSP en un programme mathématique en nombres entiers et sur la modélisation des ressources pour les horaires de personnel soignant. Enfin, en 2000, le mémoire de maîtrise de Pascal Labit [3] a approfondi la modélisation des ressources pour répondre à un niveau de raffinement élevé de la description des contraintes propres aux infirmières.

Toutefois, aucun des travaux mentionnés ci-haut n'ont étudié l'adaptation du branch and price pour la résolution du NSP en plus des différentes subtilités qui existent entre cet algorithme et le problème auxiliaire. C'est pourquoi nous allons approfondir ces deux points du projet à l'intérieur du présent mémoire.

Structure du mémoire

Le premier chapitre a pour tâche d'introduire le lecteur au NSP et à la méthode que nous avons choisie pour résoudre ce problème. Nous présentons aussi une revue de la littérature sur ce problème en nous concentrant sur les méthodes exactes et les méthodes heuristiques. Pour les deux versions du NSP, dont l'une n'a jamais été présentée auparavant, nous exposons leur description et leur modélisation. Aussi, nous présentons brièvement les trois différents groupes d'algorithmes qui utilisent la méthode de génération de colonnes et nous introduisons le lecteur au branch and price.

Le chapitre deux décrit de façon détaillée la plupart des composantes du branch and price : méthode de génération de colonnes, l'algorithme "pricing", les deux méthodes de séparation utilisées (branchement binaire et branchement Ryan-Foster) et les deux algorithmes possibles pour l'exploration de l'espace de solutions (recherche

en profondeur d'abord et recherche avec le meilleur des deux fils d'abord). Nous terminons en discutant de l'exhaustivité du branch and price.

Le troisième chapitre est entièrement consacré aux deux méthodes de séparation. Pour le branchement Ryan-Foster et le branchement binaire, nous présentons les conditions nécessaires à leur application et nous donnons leur définition. Aussi, quelques observations importantes pour chacun de ces deux branchements sont présentées. En outre, nous montrons comment il est parfois possible d'obtenir de nouvelles contraintes lors de l'application d'un branchement binaire. Enfin, le nouveau concept de "combinaison du respect des contraintes Ryan-Foster" (RFRC) et l'algorithme s'y rattachant sont présentés.

Le chapitre quatre porte sur une description des algorithmes et des structures du problème auxiliaire. Nous décrivons le graphe des ressources, le graphe des états réalisables, la structure de données utilisée pour les ensembles d'états et les deux phases de l'algorithme de plus court chemin de Vovor [1]. Ensuite, nous développons un nouveau algorithme qui dénombre le nombre d'horaires réalisables pour une infirmière donnée et nous exposons quelques utilisations possibles de cette méthode dans le branch and price. Finalement, les modifications à apporter au problème auxiliaire pour son adaptation à la deuxième version du NSP sont présentées.

Enfin, le chapitre cinq présente 3 jeux de données et les résultats obtenus pour chacun d'eux. Ensuite, nous analysons ces résultats afin de comparer quelques stratégies du branch and price.

Apports du mémoire

Finalement, nous énumérons ci-dessous les principaux apports théoriques du présent mémoire sur l'avancement du projet.

1. Première étude sur la faisabilité d'adapter le projet pour la résolution de la seconde version du NSP.
2. Définition du concept RFRC et développement de l'algorithme s'y rattachant.
3. Application de déductions logiques afin d'obtenir (peut-être) plus d'une contrainte de branchement lors de l'usage d'un branchement binaire.
4. Développement d'une méthode dénombrant le nombre d'horaires réalisables pour une infirmière donnée.

CHAPITRE 1 : PROBLÉMATIQUE ET MÉTHODE UTILISÉE

L'objectif de ce chapitre est d'introduire le lecteur au problème de confection automatisée d'horaires d'infirmières. Tout d'abord, une brève description du problème est donnée. Deuxièmement, nous survolons les différentes méthodes déjà existantes pour résoudre le problème. Ensuite, une première version du problème [1] et sa modélisation mathématique en un programme en nombres entiers sont décrites. Quatrièmement, nous présentons la méthode retenue, qui est un algorithme de branch and price décomposant le problème en un problème maître et un problème auxiliaire. Finalement, nous décrivons une deuxième version du problème de confection d'horaires d'infirmières [2] ainsi que sa modélisation mathématique. Cette seconde version est une généralisation de la première par rapport aux contraintes de quotas que nous allons définir un peu plus loin.

Les différences entre les problèmes NSP1 et NSP2 sont peu nombreuses. NSP1 permet une seule compétence par infirmière et chaque contrainte de quotas est définie sur une période de la journée. NSP2 permet plus d'une compétence par infirmière et chaque contrainte de quotas est définie sur un quart et peut être de type A ou B, contrairement au problème NSP1 où seules les contraintes de type A sont possibles.

Il est important de mentionner que plusieurs travaux ont déjà présenté et traité le problème NSP1. Premièrement, il y a eu la thèse de Vovor [1] et l'article de Jaumard,

[1] La première version du problème de génération automatisée d'horaires d'infirmières sera notée NSP1 (Nurse Scheduling Problem).

[2] La deuxième version sera notée NSP2.

Semet et Vovor [2] qui décrivent la modélisation mathématique et le problème auxiliaire. Ensuite, le mémoire de maîtrise de Labit [3] décrit très bien la problématique et l'adaptation apportée au problème auxiliaire pour répondre à un niveau élevé de la modélisation du problème.

Concernant le présent mémoire, ses apports couvrent principalement l'algorithme de branch and price, un algorithme de plus court chemin pour résoudre le problème auxiliaire et les méthodes de branchement. Tout ceci n'étant appliqué qu'au problème NSP1. En effet, concernant le problème NSP2, nous ne ferons que décrire la problématique, la modélisation mathématique et les modifications à apporter au problème auxiliaire.

Finalement, pour avoir plus de détails sur les contraintes intrinsèques à une infirmière et sur leurs modélisations dans le problème auxiliaire, nous référons le lecteur à l'un ou l'autre des trois documents cités ci-haut.

1.1 Présentation du problème de confection d'horaires d'infirmières

Maintenant, décrivons les données reçues pour ensuite expliquer l'objectif à atteindre tout en respectant certaines contraintes. Premièrement, un horizon de planification est donné. Il est composé d'une date de début et d'une date de fin. Ensuite, une banque d'environ 10 à 80 infirmières est fournie. Pour chacune de ces infirmières, des contraintes intrinsèques sont définies. Troisièmement, un ensemble de contraintes de quotas est donné. Chacune de ces contraintes exige de totaliser un nombre donné d'infirmières ayant une compétence donnée, travaillant à un jour et à une période donnés. Par exemple, une contrainte de quota pourrait exiger qu'il y ait 5 infirmières de compétence 3 tous les lundis matin de l'horizon.

1.1.1 L'objectif

L'objectif consiste à affecter à chaque infirmière un horaire qui respecte ses contraintes intrinsèques. On nommera par la suite ces horaires comme suit : horaires individuels. Ensuite, l'ensemble de ces horaires individuels pris ensemble doit satisfaire les contraintes de quotas. Le terme horaire général sera utilisé par la suite pour désigner un ensemble d'horaires individuels.

Ainsi, l'objectif est de trouver un horaire général satisfaisant toutes les contraintes (contraintes intrinsèques et contraintes de quotas). Chaque horaire général généré devra être réalisable. Mais en plus, parmi l'ensemble de tous les horaires généraux réalisables, il suffit de trouver le meilleur, soit celui qui minimise la fonction objectif.

1.1.2 Les contraintes

Les contraintes du problème se répartissent sur deux niveaux : (1) les contraintes de quotas et (2) les contraintes intrinsèques à une infirmière.

En premier lieu, les contraintes de quotas s'appliquent à l'horaire général. Une telle contrainte s'applique sur un jour de l'horizon et sur une période de la journée (p. ex. : matin, midi, soir ou nuit) et demande un "quota cible" d'infirmières travaillant sur cette paire jour-période. De plus, chaque contrainte de quotas possède un ensemble de compétences. Pour une contrainte de quotas considérée, les infirmières qui travaillent sur la paire jour-période correspondante doivent avoir l'une des compétences de l'ensemble de compétences de cette contrainte afin d'apporter sa contribution à la satisfaction de cette dernière. Enfin, chaque contrainte de quotas accepte un déficit et un surplus maximaux sur le quota cible demandé.

Il y a deux types de contraintes de quotas : A et B. Celles de type A correspondent à celles décrites par Labit [3] et nous y reviendrons lors de la description du problème NSP1. Celles de type B, qui n'ont jamais été présentées auparavant, seront décrites à la section réservée au problème NSP2.

En deuxième lieu, les contraintes intrinsèques à une infirmière s'appliquent à l'horaire individuel de cette même infirmière. Chacune de ces contraintes est soit dure ou soit molle. Celles qui sont dures doivent absolument être satisfaites dans l'horaire affecté à l'infirmière. Tandis que le respect des contraintes molles n'est pas nécessaire, le non-respect de celles-ci affecte le poids de cet horaire dans la fonction objectif.

Comme nous allons le voir dans les différents chapitre de ce mémoire, les contraintes de quotas (premier niveau) sont définies dans le PLNE (P) (Problème Linéaire en Nombres Entiers) alors que le respect des contraintes intrinsèques à une infirmière relève du problème auxiliaire. Le PLNE (P) sera défini à la section 1.3.3 et le problème auxiliaire sera introduit à la section 2.3.2.2 et entièrement expliqué au chapitre 4.

Premièrement, commençons par énumérer les contraintes intrinsèques, à une infirmière, dites dures. Celles-ci étant nombreuses, nous allons donner les plus importantes :

1. contrainte sur la *"charge de travail"* qui impose un nombre total d'heures travaillées pour une période de l'horizon donnée,
2. contrainte sur la *"rotation entre les fins de semaines travaillées et non-travaillées"*,
3. contrainte sur la *"rotation entre les différents types de quarts travaillés"*,

4. contrainte comptabilisant les *"jours de vacances"* accordés à l'infirmière.

L'horizon est divisé en plusieurs périodes qui forment une partition de celui-ci. Une période est composée d'une date de début et d'une date de fin. Ce sont ces périodes auxquelles on fait référence pour la contrainte sur la *charge de travail.*

Aussi, il est important de faire la distinction entre une journée de vacances et une journée de repos. La première est un jour de congé payé qui est accordé à l'infirmière. Cette journée de vacances est comptabilisée dans la charge de travail de la période à laquelle le jour considéré appartient. Ensuite, une journée de repos est un jour où l'infirmière ne travaille pas et n'est pas payée. Cette journée n'est donc pas considérée dans la charge de travail de l'infirmière.

Deuxièmement, les contraintes dites molles se résument aux préférences et aux aversions exprimées par l'infirmière. Les préférences (respectivement aversions) consistent en un ensemble d'affectations (jour de l'horizon, quart de travail) avec un poids associé à chacun pour indiquer le désir de l'infirmière de (respectivement ne pas) travailler durant cette affectation.

Finalement, pour toutes les contraintes du deuxième niveau, soit celles qui sont intrinsèques à une infirmière, nous référons le lecteur à Labit [3] pour connaître leur description exacte et leur impact respectif dans le modèle. De plus, la référence de base avec laquelle Labit s'est inspirée pour décrire ces contraintes est la convention collective du personnel infirmier de l'Hôpital Royal Victoria (voir [4]).

1.2 Revue des méthodes existantes

De nombreux auteurs ont proposé différentes méthodes pour résoudre le problème de confection d'horaires d'infirmières. Toutes ces méthodes se classent en deux écoles de pensées totalement opposées et distinctes. La première école renferme les méthodes dites "exactes" et la deuxième regroupe les méthodes dites "heuristiques".

Premièrement, les méthodes exactes sont celles ayant comme finalité de trouver une solution de valeur minimale et de prouver mathématiquement que cette valeur est optimale. En d'autres termes, la meilleure solution trouvée par ces méthodes doit être celle ayant la meilleure valeur parmi l'ensemble de toutes les solutions possibles du problème considéré.

Deuxièmement, les méthodes heuristiques ont l'objectif de trouver une solution dont la valeur est considérée minimale selon leurs propres critères, mais ils n'ont pas la tâche de prouver l'optimalité de cette solution. La plupart de ces méthodes consistent simplement à trouver une première solution ou une solution de qualité relativement bonne.

Afin de choisir judicieusement l'une ou l'autre des méthodes pouvant résoudre le NSP, il est indispensable de connaître nos besoins. Lorsque l'utilisation prévue de la méthode est de générer un ou plusieurs horaires réalisables avec un bon rapport qualité/temps, le choix se porte généralement sur une méthode heuristique. Ceci est généralement le cas dans le domaine de la santé, du transport, de l'aviation, etc. Toutefois, il peut exister quelques cas où la connaissance de la valeur optimale d'un problème donné est désirée et c'est ici qu'une méthode exacte peut être d'un grand apport. Par exemple, lors du développement d'une méthode heuristique, la

connaissance de cette valeur optimale peut être utilisée pour évaluer la qualité de la solution de valeur minimale et pour guider au développement de stratégies améliorant cette heuristique.

1.2.1 Travaux précurseurs de notre méthode exacte

L'approche que nous proposons pour résoudre le NSP est une méthode exacte. Elle est l'aboutissement de sept ans d'études et du jumelage des travaux de Jaumard, Semet et Vovor [2], de ceux de Vovor [1] et de ceux de Labit [3].

Tout d'abord, l'article de Jaumard, Semet et Vovor introduit le modèle de génération de colonnes en variables entières, soit la première version de notre PLNE (P). Aussi, il décrit le problème auxiliaire et son algorithme de plus court chemin avec contraintes de ressources.

Ensuite, la thèse de doctorat de Vovor [1] porte essentiellement sur le développement de l'algorithme de plus court chemin avec contraintes de ressource. Son algorithme pseudo-polynomial est basé sur la programmation dynamique et sur une approche en deux phases. Cette décomposition permet de traiter efficacement les cas de réoptimisation, en particulier lorsque le coût des arcs changent. De plus, il a démontré mathématiquement l'exactitude de son algorithme en plus d'analyser sa complexité. Il a aussi fait la modélisation de ressources pour adapter son algorithme au personnel soignant.

Enfin, le mémoire de maîtrise de Labit [3] a repris la modélisation de ressources entamée par Vovor [1] afin de répondre à un niveau élevé de la modélisation du

problème. En outre, il a défini de façon approfondie les quatres ressources nécessaires à la modélisation des quatres contraintes intrinsèques décrites à la section 1.1.2.

L'ensemble de ces travaux a donné naissance à la première version du programme Iris. Toutefois, il s'est révélé que la méthode n'était pas exacte par cause du manque de contrôle dans le noyau dur gérant l'algorithme de branch and price, soit la librairie ABACUS [5].

Malheureusement, la seule méthode exacte connue par l'auteur est celle que nous présentons dans le présent mémoire.

1.2.2 Méthodes heuristiques

Une des premières méthodes heuristiques proposée pour la génération d'horaires non cycliques est celle de Warner [6]. Dans son approche, les horaires réalisables sont ceux qui respectent les demandes définies sur l'horizon en plus de satisfaire la charge de travail des infirmières. Son approche consiste tout d'abord à trouver une première solution réalisable. Ensuite, à l'aide de l'application d'une recheche locale où chaque itération tente d'améliorer la valeur de la solution précédente, une solution de valeur minimale est trouvée. Toutefois, il est important de préciser que cette approche ne considère pas les affectations portant sur les jours de fin de semaine étant données qu'elles sont pré-définies à l'avance. Évidemment, ceci simplifie de beaucoup la difficulté du problème.

Dowsland [7] a proposé une approche de type recherche tabou pour résoudre le NSP. Sa méthode nécessite un pré-traitement qui a la tâche d'énumérer et d'évaluer les horaires des infirmières. Ensuite, l'optimisation se fait par l'interaction entre

deux phases. La première phase a pour objectif de définir un horaire général qui est réalisable au niveau du respect des contraintes de quotas et la seconde phase tend à minimiser le poids de l'horaire global. L'avantage de cette méthode réside dans la modélisation utilisée car elle permet d'inclure la majorité des règles ergonomiques. Toutefois, l'inconvénient majeur réside dans l'énumération des horaires qui est dispendieux en termes de temps et d'espace en plus de limiter à une ou deux semaines la taille de l'horizon de planification.

Abdennadher et Schenker [8] proposent une méthode basée sur la programmation par contraintes et l'intelligence artificielle. Leur algorithme est divisé en trois phases : affectation des jours de repos, affectation des quarts de nuits et affectation des quarts de jour. Toutefois, une différence majeure entre cette méthode et les autres existantes est la nécessité de l'intervention de l'utilisateur pour résoudre certains conflits dans les horaires générés. Malgré tout, le nombre d'infirmières considérées est de 20 et la taille de l'horizon de planification est de 4 semaines.

1.3 Description et modélisation du problème NSP1

L'objectif de cette section est de survoler rapidement le problème NSP1. Premièrement, nous commençons pas décrire les données qui sont fournies pour la résolution du problème. Ensuite, un exemple de petite dimension est donné pour aider à la compréhension de la problématique. Finalement, la modélisation mathématique du problème en un programme en nombres entiers est décrite en profondeur.

1.3.1 Description du problème

Les données du problème se répartissent sur deux niveaux : (1) les caractéristiques générales du problème et (2) les caractéristiques relatives à une infirmière donnée.

Les données du premier niveau sont par exemple : le nombre d'infirmières auquel un horaire doit être affecté, les dates de début et de fin de l'horizon de planification, les contraintes de quotas à satisfaire, etc. Ensuite, les données du deuxième niveau sont par exemple : le nom de l'infirmière, la compétence qu'elle possède, les quarts de travail que l'on peut lui affecter, des données relatives aux quatre contraintes intrinsèques dites dures (voir la section 1.1.2), etc.

Pour avoir plus de détails sur les données du deuxième niveau, nous référons le lecteur au mémoire de Labit [3]. Maintenant, voici les données qui sont nécessaires à la compréhension du présent mémoire.

- Les infirmières indicées k. Chacune d'entre elles possède des caractéristiques et des contraintes intrinsèques (comme des disponibilités, des préférences personnelles, une ou plusieurs compétences, ...). On notera dans la suite $\mathcal{K}$ l'ensemble des indices d'infirmières [3] .

- L'horizon de planification. Il s'agit ici de l'ensemble des jours pour lesquels on veut générer un horaire pour chaque infirmière. Cet ensemble est noté par $\mathcal{D} = \{ d_1, d_2, ..., d_{|\mathcal{D}|} \}$, où d_1 et $d_{|\mathcal{D}|}$ correspondent respectivement au premier et au dernier jour de l'horizon. Pour désigner un jour quelconque, aucun indice ne sera utilisé et ce jour sera noté par $d \in \mathcal{D}$.

- Les quarts de travail que l'on peut affecter dans l'horaire de chaque infirmière et pour chaque jour de l'horizon courant. Un jour où aucun quart de travail n'est affecté est un jour de repos ou de vacance. L'ensemble des quarts de travail est noté par $\mathcal{T} = \{ t_1, t_2, ..., t_{|\mathcal{T}|} \}$ et est commun à tous les jours de l'horizon. Pour désigner un quart quelconque, aucun indice ne sera utilisé et ce quart sera noté par $t \in \mathcal{T}$.

[3] Pour désigner l'infirmière d'indice $k \in \mathcal{K}$, nous dirons l'infirmière $k \in \mathcal{K}$.

- On note $\mathcal{T}_k \subseteq \mathcal{T}$ l'ensemble des quarts de travail possibles pour l'infirmière k.

- Les périodes forment une partition de la journée telle que chaque quart couvre au moins une période. Les périodes servent à exprimer les contraintes de quotas et une constante (notée α_{tp}) sera utilisée pour assurer la correspondance quart-période (cette relation est décrite en détail ci-dessous). L'ensemble des périodes est noté par $\mathcal{P} = \{ p_1, p_2, ..., p_{|\mathcal{P}|} \}$. Pour désigner une période quelconque, aucun indice ne sera utilisé et cette période sera notée par $p \in \mathcal{P}$.

- Un ensemble de compétences $\Lambda = \{ \ell_1, \ell_2, \ldots, \ell_{|\Lambda|} \}$ qui servira à la fois à caractériser les infirmières et à exprimer les contraintes de quotas.
  - On notera $\Lambda_k \subseteq \Lambda$ l'ensemble des compétences de l'infirmière k [4].
  - $\mathcal{L}_{dp}$ un ensemble d'ensembles de compétences. C'est l'ensemble de tous les ensembles de compétences distincts exprimés pour les contraintes de quotas de type A (définies ci-dessous) portant sur la paire $(d, p) \in \mathcal{D}_{\mathcal{P}}$. On notera dans la suite $\mathcal{L}_{dp} = \{L_1, L_2, \ldots, L_{|\mathcal{L}_{dp}|}\}$, où $L \in \mathcal{L}_{dp}$ est tel que $L \subseteq \Lambda$.

- Un ensemble de contraintes de quotas de type A. Chacune d'entres elles est définie sur une paire $(d, p) \in \mathcal{D}_{\mathcal{P}}$ et un ensemble de compétences $L \in \mathcal{L}_{dp}$. Une telle contrainte est notée par (Q_L) et est associée au triplet suivant : (q_L, q_L^-, q_L^+).
  - L spécifie lesquelles des compétences sont acceptées pour la contrainte. Ainsi, chaque infirmière $k \in \mathcal{K}$ comptabilisée dans la contrainte doit avoir exactement une compétence élément de L : $|L \bigcap \Lambda_k| = 1$.
  - $d \in \mathcal{D}$ représente le jour et $p \in \mathcal{P}$ la période où la contrainte de quotas doit être respectée.
  - q_L représente le nombre d'infirmières requis (quota cible). q_L^- et q_L^+ sont respectivement le déficit et le surplus maximaux acceptés pour satisfaire cette contrainte de quotas. Ainsi, sa fenêtre de réalisabilité est $[\, q_L - q_L^- \,,\, q_L + q_L^+ \,]$, avec q_L comme quota cible.

[4] Pour NSP1, $|\Lambda_k| = 1$. Alors que pour NSP2, $|\Lambda_k| \geq 1$.

- Des pénalités pour le dépassement (*overstaffing*) et le manque de personnel (*understaffing*) permettent de contrôler le respect des quotas au plus serré. En effet, lorsque le quota cible q_L est dépassé par un surplus de s_L^+ (où $0 \leq s_L^+ \leq q_L^+$), une pénalité de p^+ est accordée à chaque unité de ce surplus afin de le minimiser. Le même principe s'applique lorsque le quota cible q_L n'est pas franchi par cause d'un manque de s_L^- (où $0 \leq s_L^- \leq q_L^-$), une pénalité de p^- est accordée à chaque unité de ce déficit afin de le minimiser [5].

Note sur les contraintes de quotas

Il est important de noter que pour deux contraintes de quotas distinctes $(Q_{L'})$ et $(Q_{L''})$, qui sont définies sur une même paire $(d, p) \in \mathcal{D}_\mathcal{P}$ ($L', L'' \in \mathcal{L}_{dp}$ et $L' \neq L''$), les données sont telles que $L' \bigcap L'' = \emptyset$.

Relation quart-période

Un jour de l'horizon est partitionné en un ensemble de périodes noté $\mathcal{P}$. Chaque quart de travail de l'ensemble $\mathcal{T}$ débute au début d'une période et termine à la fin de cette même période ou à la fin d'une période subséquente à celle-ci. Par conséquent, tout quart de travail couvre au moins une période de l'ensemble $\mathcal{P}$. Un exemple de cette relation quart-période est donné à la figure 1.1.

1.3.2 Présentation d'un exemple

Pour s'assurer de bien comprendre le problème NSP1, nous présentons un exemple en décrivant ses données, tous les horaires possibles pour chacune de ses infirmières et toutes ses solutions possibles, incluant bien sûr sa solution optimale.

[5] Nous verrons plus loin que s_L^+ et s_L^- sont des variables de notre modèle.

1.3.2.1 Les données de l'exemple

Nous décrivons ci-dessous les données de notre exemple. Comme nous l'avons dit précédemment, les données du deuxième niveau ne sont pas nécessaires à la compréhension du présent mémoire. C'est pour cette raison qu'elles ne sont pas fournies dans cet exemple. Les données du premier niveau sont décrites ci-dessous.

- L'ensemble des infirmières : $\mathcal{K} = \{k_1,\ k_2,\ k_3\}$.
- L'horizon est composé de 3 jours : $\mathcal{D} = \{d_1,\ d_2,\ d_3\}$, où d_1 représente le lundi 08/09/2003, d_2 le mardi 09/09/2003 et d_3 le mercredi 10/09/2003.
- L'ensemble des quarts de travail : $\mathcal{T} = \{t_1,\ t_2,\ t_3,\ t_4\}$.
- L'ensemble des périodes d'une journée : $\mathcal{P} = \{p_1,\ p_2,\ p_3\}$.
- L'ensemble des compétences : $\Lambda = \{\ell_1,\ \ell_2,\ \ell_3\}$.
- Pénalité par unité de déficit et de surplus accordée à chaque contrainte de quotas : respectivement $p^- = 1$ et $p^+ = 1$.
- Les contraintes de quotas qui sont au nombre de 5 et qui sont décrites au tableau 1.1. Pour chaque contrainte de quotas (Q_L) $(L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_{\mathcal{P}})$ de ce tableau, l'ensemble de compétences L et le triplet (q_L, q_L^-, q_L^+) sont donnés.
- Certaines caractéristiques des infirmières k_1, k_2 et k_3.
  - L'ensemble des quarts pouvant être affectés à l'infirmière k_1 : $\mathcal{T}_{k_1} = \{t_1,\ t_2\}$. Elle possède l'ensemble de compétences suivant : $\Lambda_{k_1} = \{\ell_1\}$.
  - Pour l'infirmière k_2 nous avons $\mathcal{T}_{k_2} = \{t_1,\ t_3\}$ et $\Lambda_{k_2} = \{\ell_2\}$.
  - Pour l'infirmière k_3 nous avons $\mathcal{T}_{k_3} = \{t_1,\ t_4\}$ et $\Lambda_{k_3} = \{\ell_3\}$.

Ensuite, la relation entre l'ensemble de quarts $\mathcal{T}$ et l'ensemble de périodes $\mathcal{P}$ est définie par la figure 1.1. On remarque que le quart de travail t_1 couvre la période p_1 alors que le quart t_2 couvre les deux périodes p_1 et p_2.

Tableau 1.1: Les contraintes de quotas définies sur l'horizon.

	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
p_1	$(Q_{L_1}) : (\{\ell_1\},\ (1,0,0))$		$(Q_{L_1}) : (\{\ell_1\},\ (1,0,0))$ $(Q_{L_2}) : (\{\ell_2,\ \ell_3\},\ (2,1,0))$
p_2	$(Q_{L_1}) : (\{\ell_1,\ell_2\},\ (2,0,0))$		
p_3		$(Q_{L_1}) : (\{\ell_3\},\ (1,0,0))$	

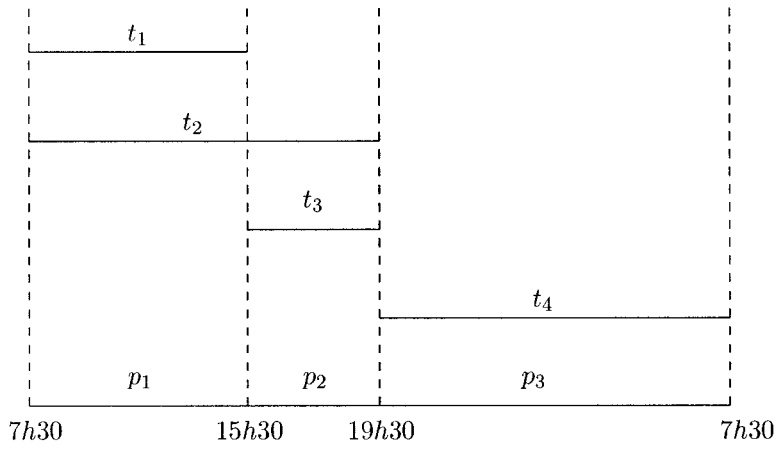


Figure 1.1: Exemple de relation quart-période.

Finalement, pour s'assurer de bien comprendre les contraintes de quotas de type A, nous allons décrire celle qui est définie pour la paire (d_1, p_2) et les deux qui sont définies pour la paire (d_3, p_1).

Premièrement, la seule contrainte qui est définie sur (d_1, p_2) est (Q_{L_1}), où $L_1 = \{\ell_1, \ell_2\}$, $(q_{L_1}, q^-_{L_1}, q^+_{L_1}) = (2,0,0)$ et $L_1 \in \mathcal{L}_{d_1 p_2}$. Comme la contrainte (Q_{L_2}) définie sur (d_3, p_1), celle-ci accepte plus qu'une compétence. En effet, toute infirmière comptabilisée dans cette contrainte doit posséder soit la compétence ℓ_1 ou soit la compétence ℓ_2.

Ensuite, concernant la paire (d_3, p_1), il y a les deux contraintes de quotas (Q_{L_1}) et (Q_{L_2}) de définies. La contrainte (Q_{L_1}), où $L_1 = \{\ell_1\}$, $(q_{L_1}, q^-_{L_1}, q^+_{L_1}) = (1, 0, 0)$ et $L_1 \in \mathcal{L}_{d_3 p_1}$, exige qu'il y ait exactement une infimière de compétence ℓ_1 qui travaille sur cette même paire. La contrainte (Q_{L_2}), où $L_2 = \{\ell_2,\ \ell_3\}$, $(q_{L_2}, q^-_{L_2}, q^+_{L_2}) = (2, 1, 0)$ et $L_2 \in \mathcal{L}_{d_3 p_1}$, demande deux infirmières de compétence ℓ_2 ou ℓ_3. Mais, contrairement à (Q_{L_1}), elle accepte un déficit de 1. Evidemment, ce déficit entraînerait une pénalité totale de valeur égale à 1 dans la fonction objectif. Notons qu'aucune des trois infirmières ne peut avoir un apport dans ces deux contraintes car les données sont telles que $L_1 \bigcap L_2 = \emptyset$ et $|\Lambda_k| = 1$ (pour le NSP1).

1.3.2.2 Les horaires existants pour les trois infirmières

Comme nous l'avons déjà mentionné, chacune des infirmières possède des contraintes intrinsèques dures et molles. Pour diminuer la taille de notre exemple, on peut supposer que l'ensemble des contraintes dures pour chacune des trois infirmières est très restrictif, de façon à ce qu'il y ait un petit nombre d'horaires possibles pour ces trois infirmières : 2 pour les infirmières k_1 et k_3 et 1 pour l'infirmière k_2. Tous ces horaires sont définis au tableau 1.2, où $S_{k_1} = \{s_1,\ s_2\}$, $S_{k_2} = \{s_1\}$ et $S_{k_3} = \{s_1,\ s_2\}$ sont respectivement l'ensemble des horaires possibles pour les infirmières k_1, k_2 et k_3.

Le coût des horaires des trois infirmières a été défini selon les contraintes intrinsèques dites molles. Donc, d'après le tableau 1.2, nous pouvons conclure que l'infirmière k_3 préférerait être affectée à l'horaire s_2 plutôt qu'à l'horaire s_1. Pour l'infirmière k_1, les deux horaires ont un coût identique.

Tableau 1.2: Les horaires possibles pour les infirmières k_1, k_2 et k_3.

	Λ_k	S_k	coût	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
k_1	$\{\ell_1\}$	s_1	0	t_2		t_1
		s_2	0	t_2	t_1	
k_2	$\{\ell_2\}$	s_1	0	t_3		t_1
k_3	$\{\ell_3\}$	s_1	2		t_4	t_1
		s_2	0		t_4	

1.3.2.3 Les solutions possibles

D'après le tableau 1.2, il y a 4 horaires généraux possibles. Par contre, seulement 2 d'entres eux respectent les contraintes de quotas définies au tableau 1.1.

Le premier horaire général consiste à affecter aux trois infirmières k_1, k_2 et k_3 leur horaire respectif s_1 (noté $(k_1 : s_1,\ k_2 : s_1,\ k_3 : s_1)$). Nous pouvons voir au tableau 1.3 la comptabilisation des quotas pour cet horaire général. Pour chaque contrainte de quotas (Q_L), le nombre total d'infirmières comptabilisées dans celle-ci est indiqué. D'ailleurs, toutes les contraintes de quotas ont atteint leur quota cible. Ainsi, par cause de l'horaire s_1 affecté à k_3, la valeur de cet horaire général est de 2.

Tableau 1.3: Comptabilisation des quotas pour l'horaire $(k_1 : s_1,\ k_2 : s_1,\ k_3 : s_1)$ de valeur 2.

	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
p_1	$(Q_{L_1}) : 1$		$(Q_{L_1}) : 1$ $(Q_{L_2}) : 2$
p_2	$(Q_{L_1}) : 2$		
p_3		$(Q_{L_1}) : 1$	

Finalement, le deuxième horaire général est $(k_1 : s_1,\ k_2 : s_1,\ k_3 : s_2)$. Nous pouvons voir au tableau 1.4 la comptabilisation des quotas pour ce deuxième horaire. La valeur

de cet horaire général est de 1, par cause du déficit de valeur égale de la contrainte (Q_{L_2}) en (d_3, p_1). Par conséquent, la valeur optimale de notre problème est 1 et l'horaire $(k_1 : s_1,\ k_2 : s_1,\ k_3 : s_2)$ est une solution optimale.

Tableau 1.4: Comptabilisation des quotas pour l'horaire $(k_1 : s_1,\ k_2 : s_1,\ k_3 : s_2)$ de valeur 1.

	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
p_1	$(Q_{L_1}) : 1$		$(Q_{L_1}) : 1$ $(\mathbf{Q_{L_2}}) : \mathbf{1}$
p_2	$(Q_{L_1}) : 2$		
p_3		$(Q_{L_1}) : 1$	

1.3.3 Modélisation mathématique

Ici, nous allons présenter le modèle mathématique du problème NSP1, soit le PLNE (P). Ce dernier donnera naissance au programme linéaire (R), qui sera utilisé par l'algorithme de branch and price et qui sera présenté au prochain chapitre.

Comme nous le verrons à la section 1.4.2 et au chapitre 2, l'algorithme de branch and price ne considère par toutes les variables (ou horaires) valides pour une infirmière donnée $k \in \mathcal{K}$. Toutefois, pour des fins de modélisation, on note par S_k l'ensemble des indices des horaires réalisables pour l'infirmière $k \in \mathcal{K}$. Même si cet ensemble n'est pas connu, nous nous permettons de l'utiliser dans notre modèle (P). De plus, il faut aussi rappeler au lecteur que tous les horaires $s \in S_k$ de l'infirmière k respectent chaque contrainte intrinsèque de cette dernière [6].

Nous commençons par décrire les constantes utilisées par le modèle (P). Deuxièmement, ses variables sont données. Troisièmement, nous expliquons la modélisation

[6]Pour désigner l'horaire d'indice $s \in S_k$ de l'infirmière k, nous dirons l'horaire $s \in S_k$.

mathématique des contraintes. Ensuite, les composantes de sa fonction objectif sont expliquées. Finalement, le modèle (P) est donné dans son ensemble.

1.3.3.1 Les constantes du PLNE (P)

Voici la description des constantes du modèle (P). La première constante est calculée par le problème auxiliaire tandis que les autres sont extraites des données.

- Un coefficient $p_{ks} \in \mathbb{N}$ indiquant le poids de l'horaire $s \in S_k$ de l'infirmière $k \in \mathcal{K}$. Essentiellement, le poids d'un horaire dépend du respect des préférences et des aversions qui sont indiquées par l'infirmière. Les détails de son calcul sont décrits dans Labit [3].
- Deux coefficients, $p^- \in \mathbb{N}$ et $p^+ \in \mathbb{N}$, reliés aux contraintes de quotas. Ils indiquent la pénalité accordée par unité d'écart, de déficit pour la première et de surplus pour la deuxième, avec le quota cible (q_L) de chaque contrainte de quotas [7].
- Chaque quart de travail couvre en entier une ou plusieurs périodes. Ainsi, nous devons définir la constante suivante pour chaque paire (t, p) (où $t \in \mathcal{T}$ et $p \in \mathcal{P}$) :

$$\alpha_{tp} = \begin{cases} 1 & \text{si le quart } t \text{ couvre la période } p, \\ 0 & \text{sinon.} \end{cases}$$

- Pour chacune des paires (k, L), où $k \in \mathcal{K}$ et $L \in (\bigcup_{(d,p)\in\mathcal{D}_\mathcal{P}} \mathcal{L}_{dp})$, nous devons définir une constante binaire pour indiquer si l'infirmière k peut apporter sa contribution à une contrainte de quotas de type A définie sur L. Cette constante est définie selon les compétences de l'infirmière et ceux de l'ensemble L comme suit :

$$b_{kL} = \begin{cases} 1 & \text{si } |L \bigcap \Lambda_k| \geq 1, \\ 0 & \text{sinon.} \end{cases}$$

[7] Voir la section 1.3.1.

1.3.3.2 Les variables du PLNE (P)

Maintenant, voyons en détail les variables de notre modèle.

- Pour chaque horaire $s \in S_k$ de l'infirmière $k \in \mathcal{K}$, nous devons définir une variable de décision pour indiquer si cet horaire est affecté à l'infirmière k dans l'horizon courant. Cette variable x_{ks} est définie comme suit :

$$x_{ks} = \begin{cases} 1 & \text{si l'infirmière } k \text{ est affectée à l'horaire } s, \\ 0 & \text{sinon.} \end{cases}$$

- Pour chaque quadruplet (k, s, d, t) (où $k \in \mathcal{K}$, $s \in S_k$ et $(d, t) \in \mathcal{D}_{\mathcal{T}_k}$), nous devons définir une variable nous indiquant si l'affectation (d, t) a été donnée à l'infirmière k dans son horaire s, qui a précédemment été généré par le problème auxiliaire. Par conséquent, cette variable, qui est propre au problème auxiliaire, est en réalité une constante pour le PLNE (P). Elle est notée a_{ksdt} et est définie comme suit :

$$a_{ksdt} = \begin{cases} 1 & \text{si l'infirmière } k \text{ est affectée à } (d, t) \text{ dans l'horaire } s, \\ 0 & \text{sinon.} \end{cases}$$

- Pour chaque contrainte de quotas (Q_L), où $L \in \mathcal{L}_{dp}$ et $(d, p) \in \mathcal{D}_{\mathcal{P}}$, nous devons définir deux variables d'écarts s_L^- et s_L^+. Elles indiquent l'écart avec le quota cible q_L, de déficit pour la première et de surplus pour la seconde, comptabilisé dans l'horaire général.

1.3.3.3 Les contraintes du PLNE (P)

Comme nous l'avons dit auparavant, le problème maître s'occupe de satisfaire les contraintes de quotas tandis que le problème auxiliaire a la tâche de générer des horaires qui satisfont les contraintes intrinsèques de l'infirmière. Donc, aucune

contrainte dans notre modèle n'est nécessaire pour modéliser ces contraintes intrinsèques.

- Pour chaque contrainte de quotas (Q_L) (où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_{\mathcal{P}}$) nous avons une contrainte pour le respect du quota cible demandé (q_L), où y sont insérées les variables d'écarts s_L^- et s_L^+. Deux autres contraintes sont imposées sur ces variables pour le respect du déficit (q_L^-) et du surplus (q_L^+) maximaux permis.
  - $\sum_{k \in \mathcal{K}} \sum_{s \in S_k} \sum_{t \in \mathcal{T}_k} \alpha_{tp}\, b_{kL}\, a_{ksdt}\, x_{ks} + s_L^- - s_L^+ = q_L$
  - $s_L^- \in \{0, 1, ..., q_L^-\}$
  - $s_L^+ \in \{0, 1, ..., q_L^+\}$

  Ici, il est important de noter que pour une même paire $(d,p) \in \mathcal{D}_{\mathcal{P}}$, l'affectation a_{ksdt} peut compter dans au plus une contrainte de type A (Q_L), où $L \in \mathcal{L}_{dp}$ et $b_{kL} = 1$. En effet, soit $L', L'' \in \mathcal{L}_{dp}$ où $L' \neq L''$, alors nous ne pouvons avoir $b_{kL'} = 1$ et $b_{kL''} = 1$ car les données sont telles que $L' \bigcap L'' = \emptyset$ et $|\Lambda_k| = 1$.

- Pour chaque infirmière $k \in \mathcal{K}$, nous avons deux contraintes pour indiquer que, dans l'horaire général, l'infirmière doit n'être affectée qu'à un seul horaire $s \in S_k$. La première se nomme *contrainte de partitionnement* et la deuxième *contrainte d'intégralité* :
  - $\sum_{s \in S_k} x_{ks} = 1,$
  - $x_{ks} \in \{0, 1\},\ s \in S_k.$

1.3.3.4 La fonction objectif du PLNE (P)

La fonction objectif à minimiser est composée de deux parties. La première consiste en la somme des poids des horaires affectés aux infirmières. La deuxième représente la somme des pénalités imposées aux déficits (p^-) ou aux surplus (p^+) sur chaque contrainte de quotas (Q_L), où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_{\mathcal{P}}$.

$$\min \Big(\sum_{k \in \mathcal{K}} \sum_{s \in S_k} p_{ks}\ x_{ks} + \sum_{(d,p) \in \mathcal{D}_\mathcal{P}} \sum_{L \in \mathcal{L}_{dp}} (p^-\ s_L^- + p^+\ s_L^+) \Big)$$

Aussi, étant donné que les coefficients p_{ks}, p^- et p^+ sont entiers, les deux composantes de la fonction objectif sont aussi de valeur entière. Par conséquent, toute solution du PLNE (P) est nécessairement de valeur entière.

Finalement, nous montrons à la figure 1.2 le PLNE (P). Ce modèle mathématique est composé de la fonction objectif, des contraintes de quotas de type A, des contraintes de partitionnement et des contraintes d'intégralité sur les variables x_{ks}, s_L^- et s_L^+.

$$\min \Big(\sum_{k \in \mathcal{K}} \sum_{s \in S_k} p_{ks}\ x_{ks} + \sum_{(d,p) \in \mathcal{D}_\mathcal{P}} \sum_{L \in \mathcal{L}_{dp}} (p^-\ s_L^- + p^+\ s_L^+) \Big)$$

s.l.c. :

$$\sum_{k \in \mathcal{K}} \sum_{s \in S_k} \sum_{t \in \mathcal{T}_k} \alpha_{tp}\ b_{kL}\ a_{ksdt}\ x_{ks} + s_L^- - s_L^+ = q_L \quad (d,p) \in \mathcal{D}_\mathcal{P},\ L \in \mathcal{L}_{dp}$$

$$\sum_{s \in S_k} x_{ks} = 1 \qquad k \in \mathcal{K}$$

$$x_{ks} \in \{0,\ 1\} \qquad k \in \mathcal{K},\ s \in S_k$$

$$s_L^- \in \{0,\ 1,\ \ldots,\ q_L^-\} \qquad (d,p) \in \mathcal{D}_\mathcal{P},\ L \in \mathcal{L}_{dp}$$

$$s_L^+ \in \{0,\ 1,\ \ldots,\ q_L^+\} \qquad (d,p) \in \mathcal{D}_\mathcal{P},\ L \in \mathcal{L}_{dp}$$

Figure 1.2: Formulation du problème NSP1 en un PLNE noté (P).

1.4 Méthode retenue

Pour résoudre efficacement ce problème, la méthode choisie est celle de la génération de colonnes. Elle a été suggérée en premier par Ford et Fulkerson [9] en 1958. Ensuite, elle a été présentée par Dantzig et Wolfe [10] (1960) pour les problèmes linéaires avec des structures décomposables. Finalement, Gilmore et Gomory [11] (1961) l'ont utilisée pour résoudre le problème du "cutting-stock". Une synthèse des principaux travaux portant sur la génération de colonnes est présentée par Soumis [12].

Dans cette section, nous commençons par expliquer les trois groupes d'algorithmes qui utilisent la méthode de génération de colonnes. Ensuite, nous expliquons notre algorithme : le branch and price.

1.4.1 Trois groupes d'algorithmes

La génération de colonnes a été très utilisée depuis longtemps dans des problèmes d'optimisation combinatoire, comme ceux reliés aux transports ou à la planification d'horaires de personnel. Elle permet aux modèles avec beaucoup de variables d'être résolus efficacement en considérant ces variables de façon implicite. Ainsi, seul un sous-ensemble de variables est généré à l'aide d'un problème auxiliaire nommé "le problème du pricing". La plupart des méthodes utilisent la génération de colonnes pour résoudre le problème linéaire, où la contrainte d'intégralité a été relaxée, pour ensuite appliquer un algorithme d'énumération implicite et trouver une bonne solution entière. Ces méthodes peuvent se classer en trois groupes.

Le premier groupe renferme les algorithmes où la génération de colonnes est utilisée de façon "off-line". Dans le sens où un certain nombre de colonnes est généré

pour ensuite résoudre le PLNE sur ce sous-ensemble de colonnes. Un exemple de cette approche se retrouve dans Hoffman and Padberg [13] (1993). Ensuite, les algorithmes du deuxième groupe utilisent une approche différente, soit la génération de colonnes dynamique. Elle consiste à utiliser la génération de colonnes pour résoudre optimalement la relaxation continue du PLNE. Ensuite, elle atteint la meilleure solution entière sur l'ensemble de colonnes générées précédemment en appliquant un algorithme d'énumération implicite. Cette approche a été utilisée pour résoudre le problème "pairage d'équipages pour les longs courriers" par Barnhart et al. [14] (1994).

Cependant, deux problèmes peuvent apparaître en utilisant l'une ou l'autre de ces deux méthodes. Il se peut que la solution trouvée soit très loin de la solution optimale. Mais, il est aussi possible qu'aucune solution ne soit trouvée, même avec un grand nombre de colonnes.

Finalement, notre méthode fait partie du troisième groupe. Elle consiste à appliquer la génération de colonnes dynamique au travers d'un algorithme d'énumération implicite, ce qu'on appelle plus communément le branch and price. Contrairement à la deuxième méthode, l'optimalité est prouvée non seulement au noeud racine mais également à tous les noeuds de notre arbre de branchement. De cette façon, nous sommes certains d'atteindre la solution entière optimale. Le désavantage réside dans le nombre de colonnes générées, l'espace mémoire et le temps requis, tous beaucoup plus grands que pour les deux autres méthodes.

Plusieurs travaux ont porté sur l'algorithme de branch and price. En outre, il y a ceux de Vanderbeck et Wolsey [15], de Barnhart et al. [16] et de Vance et al. [17]. De plus, les travaux de Vance et al. [18] proposent un branch and price heuristique afin de trouver une solution dont la valeur est le plus près possible de la valeur optimale. Leur application est le "Airline Crew Pairing Problem".

1.4.2 Le branch and price

Maintenant, nous allons voir un aperçu de la méthode branch and price. Tout d'abord, le PLNE (P) comportant des variables entières, on relaxe ces contraintes d'intégralité pour obtenir le nouveau problème linéaire relaxé (R). Ensuite, pour résoudre ce problème, il n'est pas nécessaire de considérer toutes les variables. Car, étant trop nombreuses, plusieurs d'entre elles auront une valeur nulle dans la solution optimale de (R). Ainsi, au lieu de générer toutes les variables hors base par énumération explicite, seulement celles d'entre elles qui ont un coût réduit négatif (ou celles ayant les coûts réduits les plus négatifs) sont générées. Ensuite, on les fait entrer en base. Le problème auxiliaire, qui sera décrit plus bas, a la tâche de générer ces nouvelles variables.

Par la suite, l'optimalité du problème relaxé est atteinte lorsqu'aucune nouvelle variable n'est générée par le problème auxiliaire. Dans ce cas, un branchement est effectué en imposant une contrainte aux variables du modèle relaxé. Le même principe est appliqué pour optimiser ce nouveau modèle. Et ainsi de suite jusqu'à ce que la solution optimale d'un noeud de l'arbre de branchement soit entière. Dans le cas où de nouvelles variables sont ajoutées au modèle, le problème relaxé est de nouveau optimisé, ce jusqu'à ce que l'optimalité soit atteinte.

Comme nous pouvons l'observer, la génération de colonnes est composée de deux niveaux : le problème maître et le problème auxiliaire. Le premier (le coordonnateur) s'occupe principalement de l'optimisation du problème relaxé, de vérifier l'intégralité des solutions et de générer une règle de branchement. Aussi, avec l'aide du problème auxiliaire, il vérifie l'optimalité des solutions et génère (si possible) de nouvelles colonnes de coûts réduits négatifs.

Le deuxième, le problème auxiliaire, a la tâche de générer une (ou des) colonne(s) (pour une infirmière donnée) de coût(s) réduit(s) négatif(s). De plus, il doit prendre en considération les contraintes de branchements. Ces contraintes doivent être compatibles avec le problème auxiliaire de façon à ce qu'on puisse le modifier efficacement à chaque branchement, sans toutefois changer sa structure. Ainsi, les colonnes qui ne sont pas réalisables selon ces contraintes ne seront pas retournées par le problème auxiliaire, qui lui, restera fini en termes de temps et d'espace, ceci pour tous les noeuds de l'arbre de branchement. Nous reviendrons sur le problème auxiliaire au chapitre 4.

Finalement, ceci résume en quoi consiste l'algorithme du branch and price et ses deux niveaux que sont le problème maître et le problème auxiliaire. Le modèle mathématique utilisé par cet algorithme et les détails de ce dernier, soit la méthode de génération de colonnes, les méthodes de séparation, les algorithmes d'exploration de l'arbre de recherche, etc., seront présentés au prochain chapitre.

1.5 Description et modélisation du problème NSP2

Maintenant, nous allons décrire le problème NSP2. Nous commençons par expliquer les modifications apportées au problème. Ensuite, un exemple sera donné pour s'assurer de bien comprendre les différences entre les deux problèmes NSP1 et NSP2. Enfin, le modèle mathématique de ce deuxième problème sera expliqué.

1.5.1 Description du problème

Les modifications définissant la deuxième version du problème de confection automatisée d'horaires d'infirmières sont les suivantes : (1) chaque contrainte de quotas

est définie sur un quart de travail $t \in \mathcal{T}$, (2) chaque infirmière $k \in \mathcal{K}$ peut avoir plusieurs compétences ($|\Lambda_k| \geq 1$) et (3) définition d'un nouveau type de contrainte de quotas.

Définition des contraintes de quotas sur une paire $(d,t) \in \mathcal{D}_\mathcal{T}$

Contrairement à NSP1 où les contraintes de quotas de type A sont définies sur une paire $(d,p) \in \mathcal{D}_\mathcal{P}$, ces mêmes contraintes sont définies sur une paire $(d,t) \in \mathcal{D}_\mathcal{T}$ dans NSP2. Suite à cette modification, l'ensemble $\mathcal{L}_{dp}$ pour $(d,p) \in \mathcal{D}_\mathcal{P}$ est remplacé par l'ensemble $\mathcal{L}_{dt}$ pour $(d,t) \in \mathcal{D}_\mathcal{T}$.

Enfin, le concept des périodes (l'ensemble $\mathcal{P}$) qui forment une partition de la journée n'est plus défini dans le problème NSP2 et la constante α_{tp} n'est donc plus utile.

Plusieurs compétences par infirmière

Contrairement à NSP1, où pour chaque infirmière $k \in \mathcal{K}$ nous avons $|\Lambda_k| = 1$, NSP2 ne limite pas à un le nombre de compétences d'une infirmière $k \in \mathcal{K} : \forall k \in \mathcal{K},\ |\Lambda_k| \geq 1$.

Il est important de préciser au lecteur que même si une infirmière $k \in \mathcal{K}$ peut avoir plus d'une compétence, toutes ses affectations $(d,t) \in \mathcal{D}_{\mathcal{T}_k}$ ne pourront couvrir qu'une et une seule contrainte (Q_L) (où $L \in \mathcal{L}_{dt}$ et $|L \bigcap \Lambda_k| \geq 1$). En effet, ses multiples compétences $\ell \in \Lambda_k$ ne lui permettent pas de couvrir simultanément deux (ou plus) contraintes $(Q_{L'})$ et $(Q_{L''})$ (où $L', L'' \in \mathcal{L}_{dt},\ L' \neq L'',\ |L' \bigcap \Lambda_k| \geq 1,\ |L'' \bigcap \Lambda_k| \geq 1$). Par contre, elles lui permettent de décider dans laquelle de ces contraintes son affectation (d,t) aura un apport.

Contrainte de quotas de type B

Dans le problème NSP2, en plus des contraintes de quotas de type A définies sur un triplet (d, t, L) (où $(d, t) \in \mathcal{D}_\mathcal{T}$ et $L \in \mathcal{L}_{dt}$) un nouveau type de contrainte de quotas apparaît : soit les contraintes de quotas de type B. Pour une paire $(d, t) \in \mathcal{D}_\mathcal{T}$, il est possible de définir une seule contrainte de quotas de type B notée $(\hat{Q}_{dt})$. Cette contrainte consiste à englober toutes les contraintes de type A (Q_L) (où $L \in \mathcal{L}_{dt}$), qui sont définies pour cette même paire (d, t), en faisant abstraction de leurs ensembles de compétences respectifs.

Ainsi, la contrainte $(\hat{Q}_{dt})$, à laquelle on associe le triplet $(\hat{q}_{dt},\ \hat{q}^-_{dt},\ \hat{q}^+_{dt})$, possède l'ensemble de compétences suivant : $\hat{L}_{dt} \leftarrow \bigcup_{L \in \mathcal{L}_{dt}} L$. Une description complète de cette contrainte est donnée ci-dessous.

- $\hat{L}_{dt}$ spécifie lesquelles des compétences sont acceptées pour la contrainte. Ainsi, chacune des affectations qui est comptabilisée dans une des contraintes de quotas (Q_L), où $L \in \mathcal{L}_{dt}$, est nécessairement comptabilisée dans la contrainte $(\hat{Q}_{dt})$.
- $d \in \mathcal{D}$ représente le jour et $t \in \mathcal{T}$ le quart où cette contrainte de quotas de type B doit être respectée.
- $\hat{q}_{dt}$ représente le nombre d'infirmières requis (quota cible). $\hat{q}^-_{dt}$ et $\hat{q}^+_{dt}$ sont respectivement le déficit et le surplus maximaux acceptés pour satisfaire cette contrainte de quotas. Ainsi, sa fenêtre de réalisabilité est $[\ \hat{q}_{dt} - \hat{q}^-_{dt},\ \hat{q}_{dt} + \hat{q}^+_{dt}\ ]$, avec $\hat{q}_{dt}$ comme quota cible.

1.5.2 Présentation d'un exemple

Maintenant, nous présentons un exemple pour le problème NSP2. Nous commençons par énumérer ses données. Nous décrivons ensuite tous les horaires possibles

pour chacune de ses infirmières et nous terminons en décrivant toutes ses solutions possibles, incluant sa solution optimale.

1.5.2.1 Les données de l'exemple

Comme pour l'exemple présenté à la section 1.3.2, nous ne présentons pour le présent exemple que les données du premier niveau. Rapidement, voici ces données : $\mathcal{K} = \{k_1,\ k_2,\ k_3\}$, $\mathcal{D} = \{d_1,\ d_2,\ d_3\}$, $\mathcal{T} = \{t_1\}$ et $\Lambda = \{\ell_1,\ \ell_2,\ \ell_3,\ \ell_4\}$. Les caractéristiques des infirmières sont décrites ci-dessous.

- Pour k_1, $\mathcal{T}_{k_1} = \{t_1\}$ et $\Lambda_{k_1} = \{\ell_1,\ \ell_2\}$.
- Pour k_2, $\mathcal{T}_{k_2} = \{t_1\}$ et $\Lambda_{k_2} = \{\ell_3\}$.
- Pour k_3, $\mathcal{T}_{k_3} = \{t_1\}$ et $\Lambda_{k_3} = \{\ell_4\}$.

Les contraintes de quotas sont définies par le tableau 1.5. Il y a 4 contraintes de quotas de type A et une seule de type B. Les deux contraintes de type A (Q_{L_1}) et (Q_{L_2}) définies sur (d_3, t_1) sont liées par une contrainte de type B ($\hat{Q}_{d_3t_1}$), qui est aussi définie sur cette paire. Notons que (Q_{L_1}) et (Q_{L_2}) ont toutes les deux un quota cible de 1 mais elles acceptent respectivement un déficit et un surplus tous deux de valeur maximale égale à 1. Par contre, grâce à la contrainte ($\hat{Q}_{d_3t_1}$), soit que les quotas obtenus pour les contraintes (Q_{L_1}) et (Q_{L_2}) sont tous les deux de valeur 1 ou soit qu'ils sont respectivement de valeur 0 et 2. En effet, la contrainte ($\hat{Q}_{d_3t_1}$), à laquelle sont associés l'ensemble de compétences $\hat{L} = \{\ell_1, \ell_2, \ell_3, \ell_4\}$ et le triplet $(2, 0, 0)$, oblige un quota final de 2 et n'accepte ni déficit ni surplus sur ce quota cible, qui représente le total des quotas obtenus pour les contraintes (Q_{L_1}) et (Q_{L_2}).

1.5.2.2 Les horaires existants pour les trois infirmières

Même si le problème NSP2 ne limite pas à un le nombre de compétences par infirmière, une affectation $(d, t) \in \mathcal{D}_{\mathcal{T}_k}$ d'une infirmière $k \in \mathcal{K}$ et d'un horaire $s \in$

Tableau 1.5: Les contraintes de quotas définies sur l'horizon.

	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
t_1	$(Q_{L_1}) : (\{\ell_1\},\ (1,0,0))$	$(Q_{L_1}) : (\{\ell_3, \ell_4\},\ (2,1,0))$	$(Q_{L_1}) : (\{\ell_1, \ell_3\},\ (1,1,0))$ $(Q_{L_2}) : (\{\ell_2, \ell_4\},\ (1,0,1))$ $(\hat{Q}_{d_3 t_1}) : (\ \{\ell_1, \ell_2, \ell_3, \ell_4\},$ $(2,0,0)\ )$

S_k (noté par a_{ksdt}) ne peut être comptabilisée que dans une seule contrainte de quotas de type A définie sur cette même paire. Considérons deux contraintes $(Q_{L'})$ et $(Q_{L''})$ définies sur la paire (d,t), où L' et $L'' \in \mathcal{L}_{dt}$ et $L' \neq L''$, et supposons que $L' \bigcap \Lambda_k \neq \emptyset$ et $L'' \bigcap \Lambda_k \neq \emptyset$. Nonobstant les multiples compétences de l'infirmière k qui permettent à l'affectation a_{ksdt} de couvrir ces deux contraintes, cette même affectation doit être comptabilisée soit dans $(Q_{L'})$ ou soit dans $(Q_{L''})$ et non pas simultanément dans les deux.

Nous pouvons voir au tableau 1.6 tous les horaires possibles pour les trois infirmières k_1, k_2 et k_3. Remarquons que les horaires s_1 et s_2 de l'infirmière k_1 sont différents uniquement par cause des affectations (d_3, t_1, L_1) et (d_3, t_1, L_2) qui leur sont respectivement associées. De façon plus précise, ces deux affectations auront respectivement un apport dans les contraintes (Q_{L_1}) et (Q_{L_2}), qui sont toutes les deux définies sur la paire (d_3, t_1). En effet, si NSP1 était le problème considéré, l'infirmière k_1 aurait seulement l'horaire composé des affectations (d_1, t_1) et (d_3, t_1) de valide.

1.5.2.3 Les solutions possibles

Dans cet exemple, nous avons 8 horaires généraux possibles. Par contre, seulement 3 d'entres eux respectent les contraintes de quotas : $(k_1 : s_1,\ k_2 : s_2,\ k_3 : s_2)$,

Tableau 1.6: Les horaires possibles pour les infirmières k_1, k_2 et k_3.

k	Λ_k	S_k	coût	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
k_1	$\{\ell_1,\ \ell_2\}$	s_1	0	(t_1, L_1)		(t_1, L_1)
		s_2	0	(t_1, L_1)		(t_1, L_2)
k_2	$\{\ell_3\}$	s_1	1		(t_1, L_1)	(t_1, L_1)
		s_2	0		(t_1, L_1)	
k_3	$\{\ell_4\}$	s_1	2		(t_1, L_1)	
		s_2	0			(t_1, L_2)

$(k_1 : s_2,\ k_2 : s_1,\ k_3 : s_1)$ et $(k_1 : s_2,\ k_2 : s_2,\ k_3 : s_2)$. Les valeurs de ces trois horaires généraux sont respectivement 1, 3 et 3. Par conséquent, la valeur optimale du problème est 1 et $(k_1 : s_1,\ k_2 : s_2,\ k_3 : s_2)$ est une solution optimale. Pour chacun des ces horaires, un tableau décrit la comptabilisation des contraintes de quotas.

Tableau 1.7: Comptabilisation des quotas pour l'horaire $(k_1 : s_1,\ k_2 : s_2,\ k_3 : s_2)$ de valeur 1.

	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
t_1	$(Q_{L_1}) : 1$	$(\mathbf{Q_{L_1}}) : \mathbf{1}$	$(Q_{L_1}) : 1$ $(Q_{L_2}) : 1$ $(\hat{Q}_{d_3 t_1}) : 2$

Tableau 1.8: Comptabilisation des quotas pour l'horaire $(k_1 : s_2,\ k_2 : s_1,\ k_3 : s_1)$ de valeur 3.

	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
t_1	$(Q_{L_1}) : 1$	$(Q_{L_1}) : 2$	$(Q_{L_1}) : 1$ $(Q_{L_2}) : 1$ $(\hat{Q}_{d_3 t_1}) : 2$

Tableau 1.9: Comptabilisation des quotas pour l'horaire ($k_1 : s_2$, $k_2 : s_2$, $k_3 : s_2$) de valeur 3.

	08/09/2003 (d_1)	09/09/2003 (d_2)	10/09/2003 (d_3)
t_1	$(Q_{L_1}) : 1$	$(\mathbf{Q_{L_1}}) : \mathbf{1}$	$(\mathbf{Q_{L_1}}) : \mathbf{0}$ $(\mathbf{Q_{L_2}}) : \mathbf{2}$ $(\hat{Q}_{d_3 t_1}) : 2$

1.5.3 Modélisation mathématique

Maintenant, nous allons montrer le modèle mathématique pour le problème NSP2. Ce modèle est entièrement décrit à la figure 1.3. Les différences avec le modèle mathématique du problème NSP1 sont au nombre de deux : (1) nouvelle variable pour les affectations et (2) modélisation des contraintes de quotas de type B.

Nouvelle variable pour les affectations

Pour chaque quadruplet (k, s, d, t) (où $k \in \mathcal{K}$, $s \in S_k$ et $(d, t) \in \mathcal{D}_{\mathcal{T}_k}$), nous devons remplacer la variable a_{ksdt} par plusieurs variables a_{ksdtL}, où $L \in \mathcal{L}_{dt}$. En effet, ceci est dû à une nouvelle tâche du problème auxiliaire : définir dans laquelle des contraintes (Q_L), où $L \in \mathcal{L}_{dt}$ et $|L \bigcap \Lambda_k| \geq 1$, l'affectation (d, t) de l'infirmière k et de l'horaire $s \in S_k$ aura un apport. De plus, comme pour le PLNE du NSP1, la variable a_{ksdtL} est propre au problème auxiliaire et est donc une constante dans le PLNE du NSP2.

Ainsi, considérons la paire $(d, t) \in \mathcal{D}_{\mathcal{T}}$, pour chaque quintuplet (k, s, d, t, L) (où $k \in \mathcal{K}$, $s \in S_k$ et $(d, t) \in \mathcal{D}_{\mathcal{T}_k}$) telle que $L \in \mathcal{L}_{dt}$ et $|L \bigcap \Lambda_k| \geq 1$, nous définissons notre nouvelle variable comme suit :

$$a_{ksdtL} = \begin{cases} 1 & \text{si l'infirmière } k \text{ est affectée à } (d,t) \text{ à l'horaire } s \text{ et si cette} \\ & \text{affectation est considérée dans la contrainte de quotas } (Q_L) \\ & \text{de la paire } (d,t) \\ 0 & \text{sinon.} \end{cases}$$

Suite à cette décomposition de variables, on peut noter que les constantes b_{kL} ne sont plus nécessaires dans notre nouveau modèle. De plus, comme l'était notre ancienne variable a_{ksdt}, la nouvelle variable a_{ksdtL} est aussi une variable du problème auxiliaire et elle nous permet de communiquer avec l'horaire $s \in S_k$, qui a précédemment été généré par le problème auxiliaire. De la même façon, elle est considérée comme une constante dans notre modèle.

Nous verrons au chapitre 4 les modifications à apporter au problème auxiliaire suite à cette décomposition de variables. Pour l'instant, le lecteur doit retenir que c'est la tâche du problème auxiliaire de définir dans laquelle des contraintes (Q_L) $(L \in \mathcal{L}_{dt})$ d'une paire $(d,t) \in \mathcal{D}_{\mathcal{T}_k}$, l'affectation (d,t) aura un apport.

Modélisation des contraintes de quotas de type B

Il ne nous reste qu'à ajouter les contraintes de quotas de type B dans notre modèle. Soit $\hat{Q} \subseteq \mathcal{D}_{\mathcal{T}}$, l'ensemble des paires $(d,t) \in \mathcal{D}_{\mathcal{T}}$ où une contrainte de quotas de type B $(\hat{Q}_{dt})$ a été définie. Alors, pour chaque contrainte $(\hat{Q}_{dt})$, où $(d,t) \in \hat{Q}$, nous avons trois nouvelles contraintes :

- $\sum_{k \in \mathcal{K}} \sum_{s \in S_k} \sum_{L \in \mathcal{L}_{dt}} a_{ksdtL}\ x_{ks} + \hat{s}^-_{dt} - \hat{s}^+_{dt} = \hat{q}_{dt}$,
- $\hat{s}^-_{dt} \in \{0, 1, ..., \hat{q}^-_{dt}\}$,
- $\hat{s}^+_{dt} \in \{0, 1, ..., \hat{q}^+_{dt}\}$.

Ici, nous avons introduit deux nouvelles variables : $\hat{s}^-_{dt}$ et $\hat{s}^+_{dt}$. Elles indiquent respectivement les écarts, de déficit et de surplus, comptabilisés dans l'horaire général pour la contrainte de quotas ($\hat{Q}_{dt}$). Enfin, il ne faut pas oublier d'introduire dans la fonction objectif la pénalité accordée à ces écarts : $\sum_{(d,t)\in\hat{Q}} (p^- \ \hat{s}^-_{dt} + p^+ \ \hat{s}^+_{dt})$.

$$\min \Big(\sum_{k\in\mathcal{K}} \sum_{s\in S_k} p_{ks} \ x_{ks} + \sum_{(d,t)\in\mathcal{D}_\mathcal{T}} \sum_{L\in\mathcal{L}_{dt}} (p^- \ s^-_L + p^+ \ s^+_L) + \sum_{(d,t)\in\hat{Q}} (p^- \ \hat{s}^-_{dt} + p^+ \ \hat{s}^+_{dt}) \Big)$$

s.l.c. :

$$\sum_{k\in\mathcal{K}} \sum_{s\in S_k} a_{ksdtL} \ x_{ks} + s^-_L - s^+_L = q_L \qquad (d,t) \in \mathcal{D}_\mathcal{T}, \ L \in \mathcal{L}_{dt}$$

$$\sum_{k\in\mathcal{K}} \sum_{s\in S_k} \sum_{L\in\mathcal{L}_{dt}} a_{ksdtL} \ x_{ks} + \hat{s}^-_{dt} - \hat{s}^+_{dt} = \hat{q}_{dt} \qquad (d,t) \in \hat{Q}$$

$$\sum_{s\in S_k} x_{ks} = 1 \qquad k \in \mathcal{K}$$

$$x_{ks} \in \{0,1\} \qquad k \in \mathcal{K}, \ s \in S_k$$

$$s^-_L \in \{0, \ 1, \ \ldots, \ q^-_L\} \qquad (d,t) \in \mathcal{D}_\mathcal{T}, \ L \in \mathcal{L}_{dt}$$
$$s^+_L \in \{0, \ 1, \ \ldots, \ q^+_L\} \qquad (d,t) \in \mathcal{D}_\mathcal{T}, \ L \in \mathcal{L}_{dt}$$

$$\hat{s}^-_{dt} \in \{0, \ 1, \ \ldots, \ \hat{q}^-_{dt}\} \qquad (d,t) \in \hat{Q}$$
$$\hat{s}^+_{dt} \in \{0, \ 1, \ \ldots, \ \hat{q}^+_{dt}\} \qquad (d,t) \in \hat{Q}$$

Figure 1.3: Formulation du problème NSP2 en un PLNE.

CHAPITRE 2 : L'ALGORITHME DE BRANCH AND PRICE

Le problème à résoudre comporte plusieurs difficultés dont trois sont primordiales. Premièrement, il existe un nombre exponentiel d'horaires individuels (colonnes) pour chaque infirmière [1]. Ensuite, un horaire général est réalisable si et seulement si un seul horaire (individuel) est affecté à chaque infirmière et si l'ensemble de ces horaires respecte les contraintes de quotas [2]. Troisièmement, parmi tous les horaires généraux réalisables, il faut trouver celui qui est de valeur optimale. Pour résoudre ces problèmes, on utilise l'algorithme de séparation et d'évaluation progressive (branch and price).

Dans ce chapitre, nous allons voir en détail toutes les composantes de notre algorithme ainsi que leurs adaptations pour notre application. Mais avant d'entrer dans ces détails, commençons par donner une vue d'ensemble de cette méthode.

2.1 Introduction au branch and price

Cet algorithme consiste à parcourir l'ensemble de solutions (entières) du PLNE (P) de façon intelligente et non explicite. Ceci se fait en partitionnant cet ensemble, qui est noté par S, en plusieurs sous-ensembles à travers un arbre de recherche. Dans cet arbre, chaque noeud u correspond à un PLNE (P^u), où l'ensemble des solutions réalisables de ce même problème est noté par $S^u \subseteq S$. Si nécessaire, une méthode

[1] Les termes "horaire individuel" et "colonne" seront considérés comme des synonymes.
[2] Par défaut, le terme "horaire" désigne un horaire individuel.

de séparation sera appliquée au problème (P^u) afin de créer deux problèmes fils notés (P^v) et (P^w). Cette méthode de séparation consiste à ajouter une contrainte de branchement différente à chacun de ces PLNE afin de partionner l'ensemble de solutions S^u du problème (P^u) en deux sous-ensembles disjoints : S^v et S^w.

À chaque noeud u de l'arbre, la relaxation continue (R^u) du PLNE (P^u) est résolue afin d'obtenir une borne inférieure sur la valeur de toutes les solutions de ce problème en nombres entiers. Le problème (R^u) est identique au problème (P^u), mais contrairement à ce dernier qui n'accepte que des solutions avec des variables de valeur entière, le problème (R^u) permet à ces mêmes variables des valeurs réels. Par conséquent, le problème (P^u) étant plus contraint que sa relaxation (R^u), la valeur optimale z_u^* de cette dernière est une borne inférieure sur la valeur de toutes les solutions de l'ensemble S^u. D'ailleurs, lorsque la solution x_u^n, qui est obtenue après la résolution du problème (R^u) et qui est de valeur optimale z_u^*, est entière, le problème (P^u) est résolu [3]. Dans ce cas, il est possible de mettre à jour la valeur de la meilleure solution du problème (P) actuellement connue, qu'on note par z^*. Cette valeur est aussi une borne supérieure sur la valeur optimale de ce même problème.

Voici les principales composantes de l'algorithme du branch and price :

- méthode de génération de colonnes,
- méthode de pricing (résolution du problème auxiliaire),
- méthodes de séparation,
- algorithmes d'exploration de l'arbre de recherche,
- conditions pour couper un noeud,
- stratégie de branchement prématuré,
- méthode d'arrondi.

[3] Nous verrons plus loin la signification de la notation x_u^n.

Premièrement, la méthode de génération de colonnes est utilisée pour résoudre le problème (R^u) au noeud u. Cette méthode marie l'algorithme du Simplexe avec une méthode nommée Pricing qui a la tâche de générer de nouvelles colonnes de coût réduit négatif [4]. Ensuite, lorsque la solution ainsi obtenue est fractionnaire, une méthode de séparation est appliquée au problème (P^u) et consiste à partitionner son ensemble de solutions (S^u) en deux sous-ensembles disjoints. Ceci a pour conséquence d'éliminer la réalisabilité de la solution fractionnaire x_u^n pour les deux nouveaux problèmes (P^v) et (P^w), qui sont les fils de (P^u), et possiblement d'atteindre une solution entière dans ces problèmes ou plus profondément dans l'arbre de recherche. Troisièmement, deux algorithmes sont possibles pour l'exploration de l'arbre de recherche : 1) *recherche en profondeur d'abord* et 2) *recherche avec le meilleur des deux fils d'abord* [5]. Evidemment, il y a des avantages et des inconvénients à utiliser l'un ou l'autre de ces deux algorithmes. Enfin, après avoir résolu le problème (R^u), si sa valeur optimale z_u^* est telle que $\lceil z_u^* \rceil \geq z^*$, il est alors inutile de résoudre le problème (P^u) en partitionnant son ensemble de solution S^u. Cette condition pour couper un noeud u sera développée au cours de ce chapitre.

Plusieurs stratégies peuvent être ajoutées à l'algorithme du branch and price pour diminuer son temps de traitement. La première est la stratégie du branchement prématuré. Elle consiste à arrêter prématurément l'optimisation du problème (R^u) au noeud u lorsque nous considérons qu'il est inutile et (ou) trop long de connaître sa valeur optimale z_u^*. La deuxième stratégie est la méthode d'arrondi. Elle consiste à trouver une solution entière, donc un horaire général et réalisable, à l'aide des variables déjà existantes. En plus de celles-ci, d'autres stratégies sont présentées par Desaulniers [19].

Nous pouvons voir à la figure 2.1 un exemple simple d'arbre de recherche. On peut supposer que les problèmes (P^i) ont été explorés par ordre croissant de leur indice i et

[4] Appelées colonnes CR^-.

[5] Respectivement abrégé par DFS (Depth First Search) et BSFS (Best Son First Search).

que chaque z_i^*, qui est la valeur optimale de la relaxation continue (R^i), a été atteinte. Après la résolution du problème (R^2), on obtient la première solution entière x_2^n de valeur z_2^*. Ensuite, nous obtenons au problème (R^3) la solution fractionnaire x_3^n de valeur z_3^*, où $\lceil z_3^* \rceil \geq z_2^*$. Dans ce cas, remarquons que $\lceil z_3^* \rceil$ est une borne inférieure sur la valeur de toutes solutions du problème (P^3) et que z_2^* est une borne supérieure sur la valeur optimale du problème (P). Les mêmes remarques s'appliquent au problème (P^4), où $\lceil z_4^* \rceil \geq z_2^*$. C'est pour ces raisons que les deux problèmes (P^3) et (P^4) ont été coupés.

Finalement, dans ce chapitre nous allons présenter en détail chaque composante de l'algorithme du branch and price. Nous commençons par décrire la relaxation continue (R) du problème (P). Deuxièmement, nous allons expliquer la méthode de génération de colonnes, qui est composée de deux phases et de la méthode Pricing. Ensuite, deux méthodes de séparation possibles seront présentées : règle de branchement binaire simple et règle de branchement Ryan-Foster [6]. Quatrièmement, les deux parcours possibles de l'arbre de recherche seront expliqués : le DFS et le BSFS. Finalement, nous expliquerons pourquoi notre algorithme de branch and price est exhaustif, c'est à dire qu'il calculera la valeur optimale z^* du problème (P).

2.2 Présentation du problème (R)

Comme nous l'avons mentionné à la section précédente, à chaque problème (P^u) de l'arbre de recherche correspond une relaxation continue notée (R^u) [7]. Dans cette relaxation, les contraintes d'intégralité sur les valeurs des variables x_{ks}, s_L^- et s_L^+ ont été assouplies afin de leur permettre des valeurs réelles. De façon plus précise, voici

[6] Afin d'alléger le texte, le branchement binaire simple sera nommé "branchement binaire".

[7] Le problème (P^u) est défini à la figure 2.7.

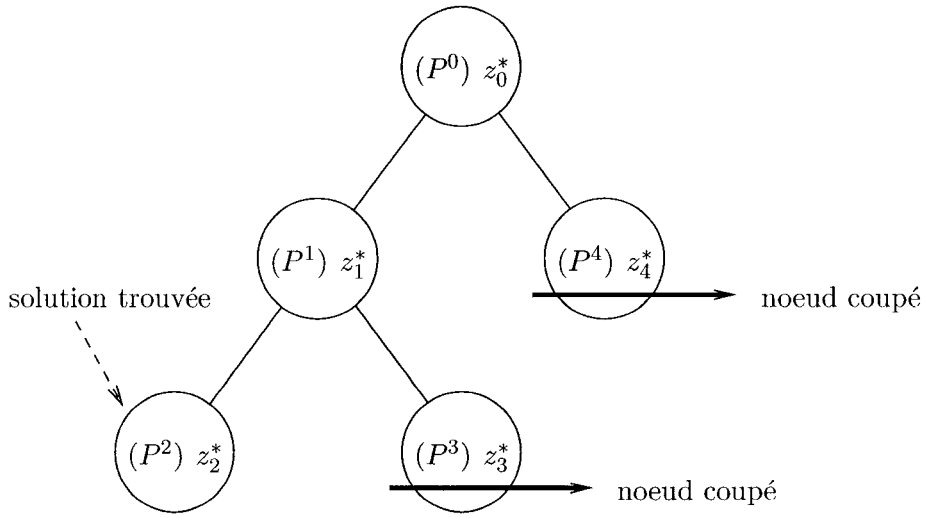


Figure 2.1: Exemple d'arbre de recherche.

les modifications qui sont apportées dans la formulation du problème (P) afin de définir la relaxation continue (R) [8] :

- $x_{ks} \in \{0,1\}$ est remplacé par $0 \leq x_{ks} \leq 1$,
- $s_L^- \in \{0,1,...,q_L^-\}$ est remplacé par $0 \leq s_L^- \leq q_L^-$,
- $s_L^+ \in \{0,1,...,q_L^+\}$ est remplacé par $0 \leq s_L^+ \leq q_L^+$.

Avec ces modifications, nous pouvons voir à la figure 2.2 la relaxation continue (R). Ces modifications sont les mêmes à apporter au PLNE (P^u) afin de définir sa relaxation continue (R^u). Aussi, nous verrons plus loin dans ce chapitre que ces modifications affecteront aussi la définition de la relaxation continue (R^{ua}), qui est utilisée par la phase 1 de la méthode de génération de colonnes lorsqu'elle est appliquée au problème (R^u).

De plus, la variable duale associée à une contrainte de quotas (Q_L), où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_{\mathcal{P}}$, et celle associée à la contrainte de partionnement d'une infirmière $k \in \mathcal{K}$

[8] Voir la figure 1.2 pour la définition du problème (P).

$$\min \Big(\sum_{k\in\mathcal{K}} \sum_{s\in S_k} p_{ks}\, x_{ks} + \sum_{(d,p)\in\mathcal{D}_\mathcal{P}} \sum_{L\in\mathcal{L}_{dp}} (p^-\, s_L^- + p^+\, s_L^+) \Big)$$

s.l.c. :

$$\sum_{k\in\mathcal{K}} \sum_{s\in S_k} \sum_{t\in\mathcal{T}_k} \alpha_{tp}\, b_{kL}\, a_{ksdt}\, x_{ks} + s_L^- - s_L^+ = q_L \quad (d,p)\in\mathcal{D}_\mathcal{P},\ L\in\mathcal{L}_{dp}$$

$$\sum_{s\in S_k} x_{ks} = 1 \qquad k\in\mathcal{K}$$

$$0 \le x_{ks} \le 1 \qquad k\in\mathcal{K},\ s\in S_k$$

$$0 \le s_L^- \le q_L^- \qquad (d,p)\in\mathcal{D}_\mathcal{P},\ L\in\mathcal{L}_{dp}$$

$$0 \le s_L^+ \le q_L^+ \qquad (d,p)\in\mathcal{D}_\mathcal{P},\ L\in\mathcal{L}_{dp}$$

Figure 2.2: Relaxation continue (R) du problème (P).

sont respectivement notées λ_L et μ_k.

2.3 Méthode de génération de colonnes en 2 phases

Dans l'algorithme de branch and price, la méthode de génération de colonnes est utilisée pour résoudre la relaxation continue (R^u) du problème (P^u) au noeud u de l'arbre de recherche. Cependant, avant d'appliquer cette technique, il faut vérifier que (R^u) est réalisable. C'est pour cette raison que l'algorithme de génération de colonnes est divisé en 2 phases : GC_{ϕ_1} et GC_{ϕ_2}. GC_{ϕ_1} aura pour tâche de vérifier si la relaxation continue (R^u) est réalisable alors que GC_{ϕ_2} devra lui trouver une base primale-duale réalisable [9].

Il est important de différencier les phases 1 et 2 propres à l'algorithme du Simplexe

[9] Primale-duale réalisable est synonyme de optimale.

(Chvátal [20]) et les phases GC_{ϕ_1} et GC_{ϕ_2} de la méthode de génération de colonnes. Les phases 1 et 2 du Simplexe sont implicitement traitées lors des différents appels à celui-ci, mais en plus elles ont les mêmes raisons d'existence que GC_{ϕ_1} et GC_{ϕ_2}. Notons que la méthode du Simplexe utilisée est celle de la librairie ILOG [21].

Comme nous allons le voir, GC_{ϕ_1} et GC_{ϕ_2} ont chacun un problème différent à résoudre. Soit la relaxation continue modifiée (R^{ua}) pour GC_{ϕ_1} et la relaxation continue (R^u) pour GC_{ϕ_2} [10]. Par contre, l'algorithme utilisé pour résoudre ces deux problèmes est commun aux deux phases et il est basé sur la méthode revisée du Simplexe (Chvátal [20], Nemhauser et Wolsey [22]). Dans la communauté scientifique, cette technique porte la dénomination "méthode de génération de colonnes". Mais afin d'alléger le présent mémoire, nous la notons par GC.

Maintenant, exposons le plan de cette section. Nous commençons par décrire de façon explicite les algorithmes GC et Pricing. Ensuite, nous expliquons GC_{ϕ_1} et nous développons le problème modifié (R^{ua}). Troisièmement, une condition nécessaire et suffisante sur la réalisabilité du problème (R^u) est donnée. Quatrièmement, GC_{ϕ_2}, qui consiste essentiellement à résoudre le problème (R^u), est exposé. Pour conclure sur cette méthode de génération de colonnes en deux phases, abrégé par GC_METHOD, nous donnons son organigramme et son algorithme pour aider à sa compréhension.

2.3.1 L'algorithme GC

Avant de commencer la description de cet algorithme, il est important de rappeler au lecteur que cette méthode est utilisée pour résoudre l'une ou l'autre des deux relaxations continues suivantes : (R^u) ou (R^{ua}). Ainsi, pour éviter toute confusion,

[10] Voir la figure 2.3 pour la définition du problème (R^{ua}).

nous allons noter par $(\dot{R}^u)$ le problème considéré par cet algorithme. De plus, la plupart des notations utilisées dans cette section sont décrites dans la "Liste des sigles et abréviations".

Encore une fois, nous devons introduire de nouvelles notations. Soit le PLNE (P^u) du noeud u, une infirmière $k \in \mathcal{K}$, le problème restreint $(\dot{R}^{ui})$ de l'itération i de GC et une RFRC $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ [11]. Notons par

- $S_k^u \subseteq S_k$, l'ensemble des horaires de l'infirmière k qui sont réalisables au PLNE (P^u) ;
- $\dot{S}_k^{ui} \subseteq S_k^u$, l'ensemble des horaires de l'infirmière k qui sont inclus dans le problème restreint $(\dot{R}^{ui})$;
- $S_k^{cu} \subseteq S_k^u$, l'ensemble des horaires de l'infirmière k qui respecte la RFRC (k, c) du PLNE (P^u).

Soit un horaire $s \in S_k^u$. Cet horaire est tel que $s \in \dot{S}_k^{ui}$ si et seulement s'il a été généré par le problème auxiliaire à une itération antérieure à l'itération i ou à un PLNE antérieur au PLNE (P^u).

Description de l'algorithme :

Ne connaissant pas toutes les variables du problème $(\dot{R}^u)$, qui sont en quantité exponentielle, une méthode nommée Pricing (voir la section 2.3.2) est utilisée pour ajouter au problème des colonnes CR^- qui sont hors base.

Essentiellement, la méthode GC utilise un algorithme itératif. À chaque itération i, le problème restreint $(\dot{R}^{ui})$ est résolu par la méthode du Simplexe. Le résultat obtenu est une base, qui est primale-duale réalisable à $(\dot{R}^{ui})$, notée $\dot{x}_u^i$ et de valeur

[11] Le concept RFRC sera introduit à la section 2.3.2.1.

optimale $\dot{z}_u^i$. Ensuite, la méthode Pricing est appelée pour générer des colonnes CR^- hors bases, soit des variables de coûts réduits négatifs qui ne sont pas déjà dans le problème restreint $(\dot{R}^{ui})$. L'ensemble de colonnes ainsi obtenu est noté $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \subseteq \{s \in S_k^{cu} : s \notin \dot{S}_k^{ui} \text{ et } c_{ks} < 0\}$, où $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ [12].

Ensuite, si $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \neq \emptyset$, $\dot{x}_u^i$ n'est donc pas primale-duale réalisable pour $(\dot{R}^u)$ et l'algorithme continue. L'ensemble de colonnes CR^- $PA_{\phi_2}(\hat{G}_{ki}^{cu})$ est ajouté au problème $(\dot{R}^{ui})$ afin de définir le nouveau problème restreint : $(\dot{R}^{u(i+1)}) \leftarrow (\dot{R}^{ui}) \oplus PA_{\phi_2}(\hat{G}_{ki}^{cu})$. En d'autres termes : $\dot{S}_k^{u(i+1)} \leftarrow \dot{S}_k^{ui} \oplus PA_{\phi_2}(\hat{G}_{ki}^{cu})$. Une nouvelle itération est alors effectuée pour le traitement de ce nouveau problème restreint.

Enfin, si $PA_{\phi_2}(\hat{G}_{ki}^{cu}) = \emptyset$, cela signifie que la condition d'optimalité $\big(\forall (k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$, $PA_{\phi_2}(\hat{G}_{ki}^{cu}) = \emptyset\big)$ est vérifiée. Autrement dit, il n'existe pas de colonnes CR^- hors bases, alors $\dot{x}_u^i$ est une base primale-duale réalisable pour le problème restreint $(\dot{R}^{ui})$ mais aussi pour le problème $(\dot{R}^u)$. Donc, $\dot{z}_u^i = \dot{z}_u^*$ est la valeur optimale de ce dernier problème et GC est terminé.

Finalement, les ensembles de variables $PA_{\phi_2}(\hat{G}_{ki}^{cu})$ et la notation $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ seront définis à la section 2.3.2, qui est réservée à la méthode Pricing. Au même moment, le fondement de la précédente condition, qui est nécessaire et suffisante à l'optimalité d'une solution, sera expliqué.

Observations importantes

Ainsi, en considérant les itérations 1 à n de GC, la série de n triplets suivants est générée : [$((\dot{R}^{u1}), \dot{x}_u^1, \dot{z}_u^1)$, $((\dot{R}^{u2}), \dot{x}_u^2, \dot{z}_u^2)$, ..., $((\dot{R}^{un}), \dot{x}_u^n, \dot{z}_u^n)$]. Plusieurs observations sont importantes :

[12] Cet ensemble et cette notation seront définis à la section 2.3.2.2.

1. chacune des solutions $(\dot{x}_u^i, \dot{z}_u^i)$ est primale-duale réalisable au problème restreint $(\dot{R}^{ui})$;
2. pour chacune des itérations i, la valeur de la fonction objectif est au moins aussi bonne que celle de l'itération précédente : $\dot{z}_u^i \leq \dot{z}_u^{i-1}$. Et s'il n'y a pas de dégénérescence, nous avons $\dot{z}_u^i < \dot{z}_u^{i-1}$;
3. considérons une paire d'itérations successives $(i-1, i)$, où $\dot{z}_u^i < \dot{z}_u^{i-1}$. Alors, la solution primale-duale $(\dot{x}_u^{i-1}, \dot{z}_u^{i-1})$ du problème $(\dot{R}^{u(i-1)})$ n'est plus duale réalisable au problème suivant $(\dot{R}^{ui})$, mais elle demeure primale réalisable pour ce même problème.

La troisième observation s'explique grâce à la relation entre un problème primal et son dual. L'ajout de nouvelles colonnes au problème primal signifie l'ajout de nouvelles lignes à son dual. Par conséquent, une solution qui est réalisable à ces deux problèmes perd sa réalisabilité duale après l'ajout de nouvelles lignes (contraintes) au problème dual. C'est par cette explication que la solution $(\dot{x}_u^{i-1}, \dot{z}_u^{i-1})$, qui est primale-duale réalisable au problème $(\dot{R}^{u(i-1)})$, n'est plus duale réalisable mais demeure primale réalisable au problème suivant $(\dot{R}^{ui})$.

Description formelle de l'algorithme

La méthode GC est explicitement donnée par l'algorithme 2.1. Elle retourne le dernier triplet qu'elle a créé, soit celui de l'itération n qui est composé du problème restreint $(\dot{R}^{un})$, de la base $\dot{x}_u^n$ et de sa valeur $\dot{z}_u^n$. Elle retourne aussi la valeur optimale $\dot{z}_u^*$, qui débute l'algorithme avec une valeur de $+\infty$. Toutefois, si un branchement prématuré est décidé, cette valeur n'est pas atteinte à l'itération n et le test $\dot{z}_u^n \stackrel{?}{=} \dot{z}_u^*$ est donc faux. Il est important de souligner que ce test sera fréquemment utilisé dans plusieurs composantes de l'algorithme de branch and price pour déterminer si la valeur optimale $\dot{z}_u^*$ du problème $(\dot{R}^u)$ a été atteinte par GC.

Aussi, il est important de souligner qu'une condition nécessaire à l'application de GC pour la résolution de la relaxation continue $(\dot{R}^u)$ doit être satisfaite. En effet, il est indispensable que ce problème soit réalisable avant de faire appel à cette méthode. Si cette condition est satisfaite, il faut que le premier problème restreint $(\dot{R}^{u1})$, qui est fourni à GC, soit aussi réalisable. Plus de détails concernant la technique utilisée pour satisfaire ces deux conditions nécessaires à la résolution des problèmes (R^{ua}) et (R^u) seront respectivement décrites aux sections 2.3.3 et 2.3.5.

Dans la prochaine section, nous décrivons la méthode Pricing, qui génère l'ensemble de colonnes CR^- hors bases $PA_{\phi_2}(\hat{G}^{cu}_{ki})$ d'une combinaison quelconque $(k,c) \in \mathcal{K}_{\mathcal{I}^u_k}$ et qui vérifie la condition d'optimalité $\big(\forall (k,c) \in \mathcal{K}_{\mathcal{I}^u_k},\ PA_{\phi_2}(\hat{G}^{cu}_{ki}) = \emptyset\big)$.

Algorithme 2.1 Méthode de génération de colonnes pour résoudre le problème linéaire $(\dot{R}^u)$, qui correspond à (R^u) ou à (R^{ua}) du problème (P^u) au noeud u.

GC**(** $(\dot{\mathbf{R}}^{\mathbf{u1}})$ **: problème initial)** $\Longrightarrow$ ($(\dot{\mathbf{R}}^{\mathbf{un}})$, $\dot{\mathbf{x}}^{\mathbf{n}}_{\mathbf{u}}$, $\dot{\mathbf{z}}^{\mathbf{n}}_{\mathbf{u}}$, $\dot{\mathbf{z}}^{*}_{\mathbf{u}}$) :
$i \leftarrow 0$
$\dot{z}^*_u \leftarrow +\infty$
tant que ($i = 0 \bigvee PA_{\phi_2}(\hat{G}^{cu}_{ki}) \neq \emptyset$) **effectuer**
  $i \leftarrow i + 1$
  ($\dot{x}^i_u$, $\dot{z}^i_u$, $[\lambda_L]^i$, $[\mu_k]^i$) $\leftarrow$ Simplexe $(\dot{R}^{ui})$
  $PA_{\phi_2}(\hat{G}^{cu}_{ki})$ $\leftarrow$ Pricing($\mathcal{K}_{\mathcal{I}^u_k}$, $[\lambda_L]^i$, $[\mu_k]^i$) {Gen. de var. CR^-}
  si $PA_{\phi_2}(\hat{G}^{cu}_{ki}) \neq \emptyset$ **alors**
    si branchementPremature($\dot{z}^i_u$) **alors**
      $n \leftarrow i$
      Aller à (*)
    $(\dot{R}^{u(i+1)}) \leftarrow (\dot{R}^{ui}) \oplus PA_{\phi_2}(\hat{G}^{cu}_{ki})$
$n \leftarrow i$
$\dot{z}^*_u \leftarrow \dot{z}^i_u$ { La valeur optimale $\dot{z}^*_u$ a été atteinte : $\dot{z}^n_u \stackrel{?}{=} \dot{z}^*_u$ est vraie }
(*) **retourner** $\big((\dot{R}^{un}), \dot{x}^n_u, \dot{z}^n_u, \dot{z}^*_u\big)$

2.3.2 L'algorithme Pricing

Avant de commencer la description de cette méthode, il est important de préciser au lecteur que beaucoup de notations sont nécessaires à sa compréhension. La plupart de celles-ci seront expliquées dans les prochains chapitres. Pour l'instant, le lecteur est référé à la "Liste des sigles et abréviations".

La méthode Pricing a pour tâche de générer des colonnes CR^{-} qui sont hors bases lors de l'itération courante i de GC. Les variables duales de chaque contrainte de quotas et de chaque contrainte de partitionnement de cette itération doivent être fournies. Elles sont respectivement notées par les vecteurs $[\lambda_L]^i$ et $[\mu_k]^i$ et sont obtenues par la méthode du Simplexe.

L'algorithme parcourt l'ensemble $\mathcal{K}$ des infirmières afin de trouver au moins une colonne CR^{-} hors base. Par contre, lorsqu'une infirmière $k \in \mathcal{K}$ est traitée, il faut considérer les différentes façons de respecter simultanément toutes les contraintes de branchement Ryan-Foster et de branchement binaire. Une telle façon ou RFRC est notée $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ [13]. Alors, pour chacune de ces combinaisons, la phase 2 du problème auxiliaire est appelée dans le but de trouver une ou plusieurs colonnes hors base qui respectent la définition de cette combinaison et qui sont de coûts réduits négatifs selon les variables duales $[\lambda_L]^i$ et $[\mu_k]^i$.

Précisons que les règles de branchement Ryan-Foster et binaire seront définies à la section 2.4.2, pour ensuite être approfondies au chapitre 3. De plus, les phases 1 et 2 du problème auxiliaire seront définies au chapitre 4.

[13] RFRC - Ryan-Foster Respect Combination.

Afin de bien comprendre l'algorithme Pricing, nous commençons par introduire le lecteur au concept de RFRC. À l'aide d'exemples et du développement d'un algorithme nommé *calculer* $\mathcal{I}_k^u$, ce concept sera entièrement étudié à la section 3.3. Ensuite, la phase 2 du problème auxiliaire est aussi introduite. Nous terminons en décrivant les deux résultats pouvant être obtenus par l'application de Pricing.

2.3.2.1 Introduction au concept de RFRC

Comme nous allons le voir aux définitions 2.4 et 2.5, il y a deux possibilités pour le respect d'une contrainte de branchement Ryan-Foster alors qu'il y en a une seule pour une contrainte de branchement binaire. Par conséquent, le nombre total de combinaisons des deux possibilités pour le respect de chaque contrainte de branchement Ryan-Foster est de $2^{|\mathcal{C}_k^u|}$, où $k \in \mathcal{K}$, u est le noeud courant et $\mathcal{C}_k^u \leftarrow \mathcal{C}_{k=}^u \bigcup \mathcal{C}_{k\neq}^u$. Les deux ensembles de contraintes de branchement Ryan-Foster $\mathcal{C}_{k=}^u$ et $\mathcal{C}_{k\neq}^u$ seront définis à la section 2.4.2 et pour le moment le lecteur doit bien noter que $|\mathcal{C}_k^u|$ représente le nombre total de ces contraintes.

Par contre, ce ne sont pas toutes les combinaisons mentionnées ci-haut qui sont possibles. En effet, certaines d'entres elles peuvent contenir une ou plusieurs contradictions. Pour l'instant, notons par $\mathcal{I}_k^u \subseteq \{0,\ 1,\ \ldots,\ 2^{|\mathcal{C}_k^u|}-1\}$ l'ensemble des indices des combinaisons qui sont sans contradiction pour $k \in \mathcal{K}$ et pour le problème (P^u). De plus, on note par $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$, où $\mathcal{K}_{\mathcal{I}_k^u} = \{(k,c) : k \in \mathcal{K},\ c \in \mathcal{I}_k^u\}$, la combinaison $c \in \mathcal{I}_k^u$ de l'infirmière $k \in \mathcal{K}$ au problème (P^u).

À la section 3.3, nous définirons l'algorithme *calculer* $\mathcal{I}_k^u$ dont les tâches sont les suivantes : (1) calculer l'ensemble $\mathcal{I}_k^u$ et (2) pour chaque $c \in \mathcal{I}_k^u$, calculer les ensembles $\mathcal{O}_k^{cu}$ et $\mathcal{F}_k^{cu}$. Ces deux ensembles représentent respectivement l'ensemble

des affectations obligatoires et l'ensemble des affectations interdites définissant la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$. Nous verrons à la section 2.3.6 que l'application de *calculer* $\mathcal{I}_k^u$ pour chaque infirmière $k \in \mathcal{K}$ et pour le problème (P^u) est un pré-traitement effectué au début de GC_METHOD.

2.3.2.2 Introduction au problème auxiliaire

Premièrement, le problème auxiliaire est divisé en deux phases : PA_{ϕ_1} et PA_{ϕ_2}. PA_{ϕ_1}, qui n'est qu'un pré-traitement à PA_{ϕ_2}, est accompli avant de débuter l'algorithme de branch and price et est appliqué sur un graphe noté G_k afin d'obtenir le graphe d'état $\hat{G}_k$. PA_{ϕ_2} a pour objectif de générer des horaires CR^- qui respectent la définition d'une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ au noeud u, en plus des contraintes intrinsèques à l'infirmière $k \in \mathcal{K}$. Cette seconde phase est un algorithme de plus court chemin appliqué au graphe d'état $\hat{G}_{ki}^{cu}$ et cette application est notée $PA_{\phi_2}(\hat{G}_{ki}^{cu})$. L'ensemble de colonnes CR^- hors base, qui est obtenu par cette application, est aussi noté $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \subseteq S_k$.

Ensuite, le graphe d'état $\hat{G}_{ki}^{cu}$, qui représente la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ et qui tient compte des variables duales $[\lambda_L]^i$ et $[\mu_k]^i$ pour le calcul des coûts réduits c_{ks} des horaires $s \in S_k^{cu}$, est défini à l'aide du graphe d'état $\hat{G}_k$. En effet, deux pré-traitements doivent être appliqués à $\hat{G}_k$ pour définir $\hat{G}_{ki}^{cu}$: 1) modification de $\hat{G}_k$ selon les contraintes définies par $\mathcal{O}_k^{cu}$ et $\mathcal{F}_k^{cu}$ et 2) modification de $\hat{G}_k$ pour la prise en compte des variables duales $[\lambda_L]^i$ et $[\mu_k]^i$.

Enfin, il est théoriquement impossible pour les colonnes obtenues par $PA_{\phi_2}(\hat{G}_{ki}^{cu})$ d'être déjà incluses dans le problème restreint $(\dot{R}^{ui})$. Autrement dit, cet ensemble est tel que $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \subseteq \{s \in S_k^{cu} : s \notin \dot{S}_k^{ui} \text{ et } c_{ks} < 0\}$. Aussi, mentionnons que $PA_{\phi_2}(\hat{G}_{ki}^{cu}) = \emptyset$ si et seulement si $\{s \in S_k^{cu} : s \notin \dot{S}_k^{ui} \text{ et } c_{ks} < 0\} = \emptyset$.

Plus de détails concernant les deux observations précédentes seront donnés à la section 4.1.3.2, qui est entièrement réservée à l'algorithme PA_{ϕ_2}. D'ailleurs, tout ce qui vient d'être expliqué sur le problème auxiliaire et ses différentes composantes sera expliqué de façon approfondie au chapitre 4.

2.3.2.3 Résultats possibles de Pricing

Après l'application de Pricing au triplet $\left(\mathcal{K}_{\mathcal{I}_k^u}, [\lambda_L]^i, [\mu_k]^i\right)$, les 2 résultats suivants sont possibles :

- soit que $\exists (k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ tel que $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \neq \emptyset$
- ou soit que $\forall (k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$, $PA_{\phi_2}(\hat{G}_{ki}^{cu}) = \emptyset$.

Le premier résultat signifie que la solution courante $\dot{x}_u^i$ de l'algorithme GC n'est pas une solution optimale du problème $(\dot{R}^u)$. En effet, il a été prouvé que pour une combinaison $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$, il existe au moins une colonne $s \in S_k^{cu}$ hors base ($s \notin \dot{S}_k^{ui}$) et de coût réduit négatif ($c_{ks} < 0$), donc $\dot{x}_u^i$ n'est pas primale-duale réalisable au problème $(\dot{R}^u)$.

Le deuxième résultat signifie qu'aucune colonne CR^- hors base n'existe pour chaque combinaison $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ et selon la valeur des variables duales $[\lambda_L]^i$ et $[\mu_k]^i$. Dans ce cas, la valeur courante $\dot{z}_u^i$ ($\dot{z}_u^n$) de l'algorithme GC est la valeur optimale du problème $(\dot{R}^u)$ ($\dot{z}_u^n \stackrel{?}{=} \dot{z}_u^*$ est vraie).

Finalement, nous pouvons voir la méthode Pricing décrite de façon formelle à l'algorithme 2.2.

Algorithme 2.2 L'algorithme Pricing où i est l'itération courante de GC.

Pricing ($\mathcal{K}_{\mathcal{I}_k^u}$: ens. de RFRC , $[\lambda_L]^i$, $[\mu_k]^i$: var. duales)
$\Longrightarrow$ $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \subseteq S_k^{cu}$

pour tout $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ **effectuer**
- $\hat{G}_{ki}^{cu} \leftarrow \hat{G}_k$
- Modififier $\hat{G}_{ki}^{cu}$ selon les ensembles $\mathcal{O}_k^{cu}$ et $\mathcal{F}_k^{cu}$ {Pré-traitement 1}
- Modififier $\hat{G}_{ki}^{cu}$ selon les variables duales $[\lambda_L]^i$ et $[\mu_k]^i$ {Pré-traitement 2}
- Appliquer $PA_{\phi_2}(\hat{G}_{ki}^{cu})$

si $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \neq \emptyset$ **alors**
- **retourner** $PA_{\phi_2}(\hat{G}_{ki}^{cu})$ $\{\exists (k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ tq. $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \neq \emptyset\}$

- **retourner** $\emptyset$ $\{\forall (k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$, $PA_{\phi_2}(\hat{G}_{ki}^{cu}) = \emptyset\}$

2.3.3 Description de la première phase GC_{ϕ_1}

L'objectif de cette phase est de déterminer si la relaxation continue (R^u) est réalisable. Pour atteindre cet objectif, GC_{ϕ_1} procède en 2 étapes : (1) définir une nouvelle relaxation continue inspirée de (R^u) et notée (R^{ua}) et (2) résoudre ce nouveau problème par la méthode GC. Cette méthode a déjà été donnée par l'algorithme 2.1, nous allons maintenant définir la relaxation continue modifiée (R^{ua}).

Le nouveau problème (R^{ua}) est obtenu par l'ajout de variables dites artificielles au problème (R^u). Les contraintes de partitionnement, les contraintes de quotas et la fonction objectif sont toutes modifiées par l'insertion de ces nouvelles variables. Ces modifications sont décrites ci-dessous.

- Pour chaque infirmière $k \in \mathcal{K}$, on crée la variable artificielle a_k telle que $0 \leq a_k \leq 1$.
- Pour chaque contrainte de quotas (Q_L), où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_{\mathcal{P}}$, on crée la variable artificielle a_L telle que $0 \leq a_L \leq q_L$.

- Les nouvelles contraintes de partitionnement sont définies comme suit :

  $\sum_{s \in S_k^u} x_{ks} + a_k = 1,\ k \in \mathcal{K}.$
- Les nouvelles contraintes de quotas sont définies comme suit :

  $\sum_{k \in \mathcal{K}} \sum_{s \in S_k^u} \sum_{t \in \mathcal{T}_k} \alpha_{tp}\ b_{kL}\ a_{ksdt}\ x_{ks} + s_L^- - s_L^+ + a_L = q_L,\ (d,p) \in \mathcal{D}_\mathcal{P},\ L \in \mathcal{L}_{dp}.$
- La nouvelle fonction objectif est $\min\big(\sum_{(d,p) \in \mathcal{D}_\mathcal{P}} \sum_{L \in \mathcal{L}_{dp}} a_L + \sum_{k \in \mathcal{K}} a_k\big)$.

Cette nouvelle relaxation continue, qui résulte de l'ajout de variables artificielles à la relaxation (R^u) du problème (P^u) au noeud u, est notée (R^{ua}) et est formellement définie à la figure 2.3. GC_{ϕ_1} consiste donc à résoudre ce nouveau problème par l'algorithme GC. Le quadruplet obtenu, noté $\big((R^{una}), x_{ua}^n, z_{ua}^n, z_{ua}^*\big)$, est ensuite retourné à GC_{ϕ_2} [14].

Enfin, la relaxation continue (R^{ua}) ainsi que le premier problème restreint (R^{u1a}), qui est fourni à GC, sont tous les deux réalisables étant donné l'existence de la solution suivante :

$$(\ [x_{ks}]\ ,\ [s_L^-]\ ,\ [s_L^+]\ ,\ [a_L]\ ,\ [a_k]\ ) = (\ [0]\ ,\ [0]\ ,\ [0]\ ,\ [q_L]\ ,\ [1]\ ).$$

[14] Ce quadruplet correspond à $\big((\dot{R}^{un}), \dot{x}_u^n, \dot{z}_u^n, \dot{z}_u^*\big)$, qui est le résultat de GC.

$$\min \Big(\sum_{(d,p)\in\mathcal{D}_\mathcal{P}} \sum_{L\in\mathcal{L}_{dp}} a_L \; + \; \sum_{k\in\mathcal{K}} a_k \Big)$$

s.l.c. :

$$\sum_{k\in\mathcal{K}} \sum_{s\in S_k^u} \sum_{t\in\mathcal{T}_k} \alpha_{tp}\; b_{kL}\; a_{ksdt}\; x_{ks} + s_L^- - s_L^+ + a_L = q_L,\;\; (d,p)\in\mathcal{D}_\mathcal{P},\;\; L\in\mathcal{L}_{dp}$$

$$\sum_{s\in S_k^u} x_{ks} + a_k = 1 \quad k\in\mathcal{K}$$

$$a_{ksdt} = 1 \qquad (d,t)\in\mathcal{C}^u_{k(one)},\;\; s\in S_k^u,\;\; k\in\mathcal{K}$$
$$a_{ksdt} = 0 \qquad (d,t)\in\mathcal{C}^u_{k(zero)},\;\; s\in S_k^u,\;\; k\in\mathcal{K}$$

$$a_{ksd_1t_1} = a_{ksd_2t_2} \qquad (d_1,t_1,d_2,t_2)\in\mathcal{C}^u_{k=},\;\; s\in S_k^u,\;\; k\in\mathcal{K}$$
$$a_{ksd_1t_1} \neq a_{ksd_2t_2} \qquad (d_1,t_1,d_2,t_2)\in\mathcal{C}^u_{k\neq},\;\; s\in S_k^u,\;\; k\in\mathcal{K}$$

$$0\le x_{ks}\le 1 \qquad k\in\mathcal{K},\;\; s\in S_k^u$$

$$0\le s_L^-\le q_L^- \qquad (d,p)\in\mathcal{D}_\mathcal{P},\;\; L\in\mathcal{L}_{dp}$$
$$0\le s_L^+\le q_L^+ \qquad (d,p)\in\mathcal{D}_\mathcal{P},\;\; L\in\mathcal{L}_{dp}$$

$$0\le a_k\le 1 \qquad k\in\mathcal{K}$$

$$0\le a_L\le q_L \qquad (d,p)\in\mathcal{D}_\mathcal{P},\;\; L\in\mathcal{L}_{dp}$$

Figure 2.3: Relaxation continue (R^{ua}) du problème (P^u) au noeud u de l'arbre de recherche.

2.3.4 Condition de réalisabilité du problème (R^u)

Après avoir utilisé la méthode GC pour résoudre la relaxation continue (R^{ua}), il suffit d'utiliser la valeur optimale z^*_{ua} pour décider si le problème (R^u) est réalisable. Notons que le résultat obtenu est le dernier quadruplet que cet algorithme a considéré : $((R^{una}), x^n_{ua}, z^n_{ua}, z^*_{ua})$ [15].

Pour simplifier le problème, supposons que la valeur z^*_{ua} a été atteinte pour que le test $z^n_{ua} \stackrel{?}{=} z^*_{ua}$ soit vrai. Ainsi, à l'aide de cette valeur optimale, nous allons définir une condition nécessaire et suffisante pour déterminer si la relaxation continue (R^u) du problème (P^u) est réalisable.

Considérons une solution de (R^{ua}) notée ($[x_{ks}]$, $[s^-_L]$, $[s^+_L]$, $[a_L]$, $[a_k]$). Cette solution est aussi réalisable à (R^u) si et seulement si $[a_L] = [a_k] = 0$. Or, si la valeur optimale de (R^{ua}) est telle que $z^*_{ua} > 0$, il n'y a aucune solution de cette relaxation continue qui est telle que $[a_L] = [a_k] = 0$ et le problème (R^u) n'est donc pas réalisable.

Ensuite, si $z^*_{ua} = 0$, (R^u) est réalisable car n'importe laquelle des solutions optimales de (R^{ua}) est telle que $[a_L] = [a_k] = 0$ et est alors une solution réalisable à (R^u). Ainsi, le théorème suivant découle directement de ces conclusions et n'a donc pas besoin de preuve.

Théorème 2.1 (Condition de réalisabilité de (R^u)) *: Le problème (R^u) est réalisable si et seulement si la valeur optimale de (R^{ua}), notée z^*_{ua}, est telle que $z^*_{ua} = 0$.*

Finalement, il est important de noter que cette condition, qui est nécessaire et suffisante à la réalisabilité du problème (R^u), est une condition nécessaire à la réalisabilité du problème (P^u).

[15] Voir la section 2.3.1 pour la description de ce résultat.

2.3.5 Description de la seconde phase GC_{ϕ_2}

Lors de la première phase GC_{ϕ_1}, la relaxation (R^{ua}) a été résolue à l'aide de l'algorithme GC. Maintenant, la seconde phase GC_{ϕ_2} consiste à résoudre la relaxation (R^u), soit celle qui nous intéresse, à l'aide de l'algorithme GC. Evidemment, il faut déterminer si cette relaxation est réalisable d'après la valeur optimale z^*_{ua} de (R^{ua}) et grâce au théorème 2.1.

Alors, si $z^*_{ua} = 0$, (R^u) est réalisable et toutes les variables artificielles a_L et a_k sont retirées de (R^{una}) afin d'obtenir la première relaxation continue restreinte (R^{u1}) de (R^u). En effet, il est possible d'éliminer ces variables artificielles car elle sont telles que $[a_L] = [a_k] = 0$ dans la solution optimale x^n_{ua}, qui demeure réalisable au problème restreint (R^{u1}). Par conséquent, les deux conditions nécessaires à l'application de GC pour résoudre (R^u), qui stipulent que cette relaxation doit être réalisable et qu'il doit exister une solution réalisable à (R^{u1}), sont toutes les deux satisfaites (voir la section 2.3.1).

Précision concernant la phase GC_{ϕ_1}

Enfin, il est important de préciser que l'intérêt d'appliquer GC_{ϕ_1} au problème (R^{ua}) n'est pas seulement de vérifier que le problème (R^u) est réalisable. En effet, l'emploi de la première phase de la méthode de génération de colonnes a aussi pour objectif de définir le premier problème restreint (R^{u1}), qui est nécessaire à la résolution de (R^u) par GC_{ϕ_2}.

Rappelons que l'objectif d'appliquer une méthode de séparation au noeud antérieur de u, qu'on note v, est d'éliminer la réalisabilité de la solution x^n_v pour la nouvelle relaxation continue (R^u). Par contre, il est aussi possible que toutes les solutions qui étaient réalisables au problème (R^v) ne le sont plus au problème (R^u).

Autrement dit, le retrait de colonnes au dernier problème restreint (R^{vn}) peut définir un problème restreint, que nous nommons le "problème transitoire du noeud v au noeud u, irréalisable. C'est-à-dire qu'il ne possède aucune solution et ne peut donc être résolu par le Simplexe. C'est pour cette raison qu'on ajoute les variables artificielles a_k et a_L à ce problème transitoire, le résultat obtenu est le premier problème restreint (R^{u1a}). Comme nous l'avons déjà mentionné, GC_{ϕ_1} se servira de (R^{u1a}) pour résoudre le problème (R^{ua}) et déterminer si le problème (R^u) est réalisable. Toutefois, il est aussi possible que le problème transitoire de v à u demeure réalisable au noeud u. Dans ce cas, le premier problème restreint (R^{u1}) est identique à ce problème transitoire et la première phase GC_{ϕ_1} ne nécessitera que d'une itération car la valeur optimale $z^*_{ua} = 0$ est immédiatement prouvée.

2.3.6 Organigramme et algorithme de la méthode de génération de colonnes en 2 phases

Nous pouvons voir ci-dessous l'algorithme 2.3 nommé GC_METHOD. Il décrit formellement la méthode de génération de colonnes en deux phases que nous venons de décrire. Cette méthode est utilisée lors des parcours DFS et BSFS, que nous verrons à la section 2.5, pour résoudre la relaxation continue (R^u) du problème (P^u) au noeud u de l'arbre de recherche.

De manière à simplifier la compréhension de l'algorithme de branch and price, nous supposons que la première phase GC_{ϕ_1} n'applique pas de branchement prématuré. Par conséquent, le test $z^n_{ua} \stackrel{?}{=} z^*_{ua}$ est vrai en tout temps et c'est pour cette raison qu'il est possible d'utiliser la valeur optimale z^*_{ua} pour vérifier si la relaxation (R^u) est réalisable.

Finalement, nous pouvons voir à la figure 2.4 l'organigramme de GC_METHOD, il

représente très bien l'algorithme GC et les 2 phases GC_{ϕ_1} et GC_{ϕ_2}. Aussi, les résultats obtenus par GC_METHOD sont représentés par le quintuplet $\left((R^{un}), x_u^n, z_u^n, z_u^*, z_{ua}^*\right)$.

Algorithme 2.3 Méthode de génération de colonnes en deux phases pour la résolution de (R^u) au noeud u (v est le noeud père).

GC_METHOD((R^u) **: relaxation continue de** (P^u) **)** $\Longrightarrow \left(R^{un}, x_u^n, z_u^n, z_u^*, z_{ua}^*\right)$

{Pre-traitement pour le calcul des ensembles $\mathcal{I}_k^u$:}
pour tout $k \in \mathcal{K}$ **effectuer**
  si $\left(\mathcal{C}_{k(one)}^u \neq \mathcal{C}_{k(one)}^v \bigvee \mathcal{C}_{k(zero)}^u \neq \mathcal{C}_{k(zero)}^v \bigvee \mathcal{C}_{k=}^u \neq \mathcal{C}_{k=}^v \bigvee \mathcal{C}_{k\neq}^u \neq \mathcal{C}_{k\neq}^v\right)$ **alors**
    calculer $\mathcal{I}_k^u$
  sinon
    $\mathcal{I}_k^u \leftarrow \mathcal{I}_k^v$ {L'ensemble des RFRC possibles n'a pas changé.}

{La phase GC_{ϕ_1} : résolution de (R^{ua})}
- Définir (R^{u1a}) par l'insertion des variables artificielles.
- $\left(R^{una}, x_{ua}^n, z_{ua}^n, z_{ua}^*\right) \leftarrow GC(R^{u1a})$

si $(\ z_{ua}^* = 0\ )$ **alors**
  {La phase GC_{ϕ_2} : résolution de (R^u)}
  - $R^{u1} \leftarrow R^{una}$ – {variables artificielles}
  - $\left(R^{un}, x_u^n, z_u^n, z_u^*\right) \leftarrow GC(R^{u1})$

retourner $\left(R^{un}, x_u^n, z_u^n, z_u^*, z_{ua}^*\right)$

2.4 Méthodes de séparation

Après avoir résolu le problème (R^u) à l'aide de l'algorithme GC_METHOD, une méthode de séparation, ou règle de branchement, est appliquée au problème (P^u) pour créer ses deux problèmes fils qui seront utilisés pour explorer son espace de solutions. Considérons le quintuplet $(R^{un}, x_u^n, z_u^n, z_u^*, z_{ua}^*)$ obtenu par l'algorithme GC_METHOD. Une telle séparation a pour objectif d'éliminer la solution fractionnaire x_u^n du problème relaxé (R^u) et de partitionner l'ensemble de solutions (entières) du problème (P^u) en deux sous-ensembles disjoints.

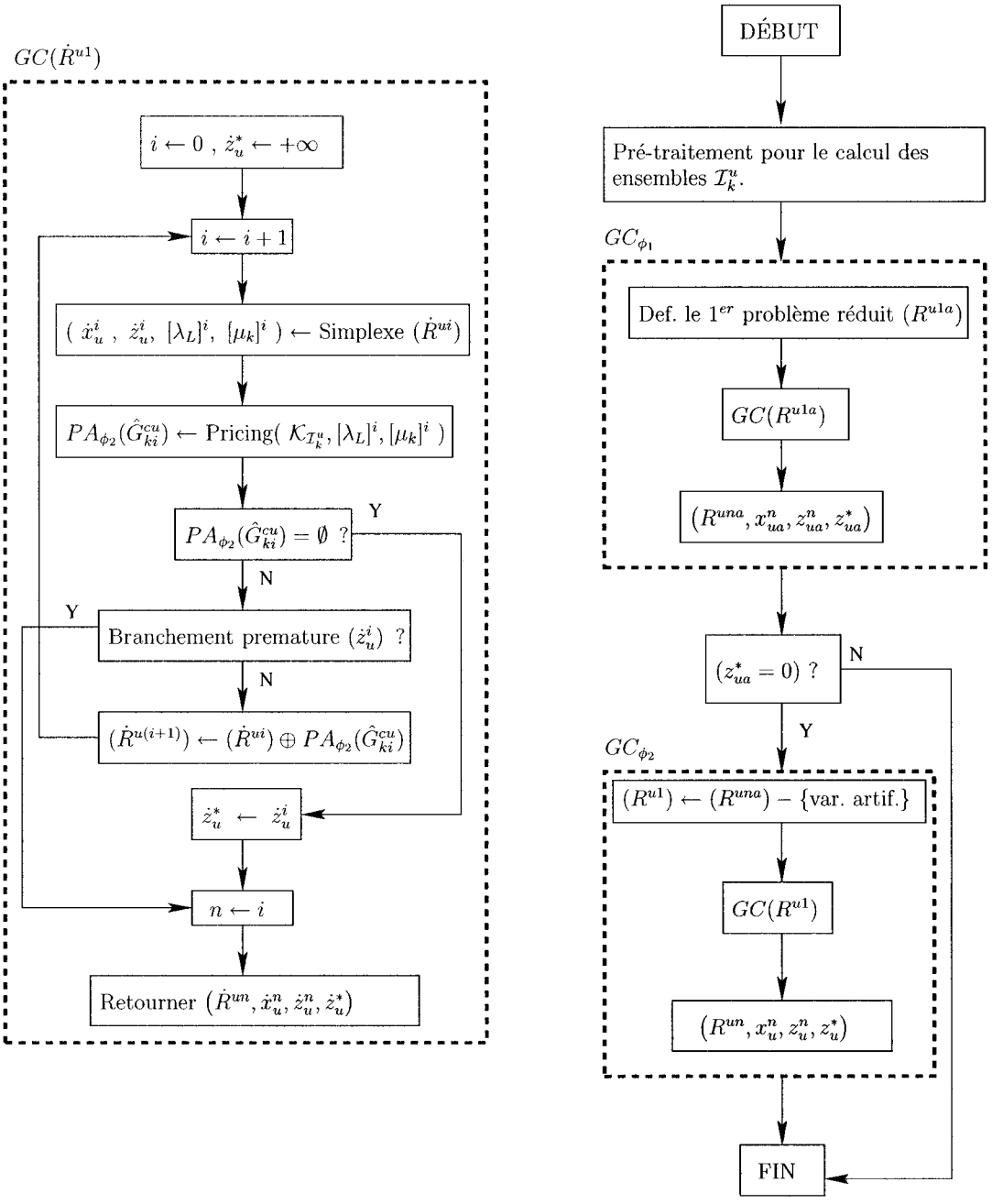


Figure 2.4: Organigramme de la méthode de génération de colonnes en deux phases pour la résolution de (R^u).

Toutefois, si $z_u^n = z_u^*$ et si $\lceil z_u^* \rceil \geq z^*$, aucune séparation n'est effectuée car le noeud u est coupé et ce même si x_u^n est fractionnaire. Plus de détails seront donnés à la section 2.6.2 pour décrire dans quels cas le noeud u est coupé et ceux dans lesquels il ne l'est pas.

Dans la section 2.4, nous commençons par donner la définition d'une séparation disjointe et complète. Ensuite, nous introduirons le lecteur aux branchements Ryan-Foster et binaire, ceux-ci seront approfondis au chapitre 3. Troisièmement, nous allons définir les deux classes de séparation possibles : en base et à demi en base. Nous concluons en affirmant qu'une séparation en base est plus efficace que celle à demi en base et en décrivant dans quel cas cette règle fait exception.

2.4.1 Séparation disjointe et complète

Une séparation appliquée au noeud u consiste à partitionner l'ensemble de solutions S^u du PLNE (P^u) en deux sous-ensembles disjoints : $S^v \subseteq S^u$ et $S^w \subseteq S^u$. S^v et S^w sont respectivement l'ensemble de solutions du problème (P^v) au noeud v et du problème (P^w) au noeud w.

Notons par $[s_{k_1}, s_{k_2}, \ldots, s_{k_{|\mathcal{K}|}}] \in S$, où pour $k \in \mathcal{K}$ $s_k \in S_k$, un horaire général réalisable quelconque. Alors, pour aider à la compréhension de ce mémoire mais surtout pour préparer notre argumentation concernant l'exhaustivité de notre algorithme de branch and price, nous devons introduire deux nouvelles définitions.

Définition 2.2 (Séparation disjointe et complète) *Une séparation appliquée au problème (P^u) est dite disjointe et complète si et seulement si les trois ensembles S^u, S^v et S^w sont tels que*

- $S^v \bigcap S^w = \emptyset$,
- $S^v \bigcup S^w = S^u$.

Ce qui revient à dire que $\forall [s_{k_1}, s_{k_2}, \ldots, s_{k_{|\mathcal{K}|}}] \in S^u$, *soit que* $[s_{k_1}, s_{k_2}, \ldots, s_{k_{|\mathcal{K}|}}] \in S^v$ *ou que* $[s_{k_1}, s_{k_2}, \ldots, s_{k_{|\mathcal{K}|}}] \in S^w$. *Ou en d'autres termes, tout horaire général qui est réalisable au problème* (P^u) *ne conserve sa réalisabilité que pour un et un seul des deux problèmes* (P^v) *et* (P^w).

Une règle de branchement, qu'elle soit Ryan-Foster ou binaire, s'applique à une infirmière $k \in \mathcal{K}$. Une telle règle partitionne l'ensemble des horaires individuels de l'infirmière k qui sont réalisables au problème (P^u). Cet ensemble est noté $S_k^u \subseteq S_k$ et le partitionnement ainsi créé consiste en deux sous-ensembles $S_k^v \subset S_k^u$ et $S_k^w \subset S_k^u$. Ces deux ensembles ont la même signification avec leur problème associé, respectivement (P^v) et (P^w), que celle qui est définie entre l'ensemble S_k^u et son problème associé (P^u).

Définition 2.3 (Branchement disjoint et complet) *Un branchement appliqué à une infirmière* $k \in \mathcal{K}$ *et au problème* (P^u) *est dit disjoint et complet si et seulement si les trois ensembles d'horaires individuels* S_k^u, S_k^v *et* S_k^w *sont tels que*

- $S_k^v \bigcap S_k^w = \emptyset$,
- $S_k^v \bigcup S_k^w = S_k^u$.

Ce qui veut dire que tout horaire individuel de l'infirmière de branchement k, *qui est réalisable au problème* (P^u), *est réalisable à un et un seul des deux problèmes* (P^v) *et* (P^w). *Formellement,* $\forall s \in S_k^u : \ (s \in S_k^v \ \wedge \ s \notin S_k^w) \ \vee \ (s \notin S_k^v \ \wedge \ s \in S_k^w)$.

Enfin, d'après les définitions des branchements Ryan-Foster et binaire, le lecteur pourra conclure de façon triviale qu'ils sont disjoints et complets. Par conséquent,

on peut affirmer que toutes les séparations de notre algorithme de branch and price sont aussi disjointes et complètes.

2.4.2 Règles de branchement Ryan-Foster et binaire

Deux méthodes de séparation sont possibles pour le partionnement de l'ensemble S^u : règle de branchement Ryan-Foster ou règle de branchement binaire. La règle de branchement choisie est effectuée sur le problème (P^u) afin de définir ses deux problèmes fils (P^v) et (P^w) et s'applique à $k \in \mathcal{K}$, qui est nommée "l'infirmière de branchement". De plus, la contrainte de branchement qui est ajoutée à l'un des problèmes est la négation de celle qui est ajoutée à l'autre, mais elles concernent toutes les deux cette infirmière.

Avant de définir les deux règles de branchement, il est important d'aviser le lecteur qu'elles sont toutes les deux définies selon le problème restreint (R^{un}) et non pas (R^u). C'est pour cette raison que les deux variables s_1 et $s_2 \in S_k$ (pour $k \in \mathcal{K}$), qui sont choisies pour définir l'un ou l'autre de ces branchements, doivent être telles que s_1 et $s_2 \in S_k^{un}$ et non pas seulement s_1 et $s_2 \in S_k^u$.

Définition 2.4 (Branchement Ryan-Foster) *Soit* $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{T_k}}$. *Si* $\exists s_1$, $s_2 \in S_k^{un}$ $(s_1 \neq s_2)$ *tel que*

- $a_{ks_1d_it_i} = a_{ks_2d_it_i}$,
- $a_{ks_1d_jt_j} \neq a_{ks_2d_jt_j}$,

il est alors possible de définir une règle de branchement de type Ryan-Foster sur l'infirmière de branchement k *et sur la paire d'affectation* (d_i, t_i, d_j, t_j). *Cette règle est notée* (k, d_i, t_i, d_j, t_j) *et définit comme suit les deux nouveaux problèmes nés de* (P^u).

- **Le problème** (P^v) **:**

  $\forall s \in S_k^v,\ a_{ksd_it_i} = a_{ksd_jt_j}\ \left(\ \mathcal{C}_{k=}^v \leftarrow \mathcal{C}_{k=}^u \bigcup \{(d_i, t_i, d_j, t_j)\}\ \right).$

- **Le problème** (P^w) **:**

  $\forall s \in S_k^w,\ a_{ksd_it_i} \neq a_{ksd_jt_j}\ \left(\ \mathcal{C}_{k\neq}^w \leftarrow \mathcal{C}_{k\neq}^u \bigcup \{(d_i, t_i, d_j, t_j)\}\ \right).$

Une représentation des conditions nécessaires à la définition d'un branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$ est donnée à la figure 2.5. Dans cette figure, l'existence des deux variables s_1 et $s_2 \in S_k^{un}$ d'une infirmière $k \in \mathcal{K}$ permet de définir la règle de branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$. Ce branchement peut être défini car ces deux variables sont telles que $a_{ks_1d_it_i} = a_{ks_2d_it_i} = 1$, $a_{ks_1d_jt_j} = 1$ et $a_{ks_2d_jt_j} = 0$. Toutefois, il est important de souligner que les deux conditions $a_{ks_1d_it_i} = a_{ks_2d_it_i}$ et $a_{ks_1d_jt_j} \neq a_{ks_2d_jt_j}$ peuvent chacune être satisfaites de deux façons. Nous discuterons plus de ce sujet à la section 3.1.

	s_1	s_2
	⋮	⋮
(d_i, t_i)	1	1
(d_j, t_j)	1	0
	⋮	⋮

Figure 2.5: Exemple d'une paire de variables s_1 et $s_2 \in S_k^{un}$ respectant les conditions nécessaires à la définition du branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$.

Définition 2.5 (Branchement binaire) *Soit* $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{\mathcal{T}_k}}$. *Si* $\exists s_1, s_2 \in S_k^{un}$ $(s_1 \neq s_2)$ *tel que* $a_{ks_1dt} \neq a_{ks_2dt}$, *il est alors possible de définir une règle de branchement de type binaire sur l'infirmière de branchement* k *et sur l'affectation* (d, t).

Cette règle est notée (k, d, t) et définit comme suit les deux nouveaux problèmes nés de (P^u).

- **Le problème (P^v) :**
  $\forall s \in S_k^v,\ a_{ksdt} = 1\ (\ \mathcal{C}_{k(one)}^v \leftarrow \mathcal{C}_{k(one)}^u \bigcup \{(d,t)\}\ ).$
- **Le problème (P^w) :**
  $\forall s \in S_k^w,\ a_{ksdt} = 0\ (\ \mathcal{C}_{k(zero)}^w \leftarrow \mathcal{C}_{k(zero)}^u \bigcup \{(d,t)\}\ ).$

La figure 2.6 donne une représentation des conditions nécessaires à la définition d'un branchement binaire $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{\mathcal{T}_k}}$. Dans cette figure, l'existence des deux variables s_1 et $s_2 \in S_k^{un}$ permet de définir le branchement binaire $(k, d_1, t_1) \in \mathcal{D}_{\mathcal{T}_k}$, car cette paire de variables est telle que $a_{ks_1d_1t_1} = 1$ et $a_{ks_2d_1t_1} = 0$.

	s_1	s_2
	⋮	⋮
(d_1, t_1)	1	0
	⋮	⋮

Figure 2.6: Exemple d'une paire de variables s_1 et $s_2 \in S_k^{un}$ respectant les conditions nécessaires à la définition du branchement binaire $(k, d_1, t_1) \in \mathcal{D}_{\mathcal{T}_k}$.

Il est important de choisir judicieusement la règle de branchement appliquée au noeud u de l'arbre de recherche. En effet, que cette règle soit de type Ryan-Foster ou binaire, elle peut affecter de deux façons différentes la réalisabilité de la solution fractionnaire x_u^n. C'est pourquoi nous allons définir à la section 2.4.3 deux classes de séparation possibles.

Enfin, nous référons le lecteur à la "Liste des sigles et abréviations" pour la définition formelle des ensembles de contraintes de branchement $\mathcal{C}_{k(one)}^u$, $\mathcal{C}_{k(zero)}^u$, $\mathcal{C}_{k=}^u$ et

$\mathcal{C}^u_{k\neq}$. Ces ensembles représentent les contraintes de branchement ajoutées au modèle initial (P) pour ainsi obtenir le PLNE (P^u) au noeud u. Ce problème est défini par la figure 2.7.

Finalement, soit un noeud u de l'arbre de recherche et une infirmière $k \in \mathcal{K}$. L'ensembles des horaires réalisables pour l'infirmière k au PLNE (P^u) est défini ci-dessous.

$$S^u_k = \left\{ s \in S_k \;\middle|\; \begin{array}{ll} \forall (d_i,t_i,d_j,t_j) \in \mathcal{C}^u_{k=}, & a_{ksd_it_i} = a_{ksd_jt_j}, \\ \forall (d_i,t_i,d_j,t_j) \in \mathcal{C}^u_{k\neq}, & a_{ksd_it_i} \neq a_{ksd_jt_j}, \\ \forall (d,t) \in \mathcal{C}^u_{k(one)}, & a_{ksdt} = 1, \\ \forall (d,t) \in \mathcal{C}^u_{k(zero)}, & a_{ksdt} = 0. \end{array} \right\} \tag{2.1}$$

$$\min \Big(\sum_{k\in\mathcal{K}} \sum_{s\in S^u_k} p_{ks}\, x_{ks} + \sum_{(d,p)\in\mathcal{D}_\mathcal{P}} \sum_{L\in\mathcal{L}_{dp}} (p^-\, s^-_L + p^+\, s^+_L) \Big)$$

s.l.c. :

$$\sum_{k\in\mathcal{K}} \sum_{s\in S^u_k} \sum_{t\in\mathcal{T}_k} \alpha_{tp}\, b_{kL}\, a_{ksdt}\, x_{ks} + s^-_L - s^+_L = q_L \;,\; (d,p) \in \mathcal{D}_\mathcal{P},\; L \in \mathcal{L}_{dp}$$

$$\sum_{s\in S^u_k} x_{ks} = 1 \qquad k \in \mathcal{K}$$

$$a_{ksdt} = 1 \qquad (d,t) \in \mathcal{C}^u_{k(one)},\; s \in S^u_k,\; k \in \mathcal{K}$$

$$a_{ksdt} = 0 \qquad (d,t) \in \mathcal{C}^u_{k(zero)},\; s \in S^u_k,\; k \in \mathcal{K}$$

$$a_{ksd_it_i} = a_{ksd_jt_j} \qquad (d_i,t_i,d_j,t_j) \in \mathcal{C}^u_{k=},\; s \in S^u_k,\; k \in \mathcal{K}$$

$$a_{ksd_it_i} \neq a_{ksd_jt_j} \qquad (d_i,t_i,d_j,t_j) \in \mathcal{C}^u_{k\neq},\; s \in S^u_k,\; k \in \mathcal{K}$$

$$x_{ks} \in \{0,\, 1\} \qquad k \in \mathcal{K},\; s \in S^u_k$$

$$s^-_L \in \{0,\, 1,\, \ldots,\, q^-_L\} \qquad (d,p) \in \mathcal{D}_\mathcal{P},\; L \in \mathcal{L}_{dp}$$

$$s^+_L \in \{0,\, 1,\, \ldots,\, q^+_L\} \qquad (d,p) \in \mathcal{D}_\mathcal{P},\; L \in \mathcal{L}_{dp}$$

Figure 2.7: Le PLNE (P^u) au noeud u de l'arbre de recherche.

2.4.3 Deux classes de séparation possibles

Après avoir choisi la règle de branchement qui sera appliquée au problème courant (P^u), il est possible que la solution fractionnaire x_u^n de la relaxation continue (R^u) soit encore réalisable et donc optimale pour une et une seule des deux nouvelles relaxations (R^v) et (R^w). C'est pourquoi il est important de choisir judicieusement cette règle de branchement dans le but d'obtenir une séparation efficace et d'éliminer, si possible, la réalisabilité de la solution x_u^n pour ces deux problèmes.

Notons par $(B^u,\ N^u)$, le partitionnement de l'ensemble des indices des variables de la base réalisable x_u^n, où B^u correspond aux variables en base et N^u aux variables hors-base. Ici, on suppose que x_u^n n'est pas une solution de base dégénérée [16]. Le branchement appliqué à (P^u) affecte $(B^u,\ N^u)$ et a pour résultat les deux nouveaux partitionnements suivants : $(B^v,\ N^v)$ et $(B^w,\ N^w)$. Ces derniers sont respectivement associés aux deux nouvelles relaxations continues (R^v) et (R^w).

De façon plus précise pour le problème (P^v), l'application de cette règle de branchement crée un sous-ensemble de variables qui ne sont plus valides à ce problème et qui sont alors retirées du partitionnement $(B^u,\ N^u)$; le résultat obtenu est $(B^v,\ N^v)$. Le même principe s'applique au problème (P^w) mais à partir d'un sous-ensemble de variables disjoint de celui du problème (P^v) et le résultat obtenu est $(B^w,\ N^w)$.

En considérant la solution x_u^n et les trois partitionnements $(B^u,\ N^u)$, $(B^v,\ N^v)$ et $(B^w,\ N^w)$, il y a deux classes de séparation possibles : en base et à demi en base.

[16] Nous rappelons au lecteur qu'une solution de base est dite "dégénérée" si et seulement si elle contient au moins une variable en base avec une valeur de 0.

Définition 2.6 (Séparation en base) *On appelle une séparation en base, une séparation qui possède les caractéristiques suivantes résultant de la règle de branchement appliquée à* (P^u) *:*

- $B^v \subset B^u$ ($\exists\ s \in B^u$ *tel que* $s \notin B^v$) *et* $N^v \subseteq N^u$,
- $B^w \subset B^u$ ($\exists\ s \in B^u$ *tel que* $s \notin B^w$) *et* $N^w \subseteq N^u$.

Pour une telle séparation, la solution x_u^n *n'est plus réalisable pour aucune des relaxations continues* (R^v) *et* (R^w).

Définition 2.7 (Séparation à demi en base) *On appelle une séparation à demi en base, une séparation qui possède les caractéristiques suivantes résultant de la règle de branchement appliquée à* (P^u) *:*

- $B^v \subset B^u$ ($\exists\ s \in B^u$ *tel que* $s \notin B^v$) *et* $N^v \subseteq N^u$,
- $B^w = B^u$ *et* $N^w \subset N^u$ ($\exists\ s \in N^u$ *tel que* $s \notin N^w$).

Pour une telle séparation, la solution x_u^n *n'est plus réalisable au problème* (R^v) *mais elle l'est encore au problème* (R^w)*; elle est d'ailleurs une solution optimale de ce dernier.*

Enfin, pour les 3 partitionnements énumérés ci-dessus, une séparation hors-base possèderait les caractéristiques suivantes :

- $B^v = B^u$ et $N^v \subset N^u$ ($\exists\ s \in N^u$ tel que $s \notin N^v$),
- $B^w = B^u$ et $N^w \subset N^u$ ($\exists\ s \in N^u$ tel que $s \notin N^w$).

Par contre, étant donné que les branchements Ryan-Foster et binaire sont disjoints et complets, il est théoriquement impossible qu'ils définissent une telle séparation. En

effet, considérons l'infirmière de branchement $k \in \mathcal{K}$. Pour cette infirmière $\exists s' \in B^u$ tel que $s' \in S_k^{un}$, mais alors $s' \in B^v$ ($s' \in S_k^v$) et $s' \in B^w$ ($s' \in S_k^w$), ce qui est en contradiction avec $S_k^v \bigcap S_k^w = \emptyset$ (voir la définition 2.3).

2.4.4 Technique pour obtenir une séparation efficace

Pour conclure sur les méthodes de séparation, rappelons qu'un de leurs principaux objectifs est d'éliminer la réalisabilité de la solution x_u^n pour les deux nouveaux problèmes (R^v) et (R^w). La séparation en base est la seule à remplir cet objectif, tandis que la séparation à demi en base n'atteint cet objectif que pour un seul de ces problèmes. On peut alors affirmer que la séparation en base est plus efficace que celle à demi en base.

Pour s'assurer d'obtenir une séparation en base, la technique utilisée consiste simplement à définir le branchement à l'aide de deux variables s' et $s'' \in S_k^{un}$ $(s' \neq s'')$ tel que $0 < x_{ks'} < 1$ et $0 < x_{ks''} < 1$. Il est possible de trouver de tels horaires appartenant à une infirmière $k \in \mathcal{K}$, qui est l'infirmière de branchement, seulement lorsque la solution x_u^n est fractionnaire ($x_u^n \notin \mathbb{N}$). Dans de tels cas, il n'y a pas de problème pour sélectionner un branchement Ryan-Foster ou, si ce n'est pas possible, un branchement binaire qui définit une séparation en base.

Par contre, il peut arriver qu'on soit incité à définir une séparation à demi en base. En effet, ceci peut survenir lorsqu'un problème de dégénérescence apparaît sur une série de solutions entières terminant avec x_u^n. Dans ce cas, il est préférable d'arrêter prématurément la résolution du problème (R^u) à l'itération n (branchement prématuré), donc de ne pas atteindre la valeur optimale z_u^*, et de définir une séparation à demi en base à partir de la solution entière x_u^n.

La solution x_u^n sera encore réalisable pour un des deux problèmes (R^v) et (R^w) et il est fort probable qu'il y ait encore de la dégénérescence composée de solutions entières lors de la résolution de ce même problème. Malgré tout, la taille de cette dégénérescence va diminuer au fur et à mesure qu'on appliquera des séparations à demi en base au cours de l'algorithme de branch and price.

2.5 Exploration de l'arbre de recherche

Nous avons défini GC_METHOD (voir l'algorithme 2.3), qui est utilisé pour résoudre la relaxation continue (R^u) du problème (P^u) au noeud u de l'arbre de recherche, et les deux méthodes de séparation ont aussi été décrites. Alors, pour compléter notre description de l'algorithme de branch and price, il ne nous reste plus qu'à expliquer les deux parcours possibles pour l'exploration de l'arbre de recherche. Ces deux parcours sont le DFS et le BSFS.

Dans cette section, nous commençons par introduire le lecteur à ces deux parcours. Ensuite, leur algorithme respectif sera formellement décrit.

2.5.1 Introduction aux parcours DFS et BSFS

Premièrement, le DFS (Depth First Search) est l'algorithme pur de parcours en profondeur d'abord. À l'aide de GC_METHOD, il commence par résoudre la relaxation continue (R^{u^0}) au noeud racine u^0. Ensuite, si les conditions pour couper le noeud u^0 ne sont pas satisfaites (voir la section 2.6.2), une règle de branchement est choisie. À l'aide de cette dernière, le nouveau problème (P^v) est créé pour ensuite recommencer ce processus de résolution et d'exploration mais sur ce nouveau problème

en résolvant la relaxation continue (R^v) et (si nécessaire) en appliquant une méthode de séparation au problème (P^v). Lorsque ceci est terminé, c'est au tour du second fils de (P^{u^0}), qui est noté (P^w), d'être créé, résolu et exploré par le même processus. Quand l'ensemble de solutions entières du problème (P^{u^0}) a été implicitement exploré par ses deux fils (P^v) et (P^w), l'algorithme de branch and price est complété.

Il est important de préciser que le parcours DFS sélectionne en tout temps le noeud de gauche ((P^v)) avant celui de droite ((P^w)) pour continuer l'exploration. D'ailleurs, dans les cas où la séparation choisie est un branchement binaire, le DFS branche en tout temps sur le noeud où une nouvelle affectation est rendue obligatoire pour l'infirmière de branchement $k \in \mathcal{K}$.

Deuxièmement, le parcours BSFS (Best Son First Search) est légérement différent du parcours DFS. En effet, contrairement à ce dernier qui choisit arbitrairement le problème (P^v) avant (P^w) pour continuer l'exploration, le BSFS sélectionne celui qui est le plus prometteur. Pour faire cette sélection, les deux nouvelles relaxations (R^v) et (R^w) sont définies et résolues afin d'obtenir les valeurs optimales z_v^* et z_w^*, qui sont respectivement des bornes inférieures sur la valeur des solutions des problèmes (P^v) et (P^w). De cette façon, nous pouvons déterminer lequel de ces deux problèmes est le plus prometteur, soit celui dont la valeur de la rexation continue est inférieure à celle de l'autre.

Théoriquement, le parcours BSFS devrait trouver plus rapidement une solution de valeur optimale que le parcours DFS (nous verrons des résultats pour appuyer ceci au chapitre 5). Toutefois, le temps requis par ce dernier pour rencontrer une première solution devrait être inférieur à celui requis par le BSFS. Intuitivement, ceci s'explique par le fait qu'à chaque niveau de l'arbre de recherche, le BSFS résout

deux relaxations continues contrairement à une seule pour le DFS, ce qui nécessite approximativement le double du temps à chaque niveau.

Les figures 2.8 et 2.9 montrent respectivement des exemples de parcours DFS et BSFS. On suppose que le partionnement de l'ensemble S des solutions du PLNE (P) a été fait de façon identique par ces deux parcours. En d'autres termes, la règle de branchement qui est appliquée au PLNE (P^i), lors du parcours DFS, est la même qui est appliqué au PLNE (P^i), lors du parcours BSFS. Évidemment, les deux arbres ne sont pas les mêmes étant donné la nature de leur parcours respectif. Le DFS trouve la première solution après la résolution de seulement 4 relaxations continues (au problème (P^3)) contrairement à 7 pour le BSFS (au problème P^{10}). Par contre, le BSFS n'a besoin que de seulement 7 problèmes pour trouver la solution optimale alors que le DFS en nécessite 11.

L'exemple présenté ci-haut est entièrement utopique étant donné le partionnement de l'ensemble S qui est commun aux deux parcours DFS et BSFS. Malgré tout, il représente bien les avantages et les désavantages propres à ces deux parcours.

2.5.2 Les algorithmes DFS et BSFS

Nous pouvons voir à l'algorithme 2.4 une description complète du parcours "depth first search". Ce parcours est entamé par l'algorithme récursif DFS afin de résoudre le PLNE (P^{u^0}) du noeud racine u^0. Notons que ce problème est identique au PLNE (P). L'algorithme DFS débute en résolvant la relaxation continue (R^u) du noeud courant u. Ensuite, selon certaines conditions qui seront décrites à la section 2.6.2, elle s'appele deux fois : une première fois pour résoudre le nouveau PLNE (P^v) et une seconde fois pour résoudre le nouveau PLNE (P^w).

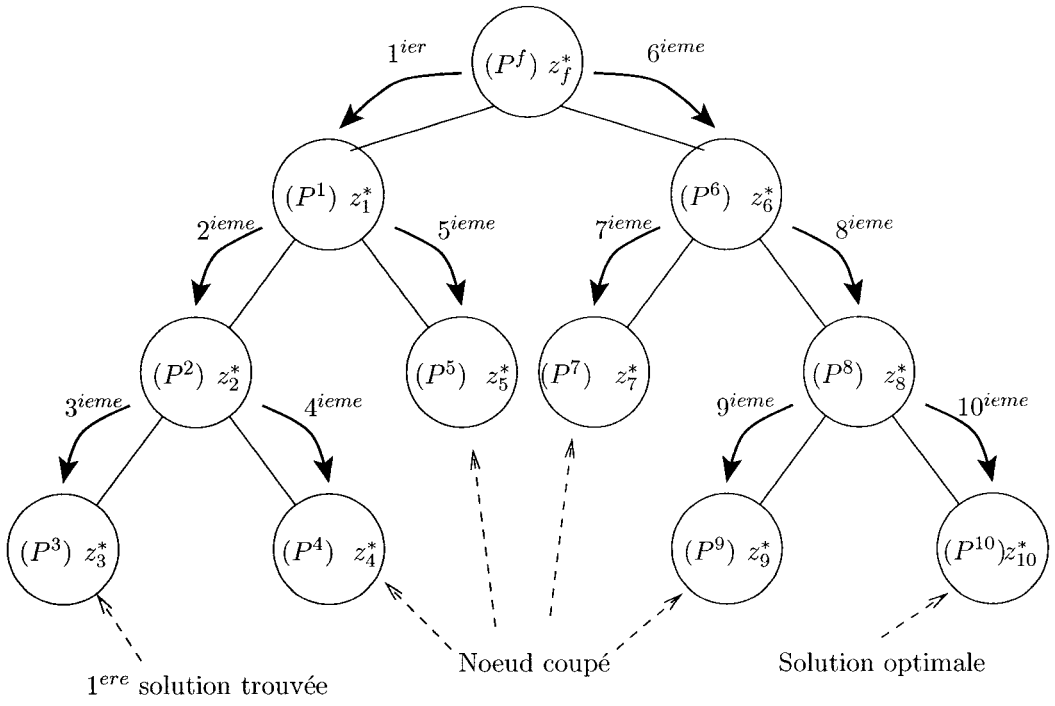


Figure 2.8: Exemple de parcours DFS.

Une description similaire à celle-ci, mais pour le parcours "best sun first search", est fournie avec l'algorithme 2.5. Maintenant, voici quelques observations qui sont communes à l'implantation de ces deux parcours.

Définition du PLNE (P^v) (idem pour (P^w))

Afin d'être fidèle à cette définition, les variables qui ne satisfont pas la nouvelle contrainte de branchement ajoutée au problème (P^v) doivent être retirées du modèle. Ceci doit se faire avant d'appliquer le parcours DFS ou BSFS pour résoudre le nouveau PLNE (P^v). Après, lorsque ceci est fait, il faut évidemment réinsérer les variables qui ont précédemment été retirées. Le retrait et la réinsertion d'une variable sont implantés en modifiant sa borne supérieure. Cette borne est posée à 0 lors du retrait et est remise à 1 lors de la réinsertion.

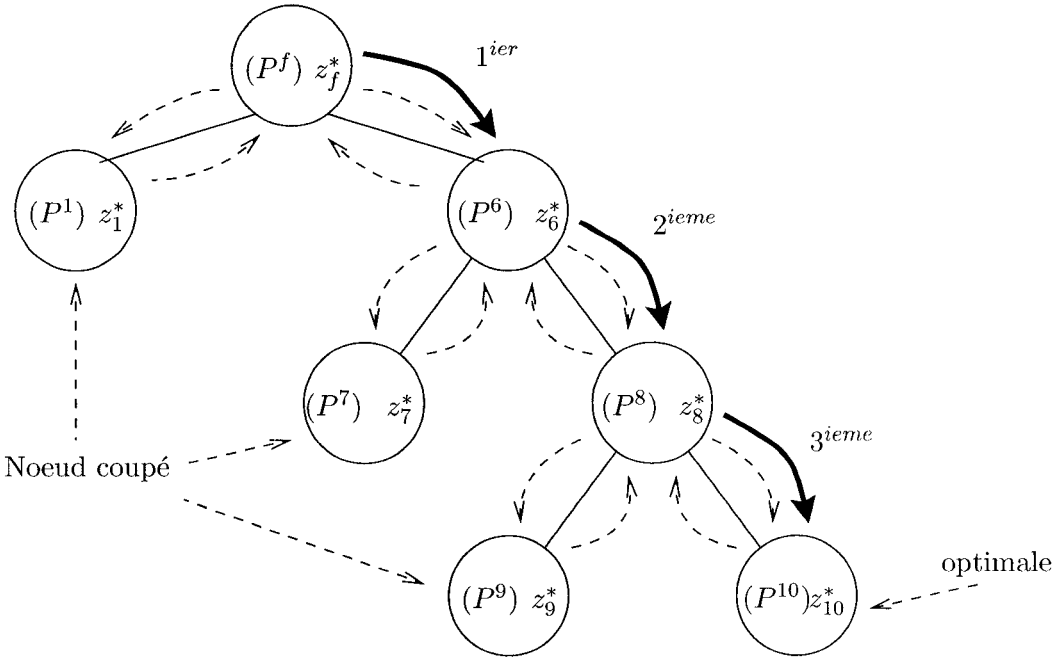


Figure 2.9: Exemple de parcours BSFS où $z_1^* > z_6^*$, $z_7^* > z_8^*$ et $z_9^* > z_{10}^*$.

Dans le parcours BSFS, le même principe s'applique lors de la définition des relaxations continues (R^v) et (R^w) selon leur PLNE, respectivement (P^v) et (P^w).

Solution entière et mise à jour de la borne supérieure

Dans les deux parcours DFS et BSFS, il est nécessaire de conserver la valeur de la meilleure solution du problème (P) qui est présentement connue. Tout au long de ces deux parcours, cette valeur est une borne supérieure sur la valeur optimale du PLNE (P), elle est notée par z^* et débute avec une valeur de $+\infty$.

Une solution réalisable peut être trouvée de deux façons : soit par la méthode d'arrondi ou lorsque la solution obtenue par GC, notée par x_u^n, est entière. La notation utilisée pour désigner une solution entière est $x_u^n \in \mathbb{N}$. Alors, si $x_u^n \in \mathbb{N}$ et si la valeur de cette solution est telle que $z_u^n < z^*$, la valeur z^* est mise à jour par z_u^n.

Ensuite, après la résolution de la relaxation continue (R^u) du noeud u de l'arbre de recherche, la méthode d'arrondi est utilisée afin de trouver un horaire général réalisable. Pour chaque infirmière $k \in \mathcal{K}$, cette méthode choisit au hasard un seul horaire $s \in S_k^u$. Si l'horaire général ainsi obtenu respecte toutes les contraintes de quotas et que sa valeur est inférieure à z^*, cette borne supérieure est mise à jour par la valeur de cet horaire général.

Finalement, quoique cette méthode d'arrondi est des plus simples et pourrait être remplacée par une méthode utilisant des heuristiques, nous verrons au chapitre 5 qu'elle a grandement amélioré les résultats de calculs.

Algorithme 2.4 Algorithme récursif du parcours DFS pour la résolution du PLNE (P^u) au noeud u.

DFS(P^u : le PLNE au noeud u) :

si $u = u^0$ **alors**
  $z^* \leftarrow +\infty$
$(R^{un}, x_u^n, z_u^n, z_u^*, z_{ua}^*) \leftarrow$ GC_METHOD (R^u) {Résolution de (R^u)}

si $(z_{ua}^* = 0 \ \wedge \ x_u^n \in \mathbb{N} \ \wedge \ z_u^n < z^*)$ **alors**
  $z^* \leftarrow z_u^n$ {Mise à jour de la borne supérieure}

si $((z_{ua}^* \neq 0) \bigvee (z_u^n = z_u^* \ \wedge \ (x_u^n \in \mathbb{N} \ \vee \ \lceil z_u^n \rceil \geq z^*)))$ **alors**
  Terminer {Le noeud u est coupé ("Backtrack") }

Générer une règle de branchement (Ryan-Foster ou binaire)
Définir le PLNE (P^v)
DFS(P^v) {Résoudre le PLNE (P^v) ; z^* peut diminuer}
si $(z_u^n = z_u^* \ \wedge \ \lceil z_u^n \rceil < z^*)$ **alors**
  Définir le PLNE (P^w)
  DFS(P^w) {Résoudre le PLNE (P^w)}

Terminer

Algorithme 2.5 Algorithme récursif du parcours BSFS pour la résolution du PLNE (P^u) au noeud u.

BSFS($\mathbf{P^u}$: le PLNE au noeud u) :

si $u = u^0$ **alors**
  $z^* \leftarrow +\infty$
  $\left(R^{un}, x_u^n, z_u^n, z_u^*, z_{ua}^*\right)$ $\leftarrow$ GC_METHOD (R^u) {Résolution de (R^u)}

si $(\ z_{ua}^* = 0 \ \wedge \ x_u^n \in \mathbb{N} \ \wedge \ z_u^n < z^*\ )$ **alors**
  $z^* \leftarrow z_u^n$ {Mise à jour de la borne supérieure}

si $(\ (z_{ua}^* \neq 0) \ \bigvee \ (\ z_u^n = z_u^* \ \wedge \ (\ x_u^n \in \mathbb{N} \ \vee \ \lceil z_u^n \rceil \geq z^*)\ )\ )$ **alors**
  Terminer {Le noeud u est coupé ("Backtrack") }

Générer une règle de branchement (Ryan-Foster ou binaire)

{Définition du meilleur des deux fils.}
Selon le PLNE (P^v), définir la relaxation continue (R^v)
$\left(R^{vn}, x_v^n, z_v^n, z_v^*, z_{va}^*\right)$ $\leftarrow$ GC_METHOD (R^v) {Résolution de (R^v)}
Selon le PLNE (P^w), définir la relaxation continue (R^w)
$\left(R^{wn}, x_w^n, z_w^n, z_w^*, z_{wa}^*\right)$ $\leftarrow$ GC_METHOD (R^w) {Résolution de (R^w)}

{Exploration en débutant avec le plus prometteur des deux fils.}
si $z_v^* < z_w^*$ **alors**
  Définir le PLNE (P^v).
  BSFS(P^v) {Résoudre le PLNE (P^v)}
  Définir le PLNE (P^w).
  BSFS(P^w) {Résoudre le PLNE (P^w)}
sinon
  Procéder dans l'ordre inverse.
Terminer

2.6 Exhaustivité du branch and price

Afin de parachever la description de notre algorithme de branch and price, il est important de montrer que cet algorithme est implicitement exhaustif. C'est-à-dire qu'il parcourt toutes les solutions du problème (P) d'une façon implicite afin de trouver un horaire général réalisable de valeur optimale z^*. Ceci est accompli grâce à la valeur optimale z_u^* de la relaxation (R^u) du problème (P^u) au noeud u de l'arbre de recherche et par les méthodes de séparations disjointes et complètes appliquées à chacun des problèmes (P^u) afin de définir leurs problèmes fils (P^v) et (P^w).

Nous commençons par montrer que l'arbre de recherche obtenu par l'application des branchements Ryan-Foster et binaire est exhaustif. Ensuite, nous décrivons les conditions nécessaires pour couper un noeud afin que les algorithmes DFS et BSFS soient des parcours implicitement exhaustifs.

2.6.1 Arbre de recherche exhaustif

Premièrement, il n'est pas garanti qu'il existe un branchement Ryan-Foster applicable pour la solution x_u^n du problème (R^u), peu importe que cette base soit entière ou non et que la séparation obtenue soit en base ou à demi en base. Considérons le problème restreint (R^{un}) et supposons que $\exists k \in \mathcal{K}$ telle que $|S_k^{un}| \geq 2$. Même sous cette condition, il est en effet possible qu'il n'existe aucune paire de variables s' et $s'' \in S_k^{un}$, pour toute infirmière $k \in \mathcal{K}$, définissant un branchement Ryan-Foster (voir la définition 2.4). Nous référons le lecteur à la proposition 1.4.1 de Labit [3] pour la preuve de cette affirmation.

Cependant, lorsqu'aucun branchement Ryan-Foster n'est applicable au problème (P^u) sous la même condition décrite ci-haut, il existe obligatoirement un branchement

binaire $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{\mathcal{T}_k}}$. Car dans le cas contraire, toutes les paires de variables s' et $s'' \in S_k^{un}$ $(s' \neq s'')$ de toute infirmière $k \in \mathcal{K}$ telle que $|S_k^{un}| \geq 2$ seraient identiques : $\forall (d, t) \in \mathcal{D}_{\mathcal{T}_k},\ a_{ks'dt} = a_{ks''dt}$. Ce qui est évidemment absurde.

Ensuite, étant donné que toute infirmière $k \in \mathcal{K}$ possède un nombre fini d'horaires réalisables, le nombre de branchements Ryan-Foster et binaire pouvant être successivement appliqués à cette infirmière est lui aussi fini. Ici, il est important de préciser que le terme "successivement appliqués" concerne uniquement les branchements appliqués aux problèmes situés sur un chemin commun de l'arbre de recherche, soit un chemin qui part du problème (P^{u^0}) du noeud racine u^0 et qui se termine au problème (P^u) d'une feuille u de ce même arbre. Notons qu'un noeud u est un noeud feuille si et seulement si chacune des infirmières $k \in \mathcal{K}$ possède un seul horaire réalisable au problème (P^u) : $\forall k \in \mathcal{K},\ |S_k^u| = 1$.

Troisièmement, on peut conclure de ce qui précède que le nombre de branchements Ryan-Foster et binaires requis pour atteindre chacun des noeuds feuilles est fini. De plus, étant donné que ces branchements sont des méthodes de séparations disjointes et complètes (voir la section 2.4.1), chaque horaire général $[s_{k_1}, s_{k_2}, \ldots, s_{k_{|\mathcal{K}|}}]$, où pour $k' \in \mathcal{K}$ $s_{k'} \in S_{k'}$, correspond à un unique problème "feuille" (P^u).

Finalement, on peut donc affirmer que l'arbre de recherche résultant de l'application d'un nombre fini de branchements Ryan-Foster et binaires est exhaustif. Ce qui revient à dire que toute solution, ou horaire général réalisable $[s_{k_1}, s_{k_2}, \ldots, s_{k_{|\mathcal{K}|}}] \in S$, correspond à un unique problème "feuille" (P^u), où $[s_{k_1}, s_{k_2}, \ldots, s_{k_{|\mathcal{K}|}}] \in S^u$.

2.6.2 Parcours implicitement exhaustif

Maintenant que nous avons montré que l'ensemble S des solutions réalisables du problème (P) peut être décomposé en un arbre de recherche exhaustif, il ne nous reste qu'à expliquer comment l'algorithme de branch and price peut se permettre de ne pas explorer un sous-arbre de cet arbre de recherche. Cette section consiste donc à définir les conditions nécessaires et suffisantes pour couper un noeud de l'arbre de recherche.

Après avoir résolu la relaxation continue (R^u) du problème (P^u) au noeud u, il est possible d'utiliser l'information obtenue par l'algorithme GC_METHOD pour décider s'il est nécessaire d'explorer l'ensemble de solutions S^u du problème (P^u). Soit $\big((R^{un}), x_u^n, z_u^n, z_u^*, z_{ua}^*\big)$, l'information obtenue par cet algorithme pour le problème (R^u) (voir la section 2.3.6 pour sa signification). Alors, on dit qu'on coupe le noeud u lorsqu'on ne partitionne pas l'ensemble S^u en deux sous-ensembles disjoints, c'est-à-dire qu'on ne descend pas plus profondément dans l'arbre de recherche. Voyons de plus prêt les scénarios possibles.

Tout d'abord, nous devons nous assurer que le problème (R^u) est réalisable en vérifiant que $z_{ua}^* = 0$. Nous référons le lecteur à la section 2.3.4 pour la description de cette condition de réalisabilité. S'il n'est pas réalisable, l'ensemble S^u des solutions réalisables du problème (P^u) est nécessairement vide et le noeud u est donc coupé. Sinon, deux cas peuvent se présenter.

Le premier cas, où $z_u^* = z_u^n$, est légèrement complexe. Notons ici que la valeur z_u^* de la solution x_u^n est une borne inférieure sur la valeur de chacune des solutions de l'ensemble S^u. De plus, ces solutions étant entières, leur valeur respective est

aussi entière (voir la section 1.3.3.4), donc $\lceil z_u^* \rceil$ est une meilleure borne inférieure sur ces valeurs. Premièrement, si la solution x_u^n est entière, le problème (P^u) est résolu et le noeud u est coupé. Deuxièmement, si nous avons $\lceil z_u^* \rceil \geq z^*$, aucune solution de l'ensemble S^u et de valeur inférieure à z^* n'existe, donc le noeud u est coupé. Enfin, lorsque $\lceil z_u^* \rceil < z^*$, le noeud u n'est pas coupé et l'exploration de l'ensemble S^u continue grâce aux deux nouveaux problèmes (P^v) et (P^w).

Le deuxième cas, où la valeur optimale z_u^* n'a pas été atteinte, survient lorsqu'un branchement prématuré a été décidé. Ici, même si la solution x_u^n est entière et (ou) si nous avons $\lceil z_u^n \rceil \geq z^*$, on ne peut pas couper le noeud car le problème n'a pas été entièrement résolu. En effet, il est possible qu'il existe une solution de l'ensemble S^u de valeur inférieure à celle de la meilleure solution du problème (P) actuellement connue, soit z^*. Encore une fois, il est donc nécessaire de continuer l'exploration de l'ensemble S^u par l'application d'un branchement Ryan-Foster ou binaire.

Pour terminer, nous avons vu lors de la description des parcours DFS et BSFS (voir la section 2.5) que ces deux algorithmes doivent vérifier les conditions énumérées ci-dessus pour se permettre d'omettre l'exploration d'un sous-arbre de l'arbre de recherche. Par conséquent, le lecteur pourra conclure que l'exploration de cet arbre par l'algorithme de branch and price trouvera une solution de valeur z^*, si le PLNE (P) est réalisable, et prouvera l'optimalité de cette valeur. Ceci, pour les deux parcours DFS et BSFS et sans toutefois avoir explicitement exploré l'arbre de recherche.

CHAPITRE 3 : LES RÈGLES DE BRANCHEMENT

Ce chapitre est entièrement consacré aux différentes méthodes de séparation. Nous commençons par approfondir le concept du branchement Ryan-Foster. Ensuite, la même chose est faite pour le branchement binaire, mais en plus nous verrons comment il peut être possible d'obtenir de nouvelles contraintes lors d'une telle séparation. Finalement, le concept de RFRC ou "Ryan-Foster respect combination", que nous avons déjà mentionné au chapitre 2, est expliqué en détail. Au même moment, l'algorithme calculer $\mathcal{I}_k^u$ sera expliqué et formellement décrit.

3.1 Règle de branchement Ryan-Foster

En 1981, lorsqu'ils considéraient le problème du partionnement d'ensemble, Ryan et Foster [23] ont introduit pour la première fois la règle de branchement qui porte leur nom. Même si leurs travaux ne portaient pas sur un algorithme de branch and bound, il s'est avéré dans les années ultérieures que leur règle de branchement soit très efficace à travers ce type d'algorithme. En outre, des applications utilisant cette règle à l'intérieur d'un branch and bound (ou d'un branch and price) sont présentées par les recherches de Vance et al. [18, 17] et celles de Desrochers et Soumis [24].

Dans cette section, nous commençons par expliquer la condition nécessaire à l'application d'un branchement Ryan-Foster en plus de donner sa définition. Ensuite, nous énumérons trois conditions à respecter pour obtenir un branchement Ryan-Foster efficace.

3.1.1 Condition nécessaire et définition

Bien que le branchement Ryan-Foster a déjà été défini à la section 2.4.2, il est important que le lecteur comprenne bien la distinction entre la condition nécessaire à l'application d'un tel branchement et sa définition.

Condition nécessaire pour un branchement Ryan-Foster

Premièrement, expliquons la condition nécessaire à l'application d'un branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$ au problème courant (P^u). Un tel branchement est possible si et seulement si $\exists s_1, s_2 \in S^{un}_k$ $(s_1 \neq s_2)$ qui respectent les deux conditions suivantes : ils doivent avoir (1) la même valeur à la ligne correspondant à l'affectation (d_i, t_i) et (2) une valeur différente à la ligne de l'affectation (d_j, t_j). De façon formelle, ces deux conditions sont décrites ci-dessous et on remarque que toutes les deux peuvent être respectées de deux façons.

1. La condition $(a_{ks_1d_it_i} = a_{ks_2d_it_i})$:
   (i) $a_{ks_1d_it_i} = 1$ et $a_{ks_2d_it_i} = 1$ ou
   (ii) $a_{ks_1d_it_i} = 0$ et $a_{ks_2d_it_i} = 0$.
2. La condition $(a_{ks_1d_jt_j} \neq a_{ks_2d_jt_j})$:
   (i) $a_{ks_1d_jt_j} = 1$ et $a_{ks_2d_jt_j} = 0$ ou
   (ii) $a_{ks_1d_jt_j} = 0$ et $a_{ks_2d_jt_j} = 1$.

Dorénavant, le terme "condition nécessaire" à l'application d'un branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$ au problème (P^u) fera référence à la condition suivante : $\exists s_1, s_2 \in S^{un}_k$ tel que $a_{ks_1d_it_i} = a_{ks_2d_it_i}$ et $a_{ks_1d_jt_j} \neq a_{ks_2d_jt_j}$.

Nous pouvons voir à la figure 3.1 une représentation matricielle de toutes les colonnes $s \in S^{un}_k$ d'une infirmière $k \in \mathcal{K}$, en plus de deux sous-matrices notées M_1

et M_2. Ces matrices correspondent respectivement aux deux branchements Ryan-Foster (k, d_1, t_2, d_2, t_1) et (k, d_1, t_1, d_1, t_2), qui respectent la "condition nécessaire" à leur application de la façon décrite ci-dessous.

- Pour (k, d_1, t_2, d_2, t_1), les deux variables s_1 et $s_2 \in S_k^{un}$ sont telles que $a_{ks_1d_1t_2} = a_{ks_2d_1t_2} = 1$, $a_{ks_1d_2t_1} = 0$ et $a_{ks_2d_2t_1} = 1$.
- Pour (k, d_1, t_1, d_1, t_2), les deux variables s_2 et $s_3 \in S_k^{un}$ sont telles que $a_{ks_2d_1t_1} = a_{ks_3d_1t_1} = 0$, $a_{ks_2d_1t_2} = 1$ et $a_{ks_3d_1t_2} = 0$.

Définition d'un branchement Ryan-Foster

Deuxièmement, la définition d'un branchement Ryan-Foster appliqué au PLNE (P^u) consiste à définir ses deux fils (P^v) et (P^w). Une telle règle de branchement est notée $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$, elle doit respecter la condition nécessaire mentionnée ci-haut et définie de la façon décrite ci-dessous ces deux nouveaux problèmes.

- **Le PLNE (P^v).**
  Ce problème est défini par l'ajout d'une nouvelle contrainte de branchement à celle(s) déjà existante(s) au problème père (P^u) : $\mathcal{C}^v_{k=} \leftarrow \mathcal{C}^u_{k=} \bigcup \{(d_i, t_i, d_j, t_j)\}$. Les trois autres ensembles de contraintes de branchement $\mathcal{C}^v_{k\neq}$, $\mathcal{C}^v_{k(one)}$ et $\mathcal{C}^v_{k(zero)}$ sont respectivement définis par $\mathcal{C}^u_{k\neq}$, $\mathcal{C}^u_{k(one)}$ et $\mathcal{C}^u_{k(zero)}$. Ainsi, pour tous les horaires $s \in S^v_k$, soit que les deux affectations (d_i, t_i) et (d_j, t_j) sont attribuées à l'infirmière k ou soit qu'aucune lui est attribuée. Formellement, la contrainte de branchement $(d_i, t_i, d_j, t_j) \in \mathcal{C}^v_{k=}$ est définie comme suit : $\forall s \in S^v_k$, $a_{ksd_it_i} = a_{ksd_jt_j}$. Ce qui peut être respecté de deux façons par un horaire $s \in S^v_k$:
  - $a_{ksd_it_i} = 1$ et $a_{ksd_jt_j} = 1$ ou
  - $a_{ksd_it_i} = 0$ et $a_{ksd_jt_j} = 0$.

- **Le PLNE (P^w).**
  De manière opposé au problème (P^v), le problème (P^w) est défini comme suit :

$\mathcal{C}^w_{k\neq} \leftarrow \mathcal{C}^u_{k\neq} \bigcup \{(d_i, t_i, d_j, t_j)\}$. Les trois autres ensembles $\mathcal{C}^w_{k=}$, $\mathcal{C}^w_{k(one)}$ et $\mathcal{C}^w_{k(zero)}$ sont respectivement définis par $\mathcal{C}^u_{k=}$, $\mathcal{C}^u_{k(one)}$ et $\mathcal{C}^u_{k(zero)}$. Ainsi, pour tous les horaires $s \in S^w_k$, une et une seule des deux affectations (d_i, t_i) et (d_j, t_j) doit être attribuée à l'infirmière k. Formellement, la contrainte $(d_i, t_i, d_j, t_j) \in \mathcal{C}^w_{k\neq}$ est définie comme suit : $\forall s \in S^w_k$, $a_{ksd_it_i} \neq a_{ksd_jt_j}$. Ce qui peut être respecté de deux façons par un horaire $s \in S^w_k$:

- $a_{ksd_it_i} = 0$ et $a_{ksd_jt_j} = 1$ ou
- $a_{ksd_it_i} = 1$ et $a_{ksd_jt_j} = 0$.

Ainsi, lorsque la méthode Pricing (voir l'algorithme 2.2) tente de générer de nouvelles colonnes pour l'infirmière k et pour le problème (P^v), les deux cas possibles pour le respect de la contrainte de branchement $(d_i, t_i, d_j, t_j) \in \mathcal{C}^v_{k=}$ sont à considérer. Le même principe s'applique au problème (P^w) avec la contrainte $(d_i, t_i, d_j, t_j) \in \mathcal{C}^w_{k\neq}$. Plus de détails concernant le concept de RFRC seront donnés à la section 3.3.

En considérant l'exemple de la figure 3.1, supposons que (k, d_1, t_2, d_2, t_1) est le branchement Ryan-Foster choisi. Les deux variables s_1 et s_3 seront donc éliminées du nouveau problème (P^v) mais pas du problème (P^w). L'inverse se produit pour l'autre variable s_2, qui sera éliminée du problème (P^w) mais pas du problème (P^v).

3.1.2 Conditions pour obtenir un branchement Ryan-Foster efficace

Maintenant, nous allons donner trois conditions importantes pour qu'un branchement Ryan-Foster soit efficace. Ces conditions s'ajoutent à la condition nécessaire à l'application d'un tel branchement (voir la section 3.1.1).

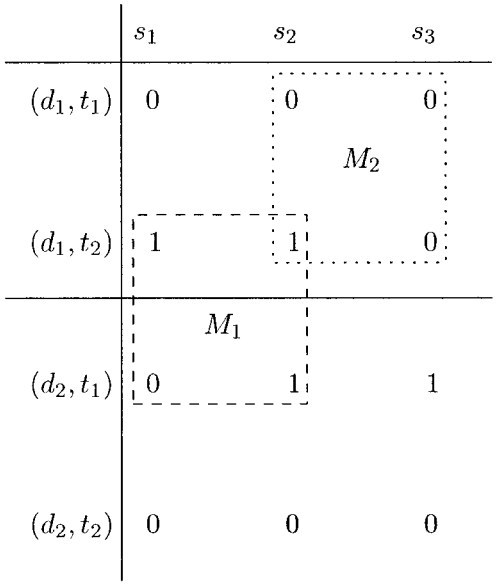


Figure 3.1: Représentation matricielle des horaires d'une infirmière $k \in \mathcal{K}$.

Considérons deux horaires s_1 et $s_2 \in S_k^{un}$ (où $s_1 \neq s_2$) qui ont été utilisés pour définir le branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$ au problème (P^u). Autrement dit, c'est grâce à l'existence de ces deux variables que la condition nécessaire à l'application de ce branchement est satisfaite.

Conditions sur l'affectation (d_i, t_i)

Premièrement, il est préférable que l'affectation (d_i, t_i) respecte les conditions suivantes : $(d_i, t_i) \notin \mathcal{O}_k$, $(d_i, t_i) \notin \mathcal{F}_k$, $(d_i, t_i) \notin \mathcal{C}^u_{k(one)}$ et $(d_i, t_i) \notin \mathcal{C}^u_{k(zero)}$. Les deux premières conditions signifient respectivement que (d_i, t_i) ne doit être ni une affectation obligatoire ni une affectation interdite pour l'infirmière k. Pareillement, la condition $(d_i, t_i) \notin \mathcal{C}^u_{k(one)}$ ($(d_i, t_i) \notin \mathcal{C}^u_{k(zero)}$) signifie que (d_i, t_i) ne doit pas être une contrainte de branchement obligeant (interdisant) à l'infirmière k de travailler à cette affectation [1].

[1] Les ensembles $\mathcal{O}_k$, $\mathcal{F}_k$, $\mathcal{C}^u_{k(one)}$ et $\mathcal{C}^u_{k(zero)}$ sont formellement définis dans la "Liste des sigles et abréviations".

Si au moins une des conditions précédentes n'est pas satisfaite par (d_i, t_i), le branchement Ryan-Foster (k, d_i, t_i, d_j, t_j) serait équivalent au branchement binaire (k, d_j, t_j). Par exemple, si $(d_i, t_i) \in \mathcal{O}_k$, nous serions dans le cas où $a_{ks_1 d_i t_i} = a_{ks_2 d_i t_i} = 1$. Par conséquent, ce branchement serait équivalent au branchement binaire (k, d_j, t_j). En effet, $a_{ksd_j t_j} = 1$ serait la contrainte ajoutée au nouveau problème (P^v) et $a_{ksd_j t_j} = 0$ serait celle ajoutée au problème (P^w). Ce cas est identique à celui où $(d_i, t_i) \in \mathcal{C}^u_{k(one)}$. Le même principe s'applique pour le cas où $(d_i, t_i) \in \mathcal{F}_k$ et celui où $(d_i, t_i) \in \mathcal{C}^u_{k(zero)}$.

Conditions sur l'affectation (d_j, t_j)

Deuxièmement, comme pour la première affectation, nous devons avoir les conditions suivantes sur la seconde : $(d_j, t_j) \notin \mathcal{O}_k$, $(d_j, t_j) \notin \mathcal{F}_k$, $(d_j, t_j) \notin \mathcal{C}^u_{k(one)}$ et $(d_j, t_j) \notin \mathcal{C}^u_{k(zero)}$. Par contre, contrairement à l'affectation (d_i, t_i), ces conditions sont obligatoirement satisfaites par (d_j, t_j). Ceci découle directement de la condition nécessaire à l'application du branchement Ryan-Foster (k, d_i, t_i, d_j, t_j). En d'autres termes, grâce à l'existence des deux variables s_1 et $s_2 \in S^{un}_k$, qui sont de valeur différente à la ligne de l'affectation (d_j, t_j), cette affectation ne peut être dans aucun des 4 ensembles mentionnés ci-haut.

Condition sur la paire de jour (d_i, d_j)

Troisièmement, il est préférable que les jours d_i et d_j soient différents. En effet, le branchement Ryan-Foster (k, d_i, t_i, d_j, t_j) ne serait pas efficace si $d_i = d_j$. Ceci s'explique par l'existence d'une contrainte intrinsèque à toute infirmière $k \in \mathcal{K}$ qui stipule qu'il n'est pas possible de leur affecter plus d'un quart de travail pour chaque jour $d \in \mathcal{D}$ de l'horizon.

Par exemple, supposons que le branchement Ryan-Foster $(k, d', t_i, d', t_j) \in \mathcal{K}_{\mathcal{D}^2_{T_k}}$ soit appliqué au problème courant (P^u). Ainsi, il n'y aura que la possibilité suivante

pour le respect de la contrainte de branchement $(d', t_i, d', t_j) \in \mathcal{C}^v_{k=}$ par un horaire $s \in S^v_k$ au nouveau problème (P^v) : $a_{ksd't_i} = a_{ksd't_j} = 0$. En effet, le cas où $a_{ksd't_i} = a_{ksd't_j} = 1$ n'est pas possible par cause de la contrainte intrinsèque définie ci-haut.

Finalement, il est important de faire la distinction entre la technique utilisée pour obtenir une séparation efficace (voir la section 2.4.4) et les conditions décrites ci-dessus. Ces dernières sont propres au branchement Ryan-Foster. Par contre, la précédente technique a pour objectif d'obtenir une séparation disjointe par l'application d'un branchement Ryan-Foster ou d'un branchement binaire.

3.2 Règle de branchement binaire

Cette section est entièrement consacrée au branchement binaire. Nous commençons par décrire la condition nécessaire à l'application d'un tel branchement en plus de donner sa définition. Deuxièmement, quelques observations importantes concernant ce branchement seront données. Finalement, nous expliquons comment il est parfois possible d'ajouter de nouvelles contraintes, en plus de celle résultant de l'application du branchement binaire, à un et un seul des PLNE fils du problème courant.

3.2.1 Condition nécessaire et définition

Contrairement au branchement Ryan-Foster, la condition nécessaire à l'application d'un branchement binaire, qui est noté $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{T_k}}$, est très simple. Un tel branchement est possible si et seulement si $\exists s_1, s_2 \in S^{un}_k$ $(s_1 \neq s_2)$ qui ont une valeur différente à la ligne correspondant à l'affectation (d, t). Formellement, la condition nécessaire à l'application d'un branchement binaire $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{T_k}}$ est définie comme suit : $\exists s_1, s_2 \in S^{un}_k$ tel que $a_{ks_1dt} \neq a_{ks_2dt}$.

Ensuite, comme la définition d'un branchement Ryan-Foster, celle d'un branchement binaire appliqué au problème (P^u) consiste à définir ses deux fils (P^v) et (P^w). Une telle règle de branchement est notée $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{T_k}}$, elle doit respecter la condition nécessaire mentionnée ci-haut et définie de la façon décrite ci-dessous ces deux nouveaux problèmes.

- **Le problème (P^v).**
  Ce problème est défini par l'ajout d'une nouvelle contrainte de branchement à celle(s) déjà existante(s) au problème père (P^u) : $\mathcal{C}^v_{k(one)} \leftarrow \mathcal{C}^u_{k(one)} \bigcup \{(d,t)\}$. Les trois autres ensembles $\mathcal{C}^v_{k=}$, $\mathcal{C}^v_{k\neq}$ et $\mathcal{C}^v_{k(zero)}$ sont respectivement définis par $\mathcal{C}^u_{k=}$, $\mathcal{C}^u_{k\neq}$ et $\mathcal{C}^u_{k(zero)}$. Ainsi, pour tous les horaires $s \in S^v_k$, l'affectation (d,t) doit obligatoirement être attribuée à l'infirmière de branchement k. Formellement, la contrainte de branchement $(d,t) \in \mathcal{C}^v_{k(one)}$ est définie comme suit : $\forall s \in S^v_k,\ a_{ksdt} = 1$.

- **Le problème (P^w).**
  De manière opposée au problème (P^v), le problème (P^w) est défini de cette façon : $\mathcal{C}^w_{k(zero)} \leftarrow \mathcal{C}^u_{k(zero)} \bigcup \{(d,t)\}$, $\mathcal{C}^w_{k=} \leftarrow \mathcal{C}^u_{k=}$, $\mathcal{C}^w_{k\neq} \leftarrow \mathcal{C}^u_{k\neq}$ et $\mathcal{C}^w_{k(one)} \leftarrow \mathcal{C}^u_{k(one)}$. Formellement, la contrainte de branchement $(d,t) \in \mathcal{C}^w_{k(zero)}$ est définie comme suit : $\forall s \in S^w_k,\ a_{ksdt} = 0$.

3.2.2 Observations importantes

Bien que le branchement binaire est plus simple que le branchement Ryan-Foster, il est important de souligner quelques observations concernant cette première méthode de séparation. Considérons deux horaires s_1 et $s_2 \in S^{un}_k$ (où $s_1 \neq s_2$) qui ont été utilisés pour définir le branchement binaire $(k, d_l, t_l) \in \mathcal{K}_{\mathcal{D}_{T_k}}$ au problème

(P^u). Autrement dit, c'est grâce à l'existence de ces deux variables que la condition nécessaire à l'application de ce branchement est satisfaite.

Premièrement, pour que le branchement binaire (k, d_l, t_l) soit efficace, l'affectation (d_l, t_l) doit respecter les conditions suivantes : $(d_l, t_l) \notin \mathcal{O}_k$, $(d_l, t_l) \notin \mathcal{F}_k$, $(d_l, t_l) \notin \mathcal{C}^u_{k(one)}$ et $(d_l, t_l) \notin \mathcal{C}^u_{k(zero)}$. Par contre, grâce à la condition nécessaire mentionnée ci-haut, ces 4 conditions sont obligatoirement satisfaites. En effet, étant donné que les deux horaires s_1 et s_2 sont tels que $a_{ks_1d_lt_l} \neq a_{ks_2d_lt_l}$, l'affectation (d_l, t_l) ne peut être dans aucun des ensembles mentionnés ci-haut.

Ensuite, nous sommes assurés que ce branchement n'a jamais été appliqué à un problème antérieur au problème (P^u). Ceci découle directement de la précédente observation et il est donc inutile de vérifier que ce branchement a déjà été appliqué.

Troisièmement, il est possible qu'un branchement Ryan-Foster $(k, d_i, t_i, d_j, t_j) \in \mathcal{K}_{\mathcal{D}^2_{\mathcal{T}_k}}$, qui a été appliqué à l'infirmière de branchement k et à un problème antérieur au problème courant (P^u), ai une et une seule de ses deux affectations (d_i, t_i) ou (d_j, t_j) en commun avec celle du branchement binaire (k, d_l, t_l). Formellement, il est possible que $\exists (d_i, t_i, d_j, t_j) \in \mathcal{C}^u_k$ tel que $\big((d_l, t_l) = (d_i, t_i) \bigvee (d_l, t_l) = (d_j, t_j)\big)$ [2].

Lorsque ceci survient, l'application du branchement Ryan-Foster (k, d_i, t_i, d_j, t_j) combinée avec celle du nouveau branchement binaire (k, d_l, t_l) éliminera un et un seul des deux cas possibles pour le respect de ce premier branchement, voir la section 3.1.1. Lors de la description du concept RFRC à la section 3.3, nous donnerons plus de détails à ce sujet.

[2] Rappellons que $\mathcal{C}^u_k \leftarrow \mathcal{C}^u_{k=} \bigcup \mathcal{C}^u_{k\neq}$.

3.2.3 Nouvelles contraintes obtenues par déductions logiques

Maintenant, nous allons expliquer comment il peut être possible d'obtenir de nouvelles contraintes lors de l'application d'un branchement binaire. Rappelons qu'un tel branchement est noté $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{\mathcal{T}_k}}$, est appliqué au problème courant (P^u) et consiste à obliger à l'infirmière de branchement k de travailler sur l'affectation (d, t) au problème (P^v) et de lui interdire au problème (P^w). Ainsi, par de simples déductions logiques, il est parfois possible d'interdire à d'autres infirmières plusieurs affectations dans la définition du nouveau problème (P^v).

3.2.3.1 Explication du principe

Il est important de préciser que les déductions logiques décrites ci-dessous ne modifieront pas le problème (P^w), qui est défini selon la définition du branchement binaire (k, d, t). De plus, pour faciliter la compréhension de cette section, nous introduisons une nouvelle notation afin de représenter l'ensemble des contraintes de quotas définies sur tout l'horizon. Cet ensemble est noté par $\mathcal{Q} = \{(Q_L) : L \in \mathcal{L}_{dp}, (d, p) \in \mathcal{D}_{\mathcal{P}}\}$.

Premièrement, pour chaque contrainte de quotas $(Q_L) \in \mathcal{Q}$, où $L \in \mathcal{L}_{dp}$ et $(d, p) \in \mathcal{D}_{\mathcal{P}}$, nous définissons l'ensemble $\mathcal{K}^u_{(Q_L)} \subseteq \mathcal{K}$. Chaque infirmière $k \in \mathcal{K}^u_{(Q_L)}$ a une compétence et une contrainte de branchement binaire qui couvrent la contrainte (Q_L) : $b_{kL} = 1$ et $\exists (d, t) \in \mathcal{C}^u_{k(one)}$ tel que $\alpha_{tp} = 1$. Remarquons que la contrainte de branchement (d, t) et la paire (d, p) (voir ci-dessus) ont le jour $d \in \mathcal{D}$ en commun. Ainsi, chaque infirmière $k \in \mathcal{K}^u_{(Q_L)}$ aura un apport dans la contrainte de quotas (Q_L) dans tous les horaires généraux qui sont réalisables au problème (P^u). Formellement, pour une contrainte de quotas $(Q_L) \in \mathcal{Q}$, où $L \in \mathcal{L}_{dp}$ et $(d, p) \in \mathcal{D}_{\mathcal{P}}$, et pour le problème courant (P^u) de l'arbre de recherche, cet ensemble est défini comme suit :

$$\mathcal{K}^u_{(Q_L)} = \{k \in \mathcal{K} : b_{kL} = 1,\ \exists (d, t) \in \mathcal{C}^u_{k(one)} \text{ tel que } \alpha_{tp} = 1\}.$$

Ensuite, supposons que la règle de branchement binaire $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{\mathcal{T}_k}}$ est appliquée au problème (P^u). Alors, au problème (P^v), chaque contrainte de quotas $(Q_L) \in \mathcal{Q}$ qui est couverte par cette règle ($b_{kL} = 1$ et $\alpha_{tp} = 1$) sera telle que $|\mathcal{K}^v_{(Q_L)}| = |\mathcal{K}^u_{(Q_L)}| + 1$. Les autres contraintes qui ne sont pas couvertes par cette règle seront telles que $|\mathcal{K}^v_{(Q_L)}| = |\mathcal{K}^u_{(Q_L)}|$. Tandis qu'au problème (P^w), ce sont toutes les contraintes $(Q_L) \in \mathcal{Q}$ qui seront telles que $|\mathcal{K}^w_{(Q_L)}| = |\mathcal{K}^u_{(Q_L)}|$.

Troisièmement, notons par $\mathcal{Q}^v \subseteq \mathcal{Q}$, l'ensemble des contraintes de quotas qui deviennent saturées au problème (P^v), grâce à l'application du branchement binaire $(k, d, t) \in \mathcal{K}_{\mathcal{D}_{\mathcal{T}_k}}$ au problème (P^u), ou qui l'étaient déjà à un problème antérieur au PLNE (P^v). Cet ensemble est formellement défini ci-dessous :

$$\mathcal{Q}^v = \{(Q_L) \in \mathcal{Q} : |\mathcal{K}^v_{(Q_L)}| = q_L + q^+_L\}.$$

Il est alors possible d'ajouter de nouvelles contraintes dans la définition du problème (P^v), en plus de la contrainte de branchement qui est obtenue par la définition du branchement binaire (k, d, t) et qui est appliquée sur l'infirmière k (voir la section 3.2.1). Pour chaque contrainte de quotas $(Q_L) \in \mathcal{Q}^v$, nous devons interdire à chaque infirmière $k' \in \mathcal{K} \setminus \mathcal{K}^v_{(Q_L)}$, où $b_{k'L} = 1$, toutes les affectations qui couvrent cette contrainte de quotas saturée.

Enfin, nous pouvons voir ci-dessous les nouvelles contraintes ajoutées au problème (P^v) et celle ajoutée au problème (P^w). Nous remarquons qu'il y a peu de différence par rapport à la définition du branchement binaire (voir la section 3.2.1). En effet, les seules nouvelles contraintes sont les contraintes d'interdiction qui ont été ajoutées au problème (P^v) et qui ont été obtenues grâce à l'application de déductions logiques sur le branchement binaire (k, d, t).

- **Considérons le problème** (P^v)**.**
  - $\mathcal{C}^v_{k(one)} \leftarrow \mathcal{C}^u_{k(one)} \bigcup \{(d,t)\}$.
  - Pour toute infirmière $k' \in \mathcal{K} \setminus \{k\}$, l'ensemble $\mathcal{C}^v_{k'(zero)}$ est défini par l'union d'ensemble décrite ci-dessous.

$$\mathcal{C}^u_{k'(zero)} \bigcup \left\{ (d',t') \in \mathcal{D}_{\mathcal{T}_{k'}} \;\middle|\; \begin{array}{l} \exists (Q_L) \in \mathcal{Q}^v \text{ où } L \in \mathcal{L}_{d'p'}, (d',p') \in \mathcal{D}_{\mathcal{P}} \\ \text{et t.q } k' \notin \mathcal{K}^v_{(Q_L)},\ \alpha_{t'p'} = 1 \text{ et } b_{k'L} = 1. \end{array} \right\}$$

- **Considérons le problème** (P^w)**.**
  - $\mathcal{C}^w_{k(one)} \leftarrow \mathcal{C}^u_{k(one)}$.
  - $\mathcal{C}^w_{k(zero)} \leftarrow \mathcal{C}^u_{k(zero)} \bigcup \{(d,t)\}$.

Finalement, nous verrons au chapitre 5 quelques résultats de calcul qui montrent clairement que ces déductions logiques ont augmenté l'efficacité de l'algorithme de branch and price.

3.2.3.2 Déductions logiques et génération de colonnes

Maintenant que le concept des déductions logiques est présenté, nous allons montrer où et comment les nouvelles contraintes ainsi obtenues peuvent avoir un apport positif dans le branch and price. En outre, nous allons expliquer pourquoi ces nouvelles contraintes n'ont un apport que dans la première phase de la méthode de génération de colonnes et non pas dans la deuxième. Rappelons que ces deux phases sont respectivement notées GC_{ϕ_1} et GC_{ϕ_2}.

Afin de bien comprendre notre raisonnement, nous commençons par décrire le contexte qui vas nous servir de base à nos explications. Rappelons que nous ne considérons que le problème (P^v), le problème (P^w) n'étant pas considéré étant donné

que sa définition demeure inchangée après l'application de déductions logiques (voir la section précédente).

Mise en contexte

Premièrement, supposons que les déductions logiques décrites ci-dessus ne sont pas appliquées à aucun moment de l'algorithme du branch and price. Soit une contrainte de quotas $(Q_L) \in \mathcal{Q}^v$, où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_\mathcal{P}$, et une infirmière $k \in \mathcal{K} \setminus \mathcal{K}^v_{(Q_L)}$ tel que $b_{kL} = 1$. De plus, supposons que $\exists\ s \in S^v_k$ tel que $a_{ksdt} = 1$ et tel que $\alpha_{tp} = 1$, cet horaire couvre donc la contrainte de quota (Q_L) alors que l'infirmière k ne possède pas de contraintes de branchement couvrant cette dernière ($k \notin \mathcal{K}^v_{(Q_L)}$).

Malgré tout, étant donnée que la contrainte (Q_L) est saturée et grâce à la modélisation du PLNE (P^v) (voir la figure 2.7), la variable associée à l'horaire $s \in S^v_k$ sera telle que $x_{ks} = 0$ dans toutes les solutions réalisables (horaires généraux) du PLNE (P^v). Ceci s'applique aussi à toutes les solutions réalisables de la relaxation continue (R^v).

Troisièmement, pour une infirmière $k \in \mathcal{K}$ et le noeud v de l'arbre de recherche, définissons le sous-ensembles d'horaires $\widetilde{S}^v_k \subseteq S^v_k$. Selon les contraintes de quotas qui sont saturées au PLNE (P^v), cet ensemble est composé d'horaires qui sont réalisables pour l'infirmière k au problème (P^v), mais dont les variables seront de valeur nulle dans toutes les solutions réalisables de ce PLNE.

$$\widetilde{S}^v_k = \left\{ s \in S^v_k \;\middle|\; \begin{array}{l} \exists (Q_L) \in \mathcal{Q}^v \text{ où } L \in \mathcal{L}_{dp},\ (d,p) \in \mathcal{D}_\mathcal{P},\ b_{kL} = 1,\ k \notin \mathcal{K}^v_{(Q_L)}, \\ \text{ et t.q. } \exists\ t \in \mathcal{T}_k \text{ où } \alpha_{tp} = 1 \text{ et } a_{ksdt} = 1. \end{array} \right\}$$

Autrement dit, pour chaque horaire $s \in \widetilde{S}^v_k$, il doit exister au moins une contrainte de quotas saturée, notée $(Q_L) \in \mathcal{Q}^v$, tel que

- l'horaire a une affectation portant sur le jour et la période de (Q_L),
- l'infirmière n'a pas de contrainte de branchement couvrant (Q_L) et
- a la compétence requise pour apporter sa contribution à (Q_L).

Maintenant, nous allons utilisé la notation décrite ci-dessus afin d'expliquer pourquoi les déductions logiques n'ont pas d'impact théorique sur la seconde phase de la méthode de génération de colonnes.

Déductions logiques et GC_{ϕ_2}

Rappelons que cette phase consiste à trouver une solution de valeur optimale à la relaxation continue (R^v). Ceci se fait par l'application de l'algorithme itératif GC, où à chaque itération i le problème restreint (R^{vi}) est résolu et la solution obtenue x_v^i est optimale à ce problème, en plus d'être une solution réalisable au problème (R^v) [3].

Premièrement, soit un horaire $s \in S_k^v \bigcap \widetilde{S}_k^v$ qui est déjà inclus dans le premier problème restreint (R^{v1}) de l'itération 1 : $s \in S_k^{v1}$. Par conséquent, pour chacune des solutions optimales x_v^i d'une itération i, la valeur de la variable associée à cet horaire est telle que $x_{ks} = 0$. Ceci revient à dire que cet horaire a été généré soit lors de la phase GC_{ϕ_1} du noeud v ou soit à un noeud antérieur à celui-ci.

Deuxièmement, soit un horaire $s \in S_k^v \bigcap \widetilde{S}_k^v$ qui n'est pas inclus dans le premier problème restreint (R^{v1}) de l'itération 1 : $s \notin S_k^{v1}$. Alors, pour toutes les autres itérations i, il est impossible pour la méthode Pricing de générer cet horaire, car la variable x_{ks} serait de valeur nulle dans toutes les solutions x_v^i. Ceci est dû au fait que $PA_{\phi_2}(\hat{G}_{ki}^{cv}) \subseteq \{s \in S_k^{cv} : s \notin S_k^{vi} \text{ et } c_{ks} < 0\}$ et que $c_{ks} \geq 0$. Ici, on suppose qu'il n'y a pas de dégénérescence.

[3] Les algorithmes GC et GC_{ϕ_2} sont respectivement décrits aux sections 2.3.1 et 2.3.5.

Enfin, on peut donc affirmer que l'application de déductions logiques n'apportent aucune contribution théorique dans la phase GC_{ϕ_2}, tant au niveau de l'algorithme GC qu'à celui de la méthode Pricing. Toutefois, il peut arriver que ces déductions logiques aient un effet positif sur le processus interne de l'algorithme du Simplexe (voir ILOG [21]) ou sur l'implémentation des algorithmes GC et PA_{ϕ_2} au niveau de la précision de calcul.

Déductions logiques et GC_{ϕ_1}

Mentionnons que cette phase est appliquée au problème (R^{va}) afin de vérifier si le problème (R^v) est réalisable et de définir le premier problème restreint (R^{v1}), qui est nécessaire à la résolution de (R^v). De plus, rappelons que la solution x_u^n du problème (R^u) n'est plus réalisable pour (R^v), dans le cas où une séparation en base a été appliquée. C'est d'ailleurs sur ce point que l'application de déductions logiques apporte une contribution significative à la phase GC_{ϕ_1}.

Premièrement, considérons le cas où les déductions logiques ne sont pas utilisées. Soit un horaire $s \in S_k^v \bigcap \widetilde{S}_k^v$ d'une infirmière $k \in \mathcal{K}$ et l'itération courante i de GC. Rappelons que lors de l'application de GC pour résoudre (R^{ua}), les différentes solutions x_{ua}^i de (R^{ua}) ne sont pas réalisables à (R^u), à l'exception de la dernière solution x_{ua}^n de valeur optimale $z_{ua}^* = 0$. Par conséquent, si $s \in S_k^{vi}$, il est possible que le coût réduit de cet horaire soit de valeur négative $(c_{ks} < 0)$ et que la variable x_{ks} "rentre" en base. Par le même raisonnement et si $s \notin S_k^{vi}$, cet horaire peut donc être généré par la méthode Pricing.

Par contre, si les déductions logiques sont utilisées, le même horaire ne pourrait être ni en base ni généré par le Pricing. Ceci s'explique par la simple raison que cet horaire ne serait pas valide au PLNE (P^v) pour l'infirmière k : $s \notin S_k^v$.

Finalement, on peut conclure que l'application des déductions logiques apportent une contribution théorique à la phase GC_{ϕ_1}. Nous verrons des résultats de calcul pour appuyer ceci au chapitre 5.

3.3 Description du concept RFRC

Dans la méthode Pricing, lorsqu'on tente de générer de nouvelles colonnes CR^- pour une infirmière $k \in \mathcal{K}$, on doit traiter les différentes façons de respecter simultanément toutes ses contraintes de branchement. Ceci a déjà été introduit à la section 2.2 sous le terme RFRC [4]. En effet, comme nous l'avons vu à la section 3.1, chacune des contraintes de branchement Ryan-Foster peut être respectée de deux façons. Par conséquent, en considérant l'ensemble de toutes les contraintes Ryan-Foster, défini par $\mathcal{C}_k^u \leftarrow \mathcal{C}_{k=}^u \bigcup \mathcal{C}_{k\neq}^u$ pour un noeud u de l'arbre de recherche et pour une infirmière $k \in \mathcal{K}$, il y a un total de $2^{|\mathcal{C}_k^u|}$ combinaisons. Par contre, parmi toutes ces combinaisons, il faut considérer que celles qui sont possibles, soit celles qui ne contiennent pas de contradiction et qui peuvent donc satisfaire ces $|\mathcal{C}_k^u|$ contraintes simultanément. On note par $\mathcal{I}_k^u \subseteq \{0, 1, \ldots, 2^{|\mathcal{C}_k^u|} - 1\}$, l'ensemble des indices des combinaisons possibles pour l'infirmière $k \in \mathcal{K}$ au noeud u.

L'objectif de cette section est de développer l'algorithme nommé *calculer* $\mathcal{I}_k^u$, dont les tâches ont déjà été décrites à la section 2.2. À l'aide d'exemples, nous commençons par expliquer comment une combinaison peut contenir des contradictions et ne serait donc pas possible. Ensuite, nous présentons la bijection utilisée par l'algorithme pour représenter de façon unique une combinaison d'indice $c \in \{0, 1, \ldots, 2^{|\mathcal{C}_k^u|} - 1\}$. Finalement, l'algorithme est expliqué et donné dans ses moindres détails.

[4] Ryan-Foster Respect Combination.

3.3.1 Présentation d'exemples

Notons par (P^u) le PLNE courant de notre algorithme de branch and price. Considérons une infirmière $k \in \mathcal{K}$ et ses contraintes de branchement Ryan-Foster qui sont définies par les deux ensembles suivants : $\mathcal{C}^u_{k=} = \{(d_1, t_1, d_2, t_2), (d_3, t_3, d_4, t_4)\}$ et $\mathcal{C}^u_{k\neq} = \{(d_5, t_5, d_6, t_6)\}$, où (d_1, t_1), (d_2, t_2), ..., $(d_6, t_6) \in \mathcal{D}_{\mathcal{T}_k}$. Remarquons que $\mathcal{C}^u_k = \{(d_1, t_1, d_2, t_2), (d_3, t_3, d_4, t_4), (d_5, t_5, d_6, t_6)\}$.

D'après le nombre de contraintes Ryan-Foster associées à l'infirmière k, qui est de $|\mathcal{C}^u_k| = 3$, il y a un maximum de $2^{|\mathcal{C}^u_k|} = 8$ combinaisons possibles définissant la nature (affectation interdite ou obligatoire) des 6 affectations énumérées ci-dessus. Nous pouvons voir au tableau 3.1 ces 8 combinaisons, qui sont numérotées par l'indice $c \in \{0, 1, \ldots, 7\}$. La nature de chaque affectation $(d_i, t_i) \in \mathcal{D}_{\mathcal{T}_k}$ est définie à l'aide de la variable du problème auxiliaire qui lui est associée, soit la variable $a_{ksd_it_i}$.

Tableau 3.1: Définition des 8 combinaisons selon $\mathcal{C}^u_k$.

c	$a_{ksd_5t_5}$	$a_{ksd_6t_6}$	$a_{ksd_3t_3}$	$a_{ksd_4t_4}$	$a_{ksd_1t_1}$	$a_{ksd_2t_2}$
0	0	1	0	0	0	0
1	0	1	0	0	1	1
2	0	1	1	1	0	0
3	0	1	1	1	1	1
4	1	0	0	0	0	0
5	1	0	0	0	1	1
6	1	0	1	1	0	0
7	1	0	1	1	1	1

Maintenant, dépendant des contraintes binaires définies par les ensembles $\mathcal{C}^u_{k(one)}$ et $\mathcal{C}^u_{k(zero)}$ et des égalités qui peuvent exister entre les 6 affectations énumérées ci-dessus, il est possible que certaines des combinaisons décrites au tableau 3.1 contiennent une

ou plusieurs contradictions. Nous expliquons les principaux cas à l'aide de 3 exemples. Afin de simplifier ces exemples, nous supposons que les deux ensembles $\mathcal{F}_k$ et $\mathcal{O}_k$ sont tous les deux vides.

Exemple 1

Considérons le cas où les 6 affectations (d_1, t_1), (d_2, t_2), ..., (d_6, t_6) sont toutes différentes l'une de l'autre. Supposons aussi que les deux ensembles de contraintes binaires $\mathcal{C}^u_{k(one)}$ et $\mathcal{C}^u_{k(zero)}$, tous deux reliés à l'infirmière k, sont vides. Dans ce cas, les 8 combinaisons énumérées au tableau 3.1 sont possibles et nous avons $\mathcal{I}^u_k = \{0, 1, \ldots, 7\}$.

Exemple 2

Reprenons les mêmes conditions qu'à l'exemple 1, à l'exception près que les deux affectations (d_2, t_2) et (d_5, t_5) sont identiques. Dans ce cas, 4 des 8 combinaisons décrites au tableau 3.1 contiennent une contradiction et ne sont donc pas possibles. Les combinaisons possibles sont celles d'indice $c \in \mathcal{I}^u_k$, où $\mathcal{I}^u_k = \{0, 2, 5, 7\}$.

Exemple 3

Considérons les mêmes conditions qu'à l'exemple 1, à l'exception près que l'affectation (d_5, t_5) est obligatoire à l'infirmière k par cause de la contrainte de branchement binaire $(d_5, t_5) \in \mathcal{C}^u_{k(one)}$. Formellement, nous avons $\mathcal{C}^u_{k(one)} = \{(d_5, t_5)\}$. Dans ce cas, 4 des 8 combinaisons décrites au tableau 3.1 ne sont pas possibles. L'ensemble des combinaisons possibles est $\mathcal{I}^u_k = \{4, 5, 6, 7\}$.

3.3.2 Représentation binaire d'une combinaison

Maintenant, nous allons présenter la bijection qui sera utilisée par notre algorithme pour définir une combinaison d'indice $c \in \{0, 1, \ldots, 2^{|\mathcal{C}_k^u|} - 1\}$. Cette bijection n'est rien de plus que la représentation binaire de cet indice sur un vecteur de taille égale à $|\mathcal{C}_k^u|$, soit le nombre de contraintes Ryan-Foster qui sont propres à l'infirmière k.

Considérons les données décrites à la section précédente. Nous avons $|\mathcal{C}_k^u| = 3$ contraintes Ryan-Foster qui sont propres à l'infirmière k. Par conséquent, chacune des 8 combinaisons d'indice $c \in \{0, 1, \ldots, 7\}$ seront représentées sur un vecteur binaire de taille 3. Par exemple, considérons la combinaison d'indice $c = 4$ qui est représentée au tableau 3.2. Dans ce tableau, le vecteur de bits se lit de droite à gauche. Chacune des 3 constantes $(d_i, t_i, d_j, t_j)_4 \in \{0, 1\}$ représente lequel des deux cas possibles est considéré par cette combinaison pour le respect de la contrainte Ryan-Foster $(d_i, t_i, d_j, t_j) \in \mathcal{C}_k^u$.

Tableau 3.2: Représentation binaire de la combinaison $c = 4$.

contrainte	$(d_5, t_5, d_6, t_6)_4$	$(d_3, t_3, d_4, t_4)_4$	$(d_1, t_1, d_2, t_2)_4$
bit	1	0	0

Soit une combinaison d'indice $c \in \{0, 1, \ldots, 2^{|\mathcal{C}_k^u|} - 1\}$. Cette combinaison est définie par les constantes décrites ci-dessous.

1. Considérons une contrainte $(d_i, t_i, d_j, t_j) \in \mathcal{C}_{k=}^u$, nous définissons alors la constante suivante :

$$(d_i, t_i, d_j, t_j)_c = \begin{cases} 0 & \text{représente le cas où } a_{ksd_it_i} = 0 \text{ et } a_{ksd_jt_j} = 0, \\ 1 & \text{représente le cas où } a_{ksd_it_i} = 1 \text{ et } a_{ksd_jt_j} = 1. \end{cases}$$

2. Considérons une contrainte $(d_i, t_i, d_j, t_j) \in \mathcal{C}^u_{k\neq}$, nous définissons alors la constante suivante :

$$(d_i, t_i, d_j, t_j)_c = \begin{cases} 0 & \text{représente le cas où } a_{ksd_it_i} = 0 \text{ et } a_{ksd_jt_j} = 1, \\ 1 & \text{représente le cas où } a_{ksd_it_i} = 1 \text{ et } a_{ksd_jt_j} = 0. \end{cases}$$

La façon d'ordonner les différentes constantes $(d_i, t_i, d_j, t_j)_c$, où $(d_i, t_i, d_j, t_j) \in \mathcal{C}^u_k$, sur le vecteur de bits est déterminée de façon arbitraire.

Enfin, maintenant que la bijection entre les indices $c \in \{0, 1, \ldots, 2^{|\mathcal{C}^u_k|} - 1\}$ et les $2^{|\mathcal{C}^u_k|}$ combinaisons est définie, il est maintenant possible d'expliquer l'algorithme calculer $\mathcal{I}^u_k$.

3.3.3 Description de l'algorithme

Pour le PLNE courant (P^u) et une infirmière $k \in \mathcal{K}$, l'algorithme que nous présentons a deux tâches : (1) définir l'ensemble $\mathcal{I}^u_k$ et (2) pour chaque $c \in \mathcal{I}^u_k$, calculer les ensembles qui définissent cette combinaison. Ces deux ensembles sont $\mathcal{O}^{cu}_k$ et $\mathcal{F}^{cu}_k$, qui représentent respectivement l'ensemble des affectations obligatoires et l'ensemble des affectations interdites définissant la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}^u_k}$. Ainsi, pour atteindre ces deux tâches, l'algorithme utilise la bijection qui a précédemment été définie. Maintenant, expliquons le fonctionnement de l'algorithme lorsqu'il considère une combinaison d'indice $c \in \{0, 1, \ldots, 2^{|\mathcal{C}^u_k|} - 1\}$.

Premièrement, l'algorithme initialise respectivement les ensembles $\mathcal{O}^{cu}_k$ et $\mathcal{F}^{cu}_k$ à partir des ensembles de contraintes binaires $\mathcal{C}^u_{k(one)}$ et $\mathcal{C}^u_{k(zero)}$. En effet, il faut aussi considérer les contraintes de branchement binaire dans la définition de chaque combinaison.

Ensuite, pour chaque contrainte Ryan-Foster $(d_i, t_i, d_j, t_j) \in \mathcal{C}^u_{k=}$, on détermine à l'aide de la constante $(d_i, t_i, d_j, t_j)_c \in \{0, 1\}$ lequel des deux cas est considéré par cette combinaison pour le respect de cette contrainte. Selon la valeur de cette constante, on ajoute les deux affectations (d_i, t_i) et (d_j, t_j) soit à l'ensemble $\mathcal{O}^{cu}_k$ ou soit à l'ensemble $\mathcal{F}^{cu}_k$.

Toutefois, il est important de s'assurer qu'il n'y a pas de contradictions lors de ces ajouts. Par exemple, supposons que $(d_i, t_i, d_j, t_j)_c = 1$. Dans ce cas, les deux affectations (d_i, t_i) et (d_j, t_j) seront considérées comme des affectations obligatoires et seront donc ajoutées à l'ensemble $\mathcal{O}^{cu}_k$. Par contre, il est possible que l'une de ces deux affectations (ou les deux) ai déjà été définie comme étant une affectation interdite, que $(d_i, t_i) \in \mathcal{F}^{cu}_k$ et/ou $(d_j, t_j) \in \mathcal{F}^{cu}_k$. Lorsque ceci survient, une contradiction apparaît dans la définition de la combinaison d'indice c, qui n'est donc pas possible : $c \notin \mathcal{I}^u_k$.

Un principe similaire s'applique pour les trois autres cas suivants :

- $(d_i, t_i, d_j, t_j)_c = 0$ et $(d_i, t_i, d_j, t_j) \in \mathcal{C}^u_{k=}$,
- $(d_i, t_i, d_j, t_j)_c = 1$ et $(d_i, t_i, d_j, t_j) \in \mathcal{C}^u_{k\neq}$,
- $(d_i, t_i, d_j, t_j)_c = 0$ et $(d_i, t_i, d_j, t_j) \in \mathcal{C}^u_{k\neq}$.

Troisièmement, dès qu'une combinaison d'indice $c \in \{0, 1, \ldots, 2^{|\mathcal{C}^u_k|} - 1\}$ présente une contradiction, elle n'est pas ajoutée à l'ensemble $\mathcal{I}^u_k$ et l'algorithme continu immédiatement avec la prochaine combinaison. Dans le cas contraire, lorsque toutes les contraintes de branchement Ryan-Foster $(d_i, t_i, d_j, t_j) \in \mathcal{C}^u_k$ ont été traîtées et que la nature des deux affectations (d_i, t_i) et (d_j, t_j) a été définie sans l'apparition de contradiction, la combinaison considérée est réalisable et est telle que $c \in \mathcal{I}^u_k$.

Par exemple, considérons les données décrites à la section 3.3.1 et l'exemple 2. Lorsque l'algorithme tentera de définir la combinaison d'indice $c = 1$, la contradiction

où $(d_2, t_2) \in \mathcal{O}_k^{cu}$ et $(d_5, t_5) \in \mathcal{F}_k^{cu}$ apparaîtra, étant donné que ces deux affectations sont identiques.

Finalement, lorsqu'une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ est considérée par la méthode Pricing, les deux ensembles $\mathcal{O}_k^{cu}$ et $\mathcal{F}_k^{cu}$ sont utilisés pour appliquer le premier prétraitement nécessaire avant d'appliquer PA_{ϕ_2} à cette combinaison. C'est ainsi que PA_{ϕ_2} tente de trouver des horaires CR^- qui respectent les contraintes définies selon la combinaison (k, c) (voir les sections 2.3.2.2 et 4.1.3.1).

Algorithme 3.1 Méthode *calculer* $\mathcal{I}_k^u$ pour le problème (P^u) et pour une infirmière $k \in \mathcal{K}$.

pour tout $c \in \{0, 1, \ldots, 2^{|\mathcal{C}_k^u|} - 1\}$ **effectuer**
  $\mathcal{O}_k^{cu} \leftarrow \mathcal{C}_{k(one)}^u \bigcup \mathcal{O}_k$
  $\mathcal{F}_k^{cu} \leftarrow \mathcal{C}_{k(zero)}^u \bigcup \mathcal{F}_k$

  pour tout $(d_i, t_i, d_j, t_j) \in \mathcal{C}_{k=}^u$ **effectuer**
    si $\big((d_i, t_i, d_j, t_j)_c = 0\big)$ **alors**
      si $\big((d_i, t_i) \in \mathcal{O}_k^{cu} \ \vee \ (d_j, t_j) \in \mathcal{O}_k^{cu}\big)$ **alors**
        Aller à "Continuer"
      sinon
        $\mathcal{F}_k^{cu} \leftarrow \mathcal{F}_k^{cu} \bigcup \{(d_i, t_i), (d_j, t_j)\}$
    sinon
      si $\big((d_i, t_i) \in \mathcal{F}_k^{cu} \ \vee \ (d_j, t_j) \in \mathcal{F}_k^{cu}\big)$ **alors**
        Aller à "Continuer"
      sinon
        $\mathcal{O}_k^{cu} \leftarrow \mathcal{O}_k^{cu} \bigcup \{(d_i, t_i), (d_j, t_j)\}$

  pour tout $(d_i, t_i, d_j, t_j) \in \mathcal{C}_{k\neq}^u$ **effectuer**
    si $\big((d_i, t_i, d_j, t_j)_c = 0\big)$ **alors**
      si $\big((d_i, t_i) \in \mathcal{O}_k^{cu} \ \vee \ (d_j, t_j) \in \mathcal{F}_k^{cu}\big)$ **alors**
        Aller à "Continuer"
      sinon
        $\mathcal{F}_k^{cu} \leftarrow \mathcal{F}_k^{cu} \bigcup \{(d_i, t_i)\}$ et $\mathcal{O}_k^{cu} \leftarrow \mathcal{O}_k^{cu} \bigcup \{(d_j, t_j)\}$
    sinon
      si $\big((d_i, t_i) \in \mathcal{F}_k^{cu} \ \vee \ (d_j, t_j) \in \mathcal{O}_k^{cu}\big)$ **alors**
        Aller à "Continuer"
      sinon
        $\mathcal{O}_k^{cu} \leftarrow \mathcal{O}_k^{cu} \bigcup \{(d_i, t_i)\}$ et $\mathcal{F}_k^{cu} \leftarrow \mathcal{F}_k^{cu} \bigcup \{(d_j, t_j)\}$

  $\mathcal{I}_k^u \leftarrow \mathcal{I}_k^u \bigcup \{c\}$
  "Continuer"

CHAPITRE 4 : LE PROBLÈME AUXILIAIRE

Dans le présent chapitre, nous allons présenter le problème auxiliaire. Il a été développé par Vovor [1] dans le cadre de sa thèse de doctorat. Il est aussi décrit par Jaumard, Semet et Vovor [2], puis modifié par Labit [3] pour obtenir une modélisation plus adaptée au problème de confection d'horaires d'infirmières.

Nous commençons par décrire les composantes du problème auxiliaire : les graphes G_k, $\hat{G}_k$ et $\hat{G}_{ki}^{cu}$ et les phases PA_{ϕ_1} et PA_{ϕ_2}. Ensuite, nous présentons un algorithme qui est inspiré de PA_{ϕ_2} et qui a la tâche de calculer le nombre d'horaires réalisables pour une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ du PLNE (P^u). Troisièmement, nous allons décrire les modifications à apporter au graphe G_k pour la résolution du problème NSP2. Finalement, nous présentons brièvement une alternative au concept RFRC pour le respect simultané des contraintes de branchement Ryan-Foster.

4.1 Description du problème auxiliaire

Le problème auxiliaire a déjà été introduit à la section 2.3.2.2. Maintenant, nous allons donner plus de détails concernant les différentes composantes de ce second niveau de l'algorithme du branch and price.

Selon les variables duales $[\lambda_L]^i$ et $[\mu_k]^i$ de l'itération courante i de GC, le problème auxiliaire a la tâche de générer des horaires CR^- qui respectent la définition d'une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$, en plus des contraintes intrinsèques de l'infirmière k. Ainsi, afin d'accomplir efficacement cette tâche, le problème auxiliaire est décomposé en

trois parties : (1) le graphe G_k associé à l'infirmière k, (2) l'algorithme PA_{ϕ_1} et (3) l'algorithme PA_{ϕ_2}.

PA_{ϕ_1} est appliqué à G_k afin de définir le graphe d'états $\hat{G}_k$. Ensuite, selon les variables duales $[\lambda_L]^i$ et $[\mu_k]^i$ et une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$, deux pré-traitements sont effectués sur ce graphe d'états afin de définir le nouveau graphe d'états $\hat{G}_{ki}^{cu}$. Finalement, PA_{ϕ_2} est appliquée sur $\hat{G}_{ki}^{cu}$ afin de générer des horaires CR^- selon ces variables duales et cette combinaison.

Dans cette section, nous allons couvrir chaque composante du problème auxiliaire. Premièrement, le graphe G_k est décrit. Ensuite, l'algorithme PA_{ϕ_1} est brièvement présenté. Finalement, l'algorithme PA_{ϕ_2}, qui nous intéresse plus que les deux composantes précédentes, sera présenté et formellement décrit.

4.1.1 Le graphe G_k

Ce graphe est composé d'un ensemble de noeuds $V_k = \{v_0, v_1, \ldots, v_n\}$ et d'un ensemble d'arcs $E_k \subseteq \{(v_i, v_j) : v_i, v_j \in V_k,\ v_i \neq v_j\}$. Il est acyclique, a un noeud source v_0 et un noeud puits v_n. Chaque noeud $v_i \in V_k \setminus \{v_0, v_n\}$ est associé à une affectation $(d_i, t_i) \in \mathcal{D}_{\mathcal{T}_k}$. Chaque arc $(v_i, v_j) \in E_k$ représente la transition entre les affectations (d_i, t_i) et $(d_j, t_j) \in \mathcal{D}_{\mathcal{T}_k}$, qui sont respectivement associées aux noeuds v_i et v_j.

4.1.1.1 Description de l'ensemble de noeuds V_k

Avant de commencer cette description, notons que les composantes $v \in V_k$ sont ordonnées de façon chronologique selon le jour $d \in \mathcal{D}$ qui leur est affecté.

Premièrement, l'ensemble de noeuds V_k est défini selon les contraintes intrinsèques de l'infirmière k. En effet, une affectation $(d,t) \in \mathcal{D}_{\mathcal{T}_k}$ est associée à un noeud $v \in V_k$ si et seulement s'il est possible que l'infirmière travaille sur cette affectation. Dans le cas contraire, aucun noeud n'est créé pour celle-ci. Ces conditions d'existence dépendent en général des pré-affectations, des affectations interdites, s'il y a au moins un arc qui atteint le noeud correspondant et s'il y en a au moins un qui le quitte.

Par exemple, supposons que pour l'infirmière k, l'affectation $(d',t') \in \mathcal{D}_{\mathcal{T}_k}$ lui est obligatoire : $(d',t') \in \mathcal{O}_k$. Dans ce cas, le seul noeud $v \in V_k$ qui est associé au jour d' est le noeud $v' = (d',t') : \nexists v'' \in V_k$, où $v'' = (d',t'') \in \mathcal{D}_{\mathcal{T}_k}$ et $t' \neq t''$. Ceci est dû à une contrainte intrinsèque à toutes infirmières qui stipule qu'elles ne peuvent avoir plus d'une affectation par jour, incluant une affectation consistant à lui accorder un jour de vacance.

Ensuite, en plus des noeuds $v \in V_k$ qui sont définis pour chaque affectation $(d,t) \in \mathcal{D}_{\mathcal{T}_k}$, un noeud "spécial" est défini pour chaque jour $d \in \mathcal{D}$ qui est un jour de vacance possible pour l'infirmière k. Un chemin passant par un tel noeud accorde à l'infirmière le jour de vacance qui est associé à ce noeud. Pour simplifier la notation et la compréhension du présent chapitre, on peut supposer qu'il existe un quart "spécial" $t \in \mathcal{T}_k$ afin de représenter l'attribution d'un jour de vacance sur un jour $d \in \mathcal{D}$, qui doit être un jour de vacance possible pour l'infirmière k. Une telle affectation est aussi notée par $(d,t) \in \mathcal{D}_{\mathcal{T}_k}$.

Finalement, il est important de préciser qu'aucun noeud n'est défini pour les jours de repos accordés à l'infirmière. Un jour $d \in \mathcal{D}$ est donc un jour de repos si et seulement s'il n'est associé à aucun noeud $v \in V_k$ faisant parti du chemin considéré.

4.1.1.2 Description de l'ensemble d'arcs E_k

Tout d'abord, rappelons que chaque arc $(v_i, v_j) \in E_k$ représente la transition entre les affectations (d_i, t_i) et $(d_j, t_j) \in \mathcal{D}_{\mathcal{T}_k}$, qui sont respectivement associées aux noeuds v_i et v_j. Pour un tel arc et pour un chemin qui passe par cet arc, s'il y a un ou plusieurs jours entre d_i et d_j, ils sont tous considérés comme des jours de repos accordés à l'infirmière k.

Ensuite, de manière similaire à l'ensemble de noeud V_k, un arc est associé à une paire d'affectation $(d_i, t_i, d_j, t_j) \in \mathcal{D}^2_{\mathcal{T}_k}$ si et seulement s'il est possible pour l'infirmière k de faire la transition entre ces deux affectations. Dans le cas où on est certain que c'est impossible, ce qu'on peut déterminer à l'aide de certaines contraintes intrinsèques à l'infirmière k, aucun arc n'est associé à cette paire.

Par exemple, supposons que l'infirmière k ne peut avoir plus de trois jours de repos consécutifs et considérons deux noeuds v_i et $v_j \in V_k$ $(v_i \neq v_j)$, qui sont respectivement associés au deux affectations (d_i, t_i) et $(d_j, t_j) \in \mathcal{D}_{\mathcal{T}_k}$. Alors, si le nombre de jours compris entre d_i et d_j est supérieur à 3, l'arc (v_i, v_j) n'est pas créé.

Enfin, un scalaire noté $c_{ij} \in \mathbb{R}$ est défini sur chaque arc $(v_i, v_j) \in E_k$. Ce scalaire représente le coût de passage sur cet arc pour tout sous-chemin $v_0 \rightsquigarrow v_i$, où $v_i \in V_k$, qui est prolongé par l'arc (v_i, v_j) afin d'obtenir le nouveau sous-chemin $v_0 \rightsquigarrow v_i \rightarrow v_j$.

4.1.1.3 Représentation schématique de $G_k = (V_k, E_k)$

Nous pouvons voir à la figure 4.1 une représentation simplifiée du graphe de notre problème auxiliaire. Il est important de faire quelques observations. Premièrement, il

n'y a aucun arc où les deux sommets ont le même jour. Ceci est dû à une contrainte dure qui dit qu'aucune infirmière n'a plus d'une affectation par jour. Ensuite, l'existence des arcs qui quittent la source dépend de la possibilité de commencer l'horaire avec l'affectation de leur noeud cible. De façon similaire pour le puits, l'existence des arcs qui l'atteignent dépend de la possibilité de terminer l'horaire avec l'affectation de leur noeud d'origine. Enfin, si $\nexists v_j \in V_k$ tel que $(v_0, v_j) \in E_k$ et/ou si $\nexists v_i \in V_k$ tel que $(v_i, v_n) \in E_k$, il n'existe evidemment pas de chemin $v_0 \rightsquigarrow v_n$ ni d'horaire pour l'infirmière k.

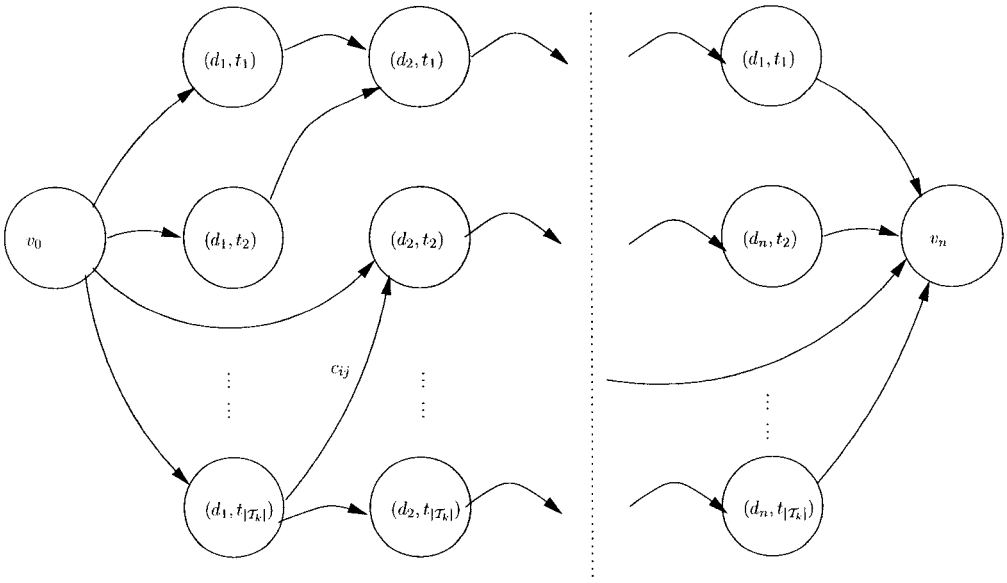


Figure 4.1: Le graphe $G_k = (V_k, E_k)$ associé à l'infirmière $k \in \mathcal{K}$.

4.1.2 Description de PA_{ϕ_1}

L'objectif de cette phase est de définir le graphe d'états $\hat{G}_k$ à l'aide du graphe G_k. Un état (vecteur de ressources) est de dimension 4 et chaque dimension (ressource) est associée à une des 4 contraintes intrinsèques à l'infirmière $k \in \mathcal{K}$ [1]. Un état x_i

[1] Ces contraintes sont énumérées à la section 1.1.2.

d'un noeud $v_i \in V_k$ représente le résultat de chacune de ces 4 ressources obtenu à ce noeud après avoir parcouru un sous-chemin $v_0 \rightsquigarrow v_i$.

Par exemple, considérons le sous-chemin $v_0 \overset{p}{\rightsquigarrow} v'$, où $p = \langle v_0, v' \rangle$, $v' = (d', t') \in \mathcal{D}_{\mathcal{T}_k}$, d' est un jours de vacances possible pour l'infirmière k et t' représente le quart spécial de vacances. Alors, la ressource associée aux jours de vacances, qui comptabilise le nombre de vacances accordées à l'infirmière k, aura une valeur de 1 au noeud v' pour ce sous-chemin.

Dans cette section, nous allons décrire PA_{ϕ_1}. Nous commençons par donner une brève description de cet algorithme. Ensuite, nous présentons la structure de données utilisée pour les ensembles d'états. Enfin, le graphe d'états $\hat{G}_k$, qui résulte de $PA_{\phi_1}(G_k)$ et qui est défini par les ensembles d'états, est donné [2].

4.1.2.1 Description de l'algorithme

PA_{ϕ_1} consiste à calculer pour chaque noeud $v_i \in V_k$ l'ensemble D_i de ses états réalisables. Un état x_i est dit réalisable au noeud $v_i \in V_k$ ($x_i \in D_i$) si et seulement s'il est le résultat d'au moins un sous-chemin $v_0 \rightsquigarrow v_i$ et s'il est le germe d'au moins un sous-chemin $v_i \rightsquigarrow v_n$. Par conséquent, pour tout état $x_i \in D_i$, il existe au moins un chemin $v_0 \rightsquigarrow v_i \rightsquigarrow v_n$ pour lequel x_i est l'état des ressources au noeud v_i.

Considérons deux états x'_i et $x''_i \in D_i$ ($x'_i \neq x''_i$) qui sont respectivement le résultat des sous-chemins $v_0 \overset{p'}{\rightsquigarrow} v_i$ et $v_0 \overset{p''}{\rightsquigarrow} v_i$. Il est important de noter que ces deux sous-chemins sont tels que $p' \neq p''$. Car, dans le cas contraire, nous aurions eu $x'_i = x''_i$.

[2] Ces ensembles d'états sont définis dans la prochaine section.

Par la suite, pour chaque état $x_j \in D_j$ du noeud $v_j \in V_k$, un ensemble de vecteurs de ressources $R(x_j)$ est défini. Chaque état $x_i \in R(x_j)$ est un état réalisable à un noeud $v_i \in V_k$, où $(v_i, v_j) \in E_k$ et $x_i \in D_i$. L'état x_i au noeud v_i a donné naissance à l'état x_j au noeud v_j par le passage sur l'arc (v_i, v_j). En effet, x_i est obtenu au noeud v_i après avoir parcouru un sous-chemin $v_0 \rightsquigarrow v_i$, ce sous-chemin est ensuite prolongé par le passage sur l'arc (v_i, v_j) et (enfin) le nouvel état réalisable x_j est obtenu au noeud v_j. Ainsi, x_j représente l'état de nos ressources pour notre nouveau sous-chemin $v_0 \rightsquigarrow v_i \rightarrow v_j$.

Enfin, lorsque PA_{ϕ_1} est complété, l'ensemble $R(x_j)$ d'un état réalisable $x_j \in D_j$ au noeud $v_j \in V_k$ contient tous les états réalisables desquels l'état x_j est né. Par conséquent, il est alors possible de définir l'ensemble des chemins $v_0 \rightsquigarrow v_n$ (donc l'ensemble des horaires $s \in S_k$) grâce à l'ensemble des états réalisables finaux D_n et aux ensembles $R(x_n)$ de chaque $x_n \in D_n$. Notons que cet ensemble d'états réalisables est celui du noeud puits v_n.

Finalement, le lecteur est référé à Vovor [1] pour avoir la description complète de PA_{ϕ_1} et à Labit [3] pour la modélisation des 4 ressources dans le graphe G_k.

4.1.2.2 Structure de données des ensembles d'états

Maintenant, nous allons présenter la structure de données utilisée pour l'implémentation des ensembles d'états D_j aux noeuds $v_j \in V_k$.

Premièrement, notons par $\mathcal{R} = \{r_1, r_2, \ldots, r_{|\mathcal{R}|}\}$ l'ensemble des ressources associées aux contraintes intrinsèques. D_j est représenté par un arbre n-aire de profondeur $|\mathcal{R}|$, où chaque niveau correspond à une ressource $r \in \mathcal{R}$ et est représenté par

un tableau unidimensionnel de taille égale au nombre de valeurs possibles pour cette ressource. Les valeurs minimale et maximale d'une ressource $r \in \mathcal{R}$ au noeud $v_j \in V_k$ sont respectivement notées par $\underline{\varphi}_{jr}$ et $\overline{\varphi}_{jr}$. La position d'une valeur x_{jr} d'une ressource $r \in \mathcal{R}$, où $\underline{\varphi}_{jr} \leq x_{jr} \leq \overline{\varphi}_{jr}$, est donnée par $x_{jr} - \underline{\varphi}_{jr} + 1$ et peut donc être calculée en temps constant $O(1)$.

Ensuite, les états $x_j \in D_j$ sont ordonnés dans un ordre lexicographique et correspondent aux feuilles de cette arborescence. Considérons un état $x'_j = (x'_{jr_1}, x'_{jr_2}, \ldots, x'_{jr_{|\mathcal{R}|}})$, où $\forall r \in \mathcal{R}\ \underline{\varphi}_{jr} \leq x'_{jr} \leq \overline{\varphi}_{jr}$. Alors, $x'_j \in D_j$ si et seulement (1) si pour $\forall r \in \mathcal{R} \setminus \{r_{|\mathcal{R}|}\}$, la position $x'_{jr} - \underline{\varphi}_{jr} + 1$ pointe vers un tableau associé à la ressource $r + 1$ et (2) si la position $x'_{jr_{|\mathcal{R}|}} - \underline{\varphi}_{jr_{|\mathcal{R}|}} + 1$ pointe vers la définition vectorielle de l'état x'_j. De plus, pour $r \in \mathcal{R} \setminus \{r_{|\mathcal{R}|}\}$ et $c \in \mathbb{N}$ tel que $\underline{\varphi}_{jr} \leq c \leq \overline{\varphi}_{jr}$, si $\nexists x_j \in D_j$ tel que $x_{jr} = c$, aucun tableau n'est pointé par la position correspondante $c - \underline{\varphi}_{jr} + 1$.

Troisièmement, considérons l'état x'_j décrite ci-dessus et supposons que $x'_j \in D_j$. La position des $|\mathcal{R}|$ valeurs x'_{jr} de chaque $r \in \mathcal{R}$ pouvant être calculée en temps $O(1)$, la feuille associée à l'état x'_j peut donc être trouvée en temps $O(|\mathcal{R}|)$. Par conséquent, le temps requis pour déterminer si un état x_j est tel que $x_j \in D_j$ est, dans le pire des cas, $O(|\mathcal{R}|)$. Enfin, le temps requis pour ajouter un état x_j à D_j est aussi $O(|\mathcal{R}|)$.

Finalement, notons par $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ l'ensemble des ressources associées à nos 4 contraintes intrinsèques et considérons l'ensemble D_j d'un noeud $v_j \in V_k$ donné en exemple par la figure 4.2 [3]. Cet ensemble est défini par $D_j = \{x_{j_1}, x_{j_2}, x_{j_3}\}$, où $x_{j_1} = (1, 1, 0, 0)$, $x_{j_2} = (1, 1, 1, 0)$ et $x_{j_3} = (3, 1, 1, 0)$ et où nous avons supposé que $\underline{\varphi}_{jr} = 0\ \forall r \in \mathcal{R}$. Il est important de remarquer que les deux états x_{j_1} et x_{j_2} ont ceci en commun : $x_{j_1 r_1} = x_{j_2 r_1}$ et $x_{j_1 r_2} = x_{j_2 r_2}$.

[3] Voir la section 1.1.2 pour la description de ces contraintes.

Dans la figure 4.2, une liste chaînée a été représentée. Nous allons dès maintenant expliquer pourquoi celle-ci a été introduite dans l'implémentation de l'ensemble D_j.

Structure de données hybride pour D_j

L'arborescence décrite ci-dessus pour un ensemble d'états D_j avait déjà été présentée par Vovor [1], quoique d'une façon très compacte. De plus, cette arborescence a aussi été utilisée pour l'implémentation des ensembles d'états U_i des noeuds $v_i \in V_k$ et U_{ij} des arcs $(v_i, v_j) \in E_k$ [4]. Pour l'ensemble d'états $R(x_j)$ d'un état $x_j \in D_j$ au noeud $v_j \in V_k$, une liste chaînée suffit car en aucun moment dans les algorithmes PA_{ϕ_1} et PA_{ϕ_2}, ses éléments $x_i \in R(x_j)$ doivent être accédés en temps "optimal" $O(|\mathcal{R}|)$. Par contre, contrairement à l'ensemble D_j, son parcours se fait en temps "optimal" $O(|R(x_j)|)$.

Intuitivement, le temps nécessaire pour parcourir l'ensemble D_j est de beaucoup supérieur au temps optimal $O(|D_j|)$. Car ce parcours se fait en traversant l'arborescence de la figure 4.2 dans un ordre lexicographique. Pour éliminer ce désavantage que possède les ensembles D_j face aux ensembles $R(x_j)$, il suffit simplement de définir une liste chaînée composée des états $x_j \in D_j$. Notons qu'il n'est pas nécessaire que les états de cette liste soient ordonnés par ordre lexicographique. Par conséquent, lorsqu'un état x', où $x' \notin D_j$, est ajouté à D_j, il sera aussi ajouté à la fin de cette liste chaînée. Le temps requis par cette opération n'est pas perturbé par la gérance de cette liste et est toujours égale à $O(|\mathcal{R}|)$.

Dans cette nouvelle structure de données hybride, qui est composée d'un arbre n-aire et d'une liste chaînée, l'opération d'ajouter un état x' à D_j et celle d'accéder à la feuille d'un état $x_j \in D_j$ se font en temps optimal $O(|\mathcal{R}|)$. Ces temps demeurent

[4] Voir Vovor [1] pour la définition de ces ensembles.

inchangés par rapport à l'ancienne structure. Par contre, grâce à l'insertion d'une liste chaînée, le parcours séquentiel des états $x_j \in D_j$ se fait en temps $O(|D_j|)$, qui est aussi optimal. L'algorithme PA_{ϕ_2} sera donc plus performant grâce à cette nouvelle arborescence hybride, car il parcourt de façon séquentiel les états $x_j \in D_j$ de chaque noeud $v_j \in V_k$.

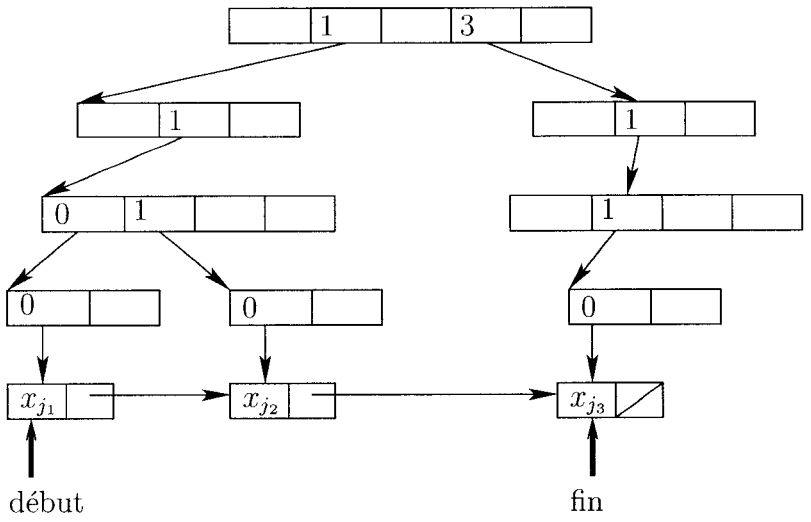


Figure 4.2: Représentation de l'ensemble d'état D_j d'un noeud $v_j \in V_k$.

4.1.2.3 Le graphe d'états $\hat{G}_k$

Le graphe d'états $\hat{G}_k$ est obtenu grâce à l'application de PA_{ϕ_1} au graphe G_k. $\hat{G}_k$ est défini par les ensembles d'états D_j de tous les noeuds $v_j \in V_k$ et par les ensembles d'états $R(x_j)$ associés à tous les états réalisables $x_j \in D_j$.

Dans ce graphe d'état, un noeud ne représente plus une affectation $(d_i, t_i) \in \mathcal{D}_{\mathcal{T}_k}$, il représente plutôt un état réalisable $x_i \in D_i$ d'un noeud $v_i \in V_k$. De plus, un arc ne représente plus la transition entre une paire d'affectation $(d_i, t_i, d_j, t_j) \in \mathcal{D}^2_{\mathcal{T}_k}$, il représente plutôt la transition entre deux états réalisables $x_i \in D_i$ au noeud $v_i \in V_k$

et $x_j \in D_j$ au noeud $v_j \in V_k$. Un tel arc existe si et seulement s'il est possible de transiter entre ces deux états : si $x_i \in R(x_j)$. Notons que si cet arc existe, l'arc qu'il représente dans le graphe G_k existe aussi : $(v_i, v_j) \in E_k$.

Donnons la définition formelle du graphe $\hat{G}_k$ et celle d'un sous-chemin d'états de ce graphe.

Définition 4.1 (Graphe d'états) *Soit une infirmière $k \in \mathcal{K}$. Le graphe d'états $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$ associé au graphe $G_k = (V_k, E_k)$ est défini comme suit :*

- *l'ensemble de noeuds $\hat{V}_k = \{v_i \cdot x_i : v_i \in V_k,\ x_i \in D_i\}$,*
- *l'ensemble d'arcs $\hat{E}_k = \{(v_i \cdot x_i,\ v_j \cdot x_j) : v_i,\ v_j \in V_k,\ x_i \in D_i,\ x_j \in D_j,\ x_i \in R(x_j)\}$.*

Le coût d'un arc d'état $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$ est identique à celui de l'arc correspondant $(v_i, v_j) \in E_k$ et est donc noté par c_{ij}.

Définition 4.2 (Sous-chemin d'états) *Considérons le graphe d'états $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$ d'une infirmière $k \in \mathcal{K}$. Un sous-chemin de ce graphe est noté par $v_0 \cdot x_0 \overset{p}{\rightsquigarrow} v_j \cdot x_j$, où $v_0 \cdot x_0$ et $v_j \cdot x_j \in \hat{V}_k$, et est défini comme suit :*

$p = \langle v'_1 \cdot x'_1,\ v'_2 \cdot x'_2,\ \ldots,\ v'_{|p|} \cdot x'_{|p|} \rangle$, *où* $v'_1 \cdot x'_1 = v_0 \cdot x_0$, $v'_{|p|} \cdot x'_{|p|} = v_j \cdot x_j$, $v'_i \cdot x'_i \in \hat{V}_k$ *et* $(v'_i \cdot x'_i,\ v'_{i+1} \cdot x'_{i+1}) \in \hat{E}_k$ *pour* $i = 1,\ 2,\ \ldots,\ |p| - 1$.

Le coût de ce chemin est noté par $c(p)$ et est défini par le coût $c_{i(i+1)}$ de ses arcs $(v'_i \cdot x'_i,\ v'_{i+1} \cdot x'_{i+1}) \in \hat{E}_k$: $c(p) = \sum_{i=1}^{|p|-1} c_{i(i+1)}$.

Finalement, une représentation du graphe d'états $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$ d'une infirmière $k \in \mathcal{K}$ est donnée par la figure 4.3. Il est important de noter qu'un seul noeud

source noté $v_0 \cdot x_0$ est défini dans ce graphe. En effet, même s'il est possible que l'ensemble D_0 des états réalisables au noeud v_0, qui est le noeud source du graphe G_k, contienne plus qu'un état, nous verrons prochainement qu'un seul de ces états est considéré par PA_{ϕ_2} pour la génération d'horaires de l'horizon courant. Aussi, nous attirons l'attention du lecteur sur le fait qu'il y a plusieurs noeuds puits définis dans ce graphe d'état. Effectivement, un noeud puits est défini pour chaque état réalisable $x_n \in D_n$ du noeud v_n, qui est le noeud puits du graphe G_k.

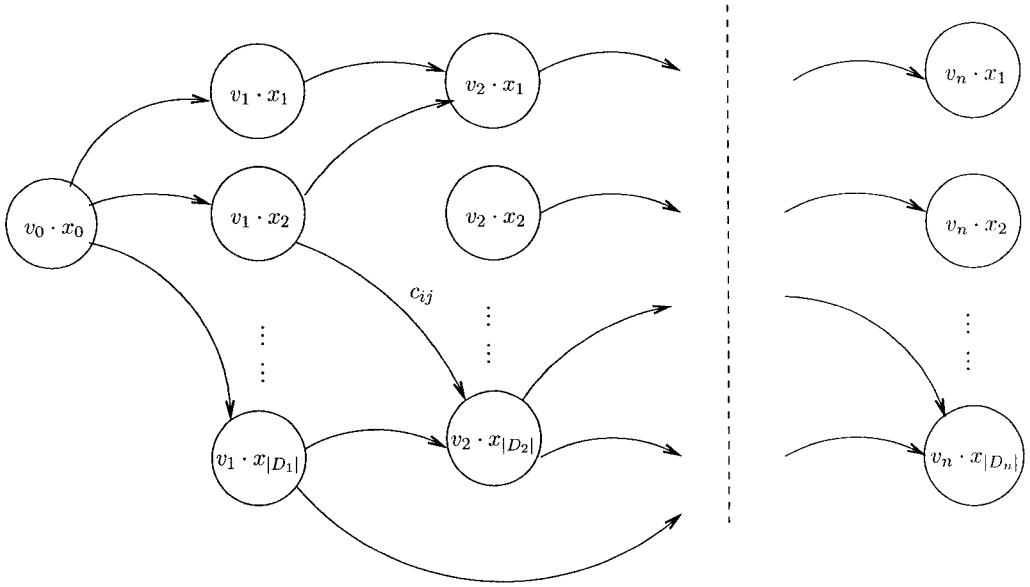


Figure 4.3: Représentation du graphe d'états $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$ de l'infirmière $k \in \mathcal{K}$.

4.1.3 Description de PA_{ϕ_2}

Après avoir obtenu le graphe d'états $\hat{G}_k$ par $PA_{\phi_1}(G_k)$, un nouveau graphe d'états noté $\hat{G}^{cu}_{ki}$ doit être défini pour représenter les variables duales de l'itération courante i de GC en plus de la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}^u_k}$. La méthode PA_{ϕ_2}, qui est un algorithme de plus court chemin, est appliquée au graphe $\hat{G}^{cu}_{ki}$ afin de générer des

horaires qui sont de coûts réduits négatifs selon ces variables duales et qui respectent les affectations obligatoires et interdites de cette combinaison.

Le coût réduit c_{ks} d'un horaire $s \in S_k$ et d'une infirmière $k \in \mathcal{K}$ est, selon les variables duales $[\lambda_L]^i$ et $[\mu_k]^i$, défini de cette façon :

$$c_{ks} = p_{ks} - \mu_k - \sum_{(d,p)\in\mathcal{D}_\mathcal{P}} \sum_{L\in\mathcal{L}_{dp}} \sum_{t\in\mathcal{T}_k} (\alpha_{tp}\ b_{kL}\ a_{ksdt}\ \lambda_L). \tag{4.1}$$

La constante p_{ks} est le poids de l'horaire s. Les variables duales μ_k et λ_L sont respectivement associées à la contrainte de partionnement de cette infirmière et à la contrainte de quotas (Q_L), où $L \in \mathcal{L}_{dp}$ et $(d,p) \in \mathcal{D}_\mathcal{P}$ [5].

Pour bien comprendre l'algorithme PA_{ϕ_2}, nous commençons par expliquer les pré-traitements nécessaires à la définition du graphe $\hat{G}^{cu}_{ki}$. Ensuite, nous décrivons de façon formelle l'algorithme PA_{ϕ_2}. Finalement, nous analysons les résultats possibles de cette méthode.

4.1.3.1 Définition du graphe $\hat{G}^{cu}_{ki}$

Le graphe $\hat{G}^{cu}_{ki}$ est défini par $\hat{G}_k$ et par l'application des deux pré-traitements décrits ci-dessous.

Le premier pré-traitement consiste à inclure toutes les affectations $(d,t) \in \mathcal{O}^{cu}_k$ et à exclure toutes les affectations $(d,t) \in \mathcal{F}^{cu}_k$ de chaque chemin d'états $v_0 \cdot x_0 \rightsquigarrow v_n \cdot x_n$ obtenu par $PA_{\phi_2}(\hat{G}^{cu}_{ki})$. Pour obtenir de tel chemin, il suffit simplement de redéfinir

[5] Pour avoir une définition de p_{ks}, le lecteur est référé à Labit [3].

le coût c_{ij} d'un arc $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$, où $v_i = (d_i, t_i) \in \mathcal{D}_{\mathcal{T}_k}$ et $v_j = (d_j, t_j) \in \mathcal{D}_{\mathcal{T}_k}$ si $v_j \neq v_n$, comme suit :

$$c_{ij} \leftarrow \begin{cases} +\infty & \text{si } v_j \neq v_n \text{ et si } \exists v' \in V_k \text{ tq. } v' = (d', t') \in \mathcal{O}_k^{cu}, \\ & (d', t') \neq (d_j, t_j) \text{ et } d_i < d' \leq d_j, \\ +\infty & \text{si } v_j = v_n \text{ et si } \exists v' \in V_k \text{ tq. } v' = (d', t') \in \mathcal{O}_k^{cu} \\ & \text{et } d_i < d', \\ +\infty & \text{si } v_j \neq v_n \text{ et si } (d_j, t_j) \in \mathcal{F}_k^{cu}, \\ c_{ij} & \text{sinon (aucune modification).} \end{cases}$$

Précisons que si $v_i = v_0$, d_0 représente le prédécesseur immédiat du premier jour de l'horizon courant, soit le jour d_1.

Grâce à ce premier pré-traitement et étant donné que PA_{ϕ_2} consiste à calculer pour chacun des puits $v_n \cdot x_n \in \hat{V}_k$ le chemin $v_0 \cdot x_0 \rightsquigarrow v_n \cdot x_n$ de coût optimal dans le graphe $\hat{G}_{ki}^{cu}$, aucune des affectations interdites ($(d,t) \in \mathcal{F}_k^{cu}$) ne sera dans ce chemin. Inversement, toutes les affectations obligatoires ($(d,t) \in \mathcal{O}_k^{cu}$) seront dans celui-ci. Ainsi, en supposant que le coût optimal associé au puits $v_n \cdot x_n \in \hat{V}_k$ est inférieur à $+\infty$, le chemin optimal de ce puits sera donc réalisable pour la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$.

Pour un noeud d'état $v_j \cdot x_j \in \hat{V}_k$, la définition donnée ci-dessous défini un sous-chemin $v_0 \cdot x_0 \rightsquigarrow v_j \cdot x_j$ réalisable pour une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$.

Définition 4.3 (Sous-chemin d'états réalisable) *Considérons le graphe d'états $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$ d'une infirmière $k \in \mathcal{K}$ et un sous-chemin $v_0 \cdot x_0 \overset{p}{\rightsquigarrow} v_j \cdot x_j$ de ce graphe, où $v_j \cdot x_j \in \hat{V}_k$ et p est défini selon la définition 4.2 : $p = \langle v'_1 \cdot x'_1,\ v'_2 \cdot x'_2,\ \ldots,\ v'_{|p|} \cdot x'_{|p|} \rangle$.*

Ce sous-chemin est dit réalisable pour la combinaison $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ *du problème* (P^u) *si et seulement :*

$$si\ v'_{|p|} = v_n \begin{cases} \forall\ v_i \cdot x_i \in V_p \setminus \{v'_{|p|} \cdot x'_{|p|}\},\ v_i = (d_i, t_i) \notin \mathcal{F}_k^{cu}, \\ \forall (d', t') \in \mathcal{O}_k^{cu},\ \exists v_i \cdot x_i \in V_p \setminus \{v'_{|p|} \cdot x'_{|p|}\}\ tq.\ v_i = (d', t'), \end{cases}$$

$$si\ v'_{|p|} \neq v_n \begin{cases} \forall\ v_i \cdot x_i \in V_p,\ v_i = (d_i, t_i) \notin \mathcal{F}_k^{cu}, \\ \forall (d', t') \in \mathcal{O}_k^{cu}\ où\ d' \leq d_{|p|},\ \exists v_i \cdot x_i \in V_p\ tq.\ v_i = (d', t'), \end{cases}$$

$où\ V_p = \{v'_2 \cdot x'_2,\ v'_3 \cdot x'_3,\ \ldots,\ v'_{|p|} \cdot x'_{|p|}\}.$

Pour un noeud d'état $v_j \cdot x_j \in \hat{V}_k$, on note par $P_k^{cu}(v_j \cdot x_j) \subseteq \{p : v_0 \cdot x_0 \overset{p}{\rightsquigarrow} v_j \cdot x_j\}$ l'ensemble des sous-chemins $v_0 \cdot x_0 \rightsquigarrow v_j \cdot x_j$ qui sont réalisables pour la combinaison $(k,c) \in \mathcal{K}_{\mathcal{I}_k^u}$ du problème (P^u). Ainsi, en considérant tous les puits $v_n \cdot x_n \in \hat{V}_k$, l'ensemble des chemins $v_0 \cdot x_0 \rightsquigarrow v_n \cdot x_n$ qui sont réalisables pour cette combinaison est noté par $P_k^{cu} = \bigcup_{v_n \cdot x_n \in \hat{V}_k} P_k^{cu}(v_n \cdot x_n)$. Ainsi, à chacun des chemins $p \in P_k^{cu}$ correspond un unique horaire $s \in S_k^{cu}$ ($|P_k^{cu}| = |S_k^{cu}|$) et nous avons

$$\forall s \in S_k^{cu} \begin{cases} \forall (d,t) \in \mathcal{F}_k^{cu},\ a_{ksdt} = 0, \\ \forall (d,t) \in \mathcal{O}_k^{cu},\ a_{ksdt} = 1. \end{cases}$$

Le deuxième pré-traitement consiste à redéfinir pour une seconde fois le coût c_{ij} d'un arc $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$ pour que le coût $c(p)$ d'un chemin d'états $v_0 \cdot x_0 \overset{p}{\rightsquigarrow} v_n \cdot x_n$, où $p \in P_k^{cu}$, corresponde au coût réduit c_{ks} de l'horaire $s \in S_k^{cu}$ associé à ce chemin [6]. Ainsi, pour chaque arc qui quitte la source $v_0 \cdot x_0$, soit les arcs $(v_0 \cdot x_0,\ v_j \cdot x_j) \in \hat{E}_k$, μ_k est incluse dans cette définition. Aussi, les variables duales λ_L de chaque contrainte de quotas (Q_L) qui sont couvertes par l'affectation du noeud cible $v_j \cdot x_j \in \hat{V}_k$ sont aussi

[6] $c(p)$ et c_{ks} ont respectivement été définis par la définition 4.2 et l'équation 4.1.

incluses dans la définition de c_{ij}. Enfin, le coût de chaque arc $(v_i \cdot x_i,\ v_n \cdot x_n) \in \hat{E}_k$ n'est pas modifié car aucune affectation n'est associée au noeud puits $v_n \cdot x_n \in \hat{V}_k$. Le coût c_{ij} d'un arc $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$, où $v_j = (d_j, t_j) \in \mathcal{D}_{\mathcal{T}_k}$ si $v_j \neq v_n$, est alors redéfini comme suit :

$$c_{ij} \leftarrow \begin{cases} c_{ij} - \mu_k - \sum\limits_{p \in \mathcal{P}} \sum\limits_{L \in \mathcal{L}_{d_j p}} (\alpha_{t_j p}\ b_{kL}\ a_{ksd_j t_j}\ \lambda_L) & \text{si } v_i = v_0, \\ c_{ij} \quad - \sum\limits_{p \in \mathcal{P}} \sum\limits_{L \in \mathcal{L}_{d_j p}} (\alpha_{t_j p}\ b_{kL}\ a_{ksd_j t_j}\ \lambda_L) & \text{si } v_i \neq v_0 \text{ et } v_j \neq v_n, \\ c_{ij} & \text{si } v_j = v_n. \end{cases}$$

Il est important de préciser que la variable μ_k est incluse une et une seule fois dans le coût $c(p)$ de tout chemin d'états $v_0 \cdot x_0 \overset{p}{\rightsquigarrow} v_n \cdot x_n$, où $p \in P_k^{cu}$. En effet, un et un seul arc $(v_0 \cdot x_0,\ v_j \cdot x_j) \in \hat{E}_k$ est inclus dans un tel chemin.

4.1.3.2 Description de l'algorithme PA_{ϕ_2}

Lorsque le graphe $\hat{G}_{ki}^{cu}$ est défini, PA_{ϕ_2} est utilisé par Pricing afin de générer des colonnes CR^- pour la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ et lors de l'itération i de GC [7]. Rappelons que cet algorithme est appliqué pour résoudre une relaxation continue $(\dot{R}^u)$ au noeud u de l'arbre de recherche [8].

Avant d'expliquer l'algorithme PA_{ϕ_2}, nous devons introduire de nouvelles notations. Pour un noeud $v_j \cdot x_j \in \hat{V}_k$, on note par $c(v_j \cdot x_j) \in \mathbb{R}$ le coût optimal de tous les sous-chemins $v_0 \cdot x_0 \overset{p}{\rightsquigarrow} v_j \cdot x_j$ qui respecte la combinaison (k, c) :

$$c(v_j \cdot x_j) = \min\{c(p) : p \in P_k^{cu}(v_j \cdot x_j)\}.$$

[7] GC et Pricing sont respectivement décrits par les algorithmes 2.1 et 2.2.
[8] $(\dot{R}^u)$ représente l'un des deux problèmes suivants : (R^u) ou (R^{ua}).

De plus, on note par $p(v_j \cdot x_j) \in \hat{V}_k$, le noeud d'état prédécesseur de $v_j \cdot x_j$ dans le sous-chemin $p' \in P_k^{cu}(v_j \cdot x_j)$ qui est de coût optimal $c(v_j \cdot x_j)$. Ce sous-chemin est alors représenté par $v_0 \cdot x_0 \overset{p''}{\rightsquigarrow} v_i \cdot x_i \rightarrow v_j \cdot x_j$, où $p'' \in P_k^{cu}(v_i \cdot x_i)$ est de coût optimal $c(v_i \cdot x_i)$, $v_i \cdot x_i = p(v_j \cdot x_j)$ et $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$.

L'algorithme PA_{ϕ_2} consiste donc à calculer pour chaque noeud $v_j \cdot x_j \in \hat{V}_k$, les valeurs $c(v_j \cdot x_j)$ et $p(v_j \cdot x_j)$. Ces valeurs sont définies par les deux équations récurrentes 4.2 et 4.3.

$$c(v_j \cdot x_j) = \begin{cases} 0, \text{ si } v_j \cdot x_j = v_0 \cdot x_0, \\ min\{c(v_i \cdot x_i) + c_{ij} : v_i \cdot x_i \in V_k,\ (v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k\}. \end{cases} \tag{4.2}$$

$$p(v_j \cdot x_j) = \begin{cases} \text{nul, si } v_j \cdot x_j = v_0 \cdot x_0, \\ \text{argmin}\{c(v_i \cdot x_i) + c_{ij} : v_i \cdot x_i \in V_k,\ (v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k\}. \end{cases} \tag{4.3}$$

Premièrement, un seul noeud source noté $v_0 \cdot x_0 \in \hat{V}_k$ est considéré par PA_{ϕ_2} pour générer les chemins optimaux. Ce noeud correspond à l'état de départ x_0 qui est défini par l'horaire affecté à l'infirmière k dans l'horizon précédent. Si $x_0 \notin D_0$, aucun chemin d'état du noeud source v_0 au noeud puits v_n n'existe pour cet état de départ et l'infirmière k ne possède aucun horaire pour l'horizon courant : $S_k = \emptyset$. Dans le cas contraire, $c(v_0 \cdot x_0) \leftarrow 0$ et $c(v_0 \cdot x_0') \leftarrow +\infty$ pour $\forall\ v_0 \cdot x_0' \in \hat{V}_k \setminus \{v_0 \cdot x_0\}$.

Ensuite, on traite chacun des noeuds $v_j \cdot x_j \in \hat{V}_k$ en débutant avec ceux associés au noeud $v_j = v_1$ et en concluant avec ceux du noeud puits $v_j = v_n$. Pour chacun de ces noeuds, tous leurs prédécesseurs possibles $v_i \cdot x_i \in \hat{V}_k$, où $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$, sont examinés afin de définir leur coût et leur prédécesseur optimaux, respectivement $c(v_j \cdot x_j)$ et $p(v_j \cdot x_j)$.

Troisièmement, après que tous les noeuds aient été traités, il est maintenant possible de définir le chemin d'états optimal pour chacun des puits $v_n \cdot x_n \in \hat{V}_k$. Le chemin $p \in P_k^{cu}(v_n \cdot x_n)$ qui est de coût optimal $c(p) = c(v_n \cdot x_n)$ est défini par la définition 4.4.

Définition 4.4 (Chemin optimal pour un puits) *Considérons un puits $v_n \cdot x_n \in \hat{V}_k$. Le chemin $v_0 \cdot x_0 \overset{p}{\rightsquigarrow} v_n \cdot x_n$, où $p \in P_k^{cu}(v_n \cdot x_n)$, qui est de coût optimal $c(v_n \cdot x_n)$ est défini comme suit :*

$p = \langle v'_1 \cdot x'_1,\ v'_2 \cdot x'_2,\ \ldots,\ v'_{|p|} \cdot x'_{|p|} \rangle$, *où* $v'_1 \cdot x'_1 = v_0 \cdot x_0$, $v'_{|p|} \cdot x'_{|p|} = v_n \cdot x_n$, $v'_i \cdot x'_i = p(v'_{i+1} \cdot x'_{i+1})$ *pour* $i = 1,\ 2,\ \ldots,\ |p| - 1$.

Pour chacun des puits $v_n \cdot x_n \in \hat{V}_k$, l'horaire $s \in S_k^{cu}$ associé au chemin optimal de ce puits est retourné par PA_{ϕ_2} si et seulement si son coût réduit c_{ks}, qui est défini par $c_{ks} \leftarrow c(v_n \cdot x_n)$, est tel que $c_{ks} < 0$. On note par $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \subseteq S_k^{cu}$, l'ensemble de colonnes CR^- obtenu par l'application de PA_{ϕ_2} au graphe $\hat{G}_{ki}^{cu}$.

4.1.3.3 Résultats possibles de $PA_{\phi_2}(\hat{G}_{ki}^{cu})$

Maintenant, nous allons présenter les dénouements possibles de l'application de PA_{ϕ_2} au graphe $\hat{G}_{ki}^{cu}$. Rappelons que nous considérons toujours la génération de colonnes CR^- pour l'itération courante i de GC, où $(\dot{R}^{ui})$ est le problème restreint de cette itération, et pour une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$.

Premièrement, l'ensemble $PA_{\phi_2}(\hat{G}_{ki}^{cu})$ est composé d'horaires $s \in S_k^{cu}$ qui ne sont pas dans le problème restreint $(\dot{R}^{ui})$ et qui sont de coûts réduits négatifs : $PA_{\phi_2}(\hat{G}_{ki}^{cu}) \subseteq$

$\{s \in S_k^{cu} : s \notin S_k^{ui} \text{ et } c_{ks} < 0\}$. Malgré que tous les horaires $s \in S_k^{cu} \bigcap S_k^{ui}$ correspondent chacun à un unique chemin $p \in P_k^{cu}$ et peuvent donc être obtenus par PA_{ϕ_2}, il est en effet impossible pour PA_{ϕ_2} de générer pour une seconde fois de tels horaires, car leur coût réduit $c_{ks} = c(p)$ est tel que $c_{ks} \geq 0$. Plus précisément, les variables $s \in S_k^{cu} \bigcap S_k^{ui}$ qui sont en bases sont telles que $c_{ks} = 0$ et celles qui sont hors bases sont telles que $c_{ks} > 0$ [9].

Soit un noeud puits $v_n \cdot x'_n \in \hat{V}_k$. Supposons que le coût du chemin optimal $v_0 \cdot x_0 \overset{p^*}{\rightsquigarrow} v_n \cdot x'_n$, où $p^* \in P_k^{cu}(v_n \cdot x'_n)$, est tel que $c(p^*) = c(v_n \cdot x'_n) \geq 0$. On peut alors conclure que $\forall p \in P_k^{cu}(v_n \cdot x'_n)$, $c(p) \geq 0$. De plus, notons par $s^* \in S_k^{cu}$ l'horaire de ce chemin optimal et supposons que $s^* \in S_k^{cu} \bigcap S_k^{ui}$. Alors, même si l'horaire s^* est un des $|\{v_n \cdot x_n \in \hat{V}_k\}|$ résultats de l'algorithme PA_{ϕ_2}, il ne sera pas retourné par ce dernier car $c_{ks^*} \geq 0$.

Ensuite, l'ensemble de colonnes CR^- obtenu par cet application est tel que $|PA_{\phi_2}(\hat{G}_{ki}^{cu})| \leq |\{v_n \cdot x_n \in \hat{V}_k\}|$. Parmi toutes ces colonnes, l'une d'elle, notée $s^* \in PA_{\phi_2}(\hat{G}_{ki}^{cu})$, est de coût optimal c_{ks^*} et est associé au chemin $p^* \in P_k^{cu}$ qui est aussi de coût optimal $c(p^*) = c_{ks^*}$:

$$\begin{aligned} c_{ks^*} &= \min\{c_{ks} : s \in S_k^{cu} \setminus S_k^{ui}\}, \\ c(p^*) &= \min\{c(v_n \cdot x_n) : v_n \cdot x_n \in \hat{V}_k\}. \end{aligned}$$

Concernant les autres colonnes $s \in PA_{\phi_2}(\hat{G}_{ki}^{cu})$, où $s \neq s^*$, elles ne sont pas nécessairement les $|PA_{\phi_2}(\hat{G}_{ki}^{cu})| - 1$ autres meilleures colonnes selon leur coût réduit c_{ks}. Effectivement, il est possible qu'il $\exists s'$ et $s'' \in S_k^{cu} \setminus S_k^{ui}$ tel que $s' \in PA_{\phi_2}(\hat{G}_{ki}^{cu})$ et $s'' \notin PA_{\phi_2}(\hat{G}_{ki}^{cu})$ alors que $c_{ks''} < c_{ks'}$. Ceci est une conséquence directe du fait que

[9] Pour avoir plus de détail sur ce postulat, le lecteur est référé à Chvatal [20] ou à Nemhauser [22].

pour chaque puits $v_n \cdot x_n \in \hat{V}_k$, un maximum d'un chemin $p \in P_k^{cu}(v_n \cdot x_n)$ est tel que $p \in PA_{\phi_2}(\hat{G}_{ki}^{cu})$.

Par exemple, considérons trois chemins p_1, p_2 et $p_3 \in P_k^{cu}$, où p_1 et $p_2 \in P_k^{cu}(v_n \cdot x'_n)$, $p_3 \in P_k^{cu}(v_n \cdot x''_n)$ et $v_n \cdot x'_n$ et $v_n \cdot x''_n \in \hat{V}_k$. Supposons que $c(p_1) = c(v_n \cdot x'_n)$, que $c(p_3) = c(v_n \cdot x''_n)$ et que $c(p_1) < c(p_2) < c(p_3)$. Dans ce cas, les colonnes s_1, s_2 et $s_3 \in S_k^{cu}$, qui sont respectivement associées aux trois chemins p_1, p_2 et p_3, seront telles que s_1 et $s_3 \in PA_{\phi_2}(\hat{G}_{ki}^{cu})$ et $s_2 \notin PA_{\phi_2}(\hat{G}_{ki}^{cu})$ alors que $c_{ks_2} < c_{ks_3}$.

Finalement, $PA_{\phi_2}(\hat{G}_{ki}^{cu}) = \emptyset$ si seulement si $\forall v_n \cdot x_n \in \hat{V}_k$, $c(v_n \cdot x_n) \geq 0$. Dans ce cas, étant donné que PA_{ϕ_2} consiste à calculer pour chaque noeud puits $v_n \cdot x_n \in \hat{V}_k$ le chemin d'état $v_0 \cdot x_0 \rightsquigarrow v_n \cdot x_n$ de coût optimal $c(v_n \cdot x_n)$, nous pouvons conclure que $\nexists s \in S_k^{cu} \setminus S_k^{ui}$ tel que $c_{ks} < 0$. En d'autres termes, il n'existe pas de colonnes de coûts réduits négatifs qui sont réalisables pour la combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ et qui sont hors base.

4.2 Méthode pour le calcul de $|S_k^{cu}|$

Maintenant, nous présentons une méthode qui calcule le nombre d'horaires réalisables pour une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ du PLNE (P^u). Nous commençons par décrire cet algorithme, qui est noté PA_{ϕ_3}, et ensuite nous expliquons comment il est possible de l'utiliser dans la méthode de branch and price.

4.2.1 Description de PA_{ϕ_3}

Cet algorithme est entièrement inspiré de PA_{ϕ_2}. Par contre, contrairement à ce dernier, PA_{ϕ_3} n'a pas besoin de connaître la valeur des variables duales de l'itération

Algorithme 4.1 Description formelle de PA_{ϕ_2} appliqué au graphe $\hat{G}^{cu}_{ki}$.

$PA_{\phi_2}\ (\hat{G}^{cu}_{ki},\ x_0) \Longrightarrow PA_{\phi_2}(\hat{G}^{cu}_{ki}) \subseteq S^{cu}_k$:

si $v_0 \cdot x_0 \notin \hat{V}_k$ **alors**
  Terminer
sinon
  $c(v_0 \cdot x_0) \leftarrow 0$
  pour tout $(v_0, x'_0) \in \hat{V}_k \setminus \{v_0 \cdot x_0\}$ **effectuer**
    $c(v_0, x'_0) \leftarrow +\infty$
pour $v_j = v_1 \ldots v_n$ **effectuer**
  pour tout $v_j \cdot x_j \in \hat{V}_k$ **effectuer**
    $c(v_j \cdot x_j) \leftarrow +\infty$
    pour tout $v_i \cdot x_i \in \hat{V}_k$ tel que $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$ **effectuer**
      si $c(v_i \cdot x_i) + c_{ij} < c(v_j \cdot x_j)$ **alors**
        $c(v_j \cdot x_j) \leftarrow c(v_i \cdot x_i) + c_{ij}$
        $p(v_j \cdot x_j) \leftarrow v_i \cdot x_i$

pour tout $v_n \cdot x_n \in \hat{V}_k$ tel que $c(v_n \cdot x_n) < 0$ **effectuer**
  À l'aide de la définition 4.4, construire le chemin $p \in P^{cu}_k(v_n \cdot x_n)$ qui est de coût optimal $c(v_n \cdot x_n)$.
  Ajouter s à PA_{ϕ_2}, où $s \in S^{cu}_k$ est l'unique horaire associé au chemin p.

courante i de GC et sera donc appliqué sur un graphe d'états noté $\hat{G}_k^{cu}$. Ce graphe est défini par le graphe d'état $\hat{G}_k$ et par le premier pré-traitement décrit à la section 4.1.3.1.

PA_{ϕ_3} calcule pour chaque noeud $v_j \cdot x_j \in \hat{V}_k$ la valeur $|P_k^{cu}(v_j \cdot x_j)|$. Soit le sous-ensemble de noeuds d'état $V(v_j \cdot x_j) = \{v_i \cdot x_i \in \hat{V}_k : (v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k,\ c_{ij} < +\infty\}$. $|P_k^{cu}(v_j \cdot x_j)|$ est définie par l'équation récurrente suivante :

$$|P_k^{cu}(v_j \cdot x_j)| = \begin{cases} 1 & \text{si } v_j \cdot x_j = v_0 \cdot x_0, \\ \sum\limits_{v_i \cdot x_i \ \in \ V(v_j \cdot x_j)} |P_k^{cu}(v_i \cdot x_i)| & \text{sinon.} \end{cases} \tag{4.4}$$

PA_{ϕ_3} est formellement décrit par l'algorithme 4.2. Tout d'abord, le nombre de sous-chemin réalisable $v_0 \cdot x_0 \rightsquigarrow v_0 \cdot x_0$ est initialisé à 1, où $v_0 \cdot x_0$ est l'unique noeud de départ et est présenté à la section 4.1.3.2. Ensuite, pour chaque noeud $v_j \cdot x_j \in \hat{V}_k \backslash \{v_0 \cdot x_0\}$, le nombre total de sous-chemins $v_0 \cdot x_0 \overset{p'}{\rightsquigarrow} v_j \cdot x_j$, où $p' \in P_k^{cu}(v_j \cdot x_j)$, est égal au nombre total de sous-chemin $v_0 \cdot x_0 \overset{p''}{\rightsquigarrow} v_i \cdot x_i$, où $p'' \in P_k^{cu}(v_i \cdot x_i)$, de chaque noeud $v_i \cdot x_i \in \hat{V}_k$ pour lequel l'arc $(v_i \cdot x_i,\ v_j \cdot x_j)$ existe et peut être traversé sans être en contradiction avec la définition de la combinaison (k, c) (si $c_{ij} < +\infty$). Enfin, les cardinalités $|P_k^{cu}|$ et $|S_k^{cu}|$ sont définies ci-dessous.

$$|P_k^{cu}| = \sum_{v_n \cdot x_n \in \hat{V}_k} |P_k^{cu}(v_n \cdot x_n)| \tag{4.5}$$

$$|S_k^{cu}| = |P_k^{cu}| \tag{4.6}$$

Finalement, le nombre d'horaires réalisables pour une infirmière $k \in \mathcal{K}$ est défini par $|S_k| = |S_k^{\circ u^0}|$, où $\circ$ représente une combinaison nulle et u^0 le noeud racine de l'arbre de recherche. Cette valeur est alors calculée par $PA_{\phi_3}(\hat{G}_k^{\circ u^0})$, où $\hat{G}_k^{\circ u^0}$ n'est rien de plus que le graphe d'état initial $\hat{G}_k$.

4.2.2 Utilisation de PA_{ϕ_3}

Avant de débuter l'algorithme de branch and price, les cardinalités $|S_k|$ de chaque infirmière $k \in \mathcal{K}$ sont définies par $|S_k| \leftarrow PA_{\phi_3}(\hat{G}_k)$. Pour un noeud $u \neq u^0$ de l'arbre de recherche, où v est son noeud père et $k' \in \mathcal{K}$ est l'infirmière du branchement appliqué à ce noeud, les cardinalités $|S_k^{cu}|$ qui doivent être recalculées sont seulement celles des combinaisons $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ tel que $k = k'$. Ici, afin de simplifier la compréhension de cette section, nous supposons qu'aucune déduction logique n'a été appliquée à travers l'algorithme de branch and price.

Nous avons expliqué à quel moment la cardinalité $|S_k^{cu}|$ d'une combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ doit être définie et redéfinie par PA_{ϕ_3} $(\hat{G}_k^{cu})$. Nous allons maintenant décrire comment il est possible d'utiliser ces cardinalités dans l'algorithme de branch and price.

Premièrement, les cardinalités $|S_k|$ de chaque infirmière $k \in \mathcal{K}$ peuvent être utilisées pour mesurer empiriquement la difficulté d'un problème NSP1. En effet, le temps requis par l'algorithme du branch and price pour calculer la valeur optimale z^* devrait directement être affecté par le nombre d'horaires réalisables pour chaque infirmière. Aussi, pour les infirmières $k \in \mathcal{K}$ qui ont un nombre relativement faible d'horaires réalisables, où $|S_k| < \eta$, l'ensemble S_k peut être calculé avant de débuter l'algorithme de branch and price. D'ailleurs, l'algorithme utilisé pour définir tous les horaires $s \in S_k$ d'une infirmière $k \in \mathcal{K}$ est très similaire à PA_{ϕ_2} et requiert aussi l'application du pré-traitement PA_{ϕ_1} pour définir le graphe d'états $\hat{G}_k$.

Ensuite, les cardinalités $|S_k^{cu}|$ des combinaisons $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ au problème (P^u) peuvent être utilisées pour éviter d'appliquer inutilement $PA_{\phi_2}(\hat{G}_{ki}^{cu})$ lors de l'itération i de GC. Effectivement, lorsque $|S_k^{cu}| = |S_k^{cu} \bigcap S_k^{ui}|$, nous avons assurément $PA_{\phi_2}(\hat{G}_{ki}^{cu}) = \emptyset$.

Troisièmement, ces mêmes cardinalités peuvent aussi être exploitées dans la méthode de Pricing afin de définir des stratégies de générations de colonnes lors de l'itération i de GC. Considérons deux combinaisons différentes (k', c') et $(k'', c'') \in \mathcal{K}_{\mathcal{I}_k^u}$ au problème (P^u), où $|S_{k'}^{c'u}| - |S_{k'}^{c'u} \bigcap S_{k'}^{ui}| \gg |S_{k''}^{c''u}| - |S_{k''}^{c''u} \bigcap S_{k''}^{ui}|$. Une telle stratégie pourrait être d'appliquer $PA_{\phi_2}(\hat{G}_{k'i}^{c'u})$ avant $PA_{\phi_2}(\hat{G}_{k''i}^{c''u})$. Car, ces ensembles sont probablement tels que $PA_{\phi_2}(\hat{G}_{k'i}^{c'u}) \gg PA_{\phi_2}(\hat{G}_{k''i}^{c''u})$.

Algorithme 4.2 Description formelle de PA_{ϕ_3} appliqué au graphe $\hat{G}_k^{cu}$.

$PA_{\phi_3}\ (\hat{G}_k^{cu},\ x_0) \Longrightarrow |P_k^{cu}|$ **est calculé :**

$|P_k^{cu}| \leftarrow 0$
si $v_0 \cdot x_0 \notin \hat{V}_k$ **alors**
  Terminer
sinon
  $|P_k^{cu}(v_0 \cdot x_0)| \leftarrow 1$
  pour tout $(v_0, x_0') \in \hat{V}_k \setminus \{v_0 \cdot x_0\}$ **effectuer**
    $|P_k^{cu}(v_0 \cdot x_0')| \leftarrow 0$

pour $v_j = v_1 \ldots v_n$ **effectuer**
  pour tout $v_j \cdot x_j \in \hat{V}_k$ **effectuer**
    $|P_k^{cu}(v_j \cdot x_j)| \leftarrow 0$
    pour tout $v_i \cdot x_i \in \hat{V}_k$ tq. $(v_i \cdot x_i,\ v_j \cdot x_j) \in \hat{E}_k$ et $c_{ij} < +\infty$ **effectuer**
      $|P_k^{cu}(v_j \cdot x_j)| \leftarrow |P_k^{cu}(v_j \cdot x_j)| + |P_k^{cu}(v_i \cdot x_i)|$

pour tout $v_n \cdot x_n \in \hat{V}_k$ **effectuer**
  $|P_k^{cu}| \leftarrow |P_k^{cu}| + |P_k^{cu}(v_n \cdot x_n)|$

4.3 Modification du graphe G_k pour le problème NSP2

Maintenant, nous allons décrire les modifications à apporter au graphe $G_k = (V_k, E_k)$ d'une infirmière $k \in \mathcal{K}$ pour la résolution du problème NSP2, qui a été présenté à la section 1.5.

Premièrement, rappelons en quoi consiste le graphe G_k pour le problème NSP1. Il modélise avec deux dimensions (jour de l'horizon et quart de travail) l'horizon de planification. Chaque noeud $v \in V_k$ représente une affectation $(d,t) \in \mathcal{D}_{\mathcal{T}_k}$ et est en bijection avec la variable a_{ksdt} d'un horaire $s \in S_k$. Dans la modélisation du problème NSP2, la variable a_{ksdt} a été remplacée par plusieurs variables a_{ksdtL} afin de déterminer dans laquelle des contraintes de quotas (Q_L), où $L \in \mathcal{L}_{dt}$, l'affectation (d,t) aura un apport [10]. Étant donné que les variables a_{ksdtL} sont en réalitées des constantes de notre modèle mathématique et des variables du problème auxiliaire, il est alors nécessaire de modifier le graphe $G_k = (V_k, E_k)$ afin de définir ces variables.

Le nouveau graphe $G_k = (V_k, E_k)$ représente toujours l'horizon de planification, mais avec 3 dimensions : jour de l'horizon, quart de travail et ensemble de compétences. Chaque noeud $v \in V_k$ est défini par $v = (d,t,L)$, où $(d,t) \in \mathcal{D}_{\mathcal{T}_k}$ et $L \in \mathcal{L}_{dt}$, et est en bijection avec la variable a_{ksdtL} d'un horaire $s \in S_k$. En plus des conditions décrites à la section 4.1.1.1 sur l'existence d'un noeud $v \in V_k$, un tel noeud est créé si l'infirmière k a au moins une compétence qui lui permet d'avoir un apport, grâce à l'affectation (d,t), dans la contrainte de quotas (Q_L) : $\Lambda_k \bigcap L \neq \emptyset$.

Les condition d'existences des arcs sont les mêmes que pour le graphe du problème NSP1. En effet, l'ajout d'une dimension pour les ensembles de compétences ne perturbe pas la définition de l'ensemble d'arcs E_k.

[10] La variable a_{ksdtL} est définie à la section 1.5.3.

Enfin, la figure 4.4 représente bien le graphe $G_k = (V_k, E_k)$ d'une infirmière $k \in \mathcal{K}$ pour le problème NSP2. On peut remarquer qu'une affectation $(d, t) \in \mathcal{D}_{\mathcal{T}_k}$ commune à deux noeuds v' et $v'' \in V_k$, respectivement associés aux variables $a_{ksdtL'}$ et $a_{ksdtL'}$ (où $L', L'' \in \mathcal{L}_{dt}$ et $L' \neq L''$), n'a pas été dédoublée inutilement.

Finalement, les autres composantes du problème auxiliaire, soit les algorithmes PA_{ϕ_1} et PA_{ϕ_2} et le graphe d'états $\hat{G}_k$, ne nécessitent aucune modification pour la résolution du problème NSP2. Pour ce qui est du graphe $\hat{G}^{cu}_{ki}$, les pré-traitements à appliquer pour sa définition sont similaires à ceux décrits à la section 4.1.3.1.

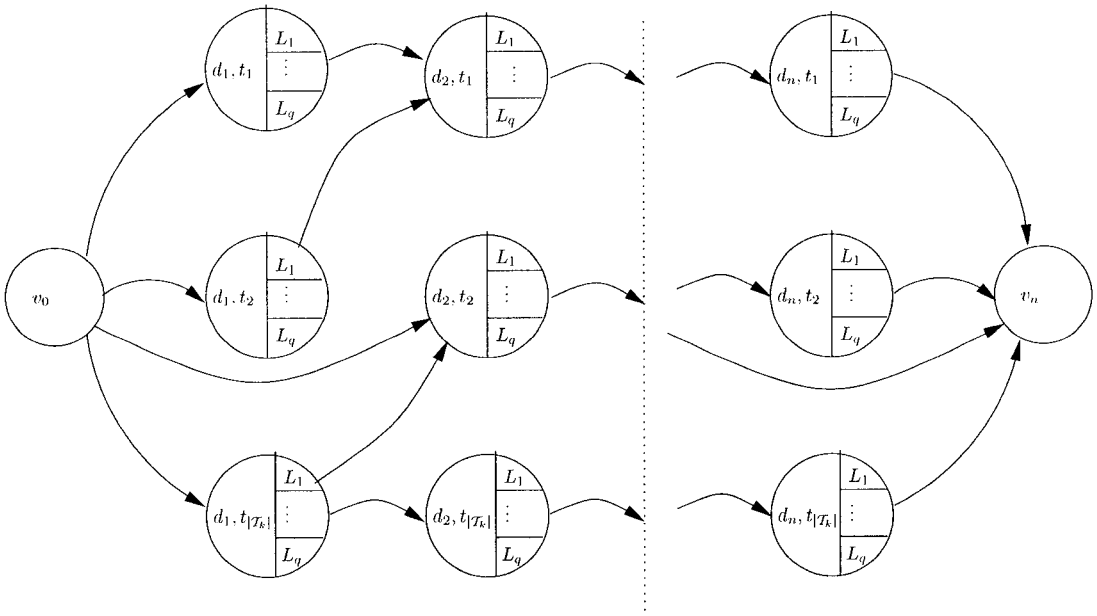


Figure 4.4: Le graphe $G_k = (V_k, E_k)$ pour le problème NSP2, où $q = |\mathcal{L}_{dp}|$ pour $(d, p) \in \mathcal{D}_{\mathcal{P}}$.

4.4 Alternative au concept RFRC

Nous montrerons au chapitre 5, lors de la présentation des résultats obtenus, que l'utilisation du concept RFRC pour respecter simultanément toutes les contraintes de branchement Ryan-Foster n'a pas porté fruit. En outre, nous allons conclure en affirmant que l'application de branchement Ryan-Foster, jumelé avec ce concept, n'est pas efficace à l'intérieur d'un algorithme de branch and price. Par conséquent, il serait intéressant de trouver une alternative au concept RFRC.

Une approche possible consiste à incorporer le respect simultané de toutes les contraintes de branchement Ryan-Foster à l'intérieur même du problème auxiliaire. Ceci est possible par la modélisation du respect d'une contrainte $(d_i, t_i, d_j, t_j) \in \mathcal{C}_k^u$ à l'aide d'une nouvelle ressource $r \in \mathcal{R}$. La modélisation d'une telle ressource serait simple.

Toutefois, l'inconvénient majeur résiderait dans l'obligation de rappliquer l'algorithme $PA_{\phi_1}(G_k)$ à chaque fois qu'une règle de branchement est appliquée au noeud u et pour une infirmière $k \in \mathcal{K}$. Ceci, pour les deux nouveaux PLNE (P^v) et (P^w). De plus, étant donné que chaque contrainte $(d_i, t_i, d_j, t_j) \in \mathcal{C}_k^u$ serait modélisée par une ressource $r \in \mathcal{R}$, le nombre total de ses ressources ($|\mathcal{R}|$) serait proportionnel à la cardinalité $|\mathcal{C}_k^u|$. Ceci pourrait avoir la fâcheuse conséquence d'augmenter le temps nécessaire au traitement de la phase PA_{ϕ_1} de façon proportionnel au nombre de contraintes de branchement Ryan-Foster. Tout de même, un avantage à ne pas oublier est que le nombre de valeurs possibles pour chacune de ces ressources serait seulement de deux ; une pour chacune des deux façons de respecter la contrainte.

CHAPITRE 5 : RÉSULTATS DE CALCUL

Cette section est entièrement consacrée à la présentation des résultats obtenus par le programme Iris. Nous décrivons comment ce logiciel a été développé, les jeux de données utilisés pour les tests et les résultats obtenus. En plus, nous analysons ces résultats afin de comparer les différentes stratégies de l'algorithme de branch and price.

5.1 Ressources informatiques utilisées

Le programme Iris a été développé et testé sur le système d'exploitation Red Hat Linux. Il a été implémenté avec le langage de programmation C++ et à l'aide du compilateur egcs++. Nous pouvons voir ci-dessous quelques caractéristiques de la machine sur laquelle nous avons faits tourné les tests :

- processeur Intel Pentium IV cadencé à 2.8 GHz,
- 1.5 Go de mémoire vive,
- système d'exploitation Red Hat Linux 9.0.

Deux outils d'optimisation ont été utilisés afin de modéliser et résoudre le problème de génération automatisée d'horaires d'infirmières (NSP1). Ces deux outils sont Concert Technology version 1.0 [25] et Cplex version 7.0 [21], tous les deux étant des produits de ILOG.

Concert Technology est une interface (API) au produit CPLEX, qui lui est l'implémentation choisie de l'algorithme du Simplexe. Cet API permet de définir le modèle

mathématique qui sera résolu par cet algorithme d'optimisation. Mais en plus, il offre la possibilité d'apporter une ou plusieurs modifications à ce modèle entre deux appels successifs à l'algorithme du Simplexe. Les modifications possibles sont les suivantes :

- l'ajout et le retrait d'une ou plusieurs colonnes ou contraintes au modèle,
- modification d'un ou plusieurs attributs propres à une colonne ou à une contrainte. Par exemple, les attributs d'une variable sont son coefficient dans la fonction objectif et ses bornes inférieure et supérieure.

5.2 Présentation des jeux de données

Maintenant, nous allons présenter les 3 jeux de données (ou fichier) utilisés pour tester le programme Iris. Deux de ces jeux de données proviennent du "Centre Universitaire de Santé McGill" (CUSM) et le troisième provient de l'urgence de l'hôpital Santa Gabrini, mais il s'applique aux médecins. Voici la provenance de ces fichiers :

- centre des naissances du CUSM,
- l'urgence du CUSM,
- l'urgence de l'hôpital Santa Gabrini (médecins).

Nous décrivons chacun de ces jeux de données à l'aide de deux tableaux. Le premier tableau indique la taille de l'horizon, le nombre d'infirmières (ou de médecins) de l'unité, le nombre de périodes et le nombre de quarts. Le deuxième décrit de façon détaillée toutes les contraintes de quotas à respecter par l'unité considérée. Pour avoir une description exacte du format d'un fichier d'entrée fourni au logiciel Iris, nous référons le lecteur à Galinier, Jaumard et Labit [26].

Enfin, les résultats obtenus par Iris sur ces 3 jeux de données et l'analyse de chacun de ces résultats seront présentés à la prochaine section.

5.2.1 Données du centre des naissances

Ce premier fichier exploite presque toutes les possibilités de la définition des problèmes qui sont acceptés par Iris. En effet, certaines périodes sont couvertes par plus d'un quart, certains quart couvrent plus qu'une période, les contraintes de quotas ont des déficits et des surplus non nuls, etc. Par contre, toutes ces contraintes sont définies sur le même ensemble de compétences (voir le tableau 5.2) [1].

Enfin, il est important de préciser que toutes les infirmières du présent jeux de données ne possèdent ni préférence ni aversion. Nous verrons l'impact de ceci lors de la présentation des résultats. De plus, la relation entre les quarts et les périodes de cette unité est définie au tableau A.1.

Tableau 5.1: Données générales du centre des naissances.

Taille de l'horizon en semaines	6
Nombre d'infirmières	41
Nombre de périodes	7
Nombre de quarts de travail	5
Nombre de contraintes de quotas	49

5.2.2 Données de l'urgence

Ce jeux de données est le deuxième du CUSM que nous présentons. Comme pour le précédent, la taille de l'horizon, le nombre de quarts de travail et le nombre d'infirmières représentent de façon empirique la grandeur type des problèmes auquels Iris a

[1] $\ell_1 - \ell_6$ correspond aux compétences $\ell_1, \ell_2, \ldots, \ell_6$.

Tableau 5.2: Description des contraintes de quotas du centre des naissances.

Jour	Période	Compétence	Quota cible	Déficit max.	Surplus max.
Lu. au Di.	p_1, p_2	$\ell_1 - \ell_6$	13	6	7
Lu. au Di.	p_3	$\ell_1 - \ell_6$	3	2	2
Lu. au Di.	p_4, p_5	$\ell_1 - \ell_6$	3	2	3
Lu. au Di.	p_6	$\ell_1 - \ell_6$	5	4	5
Lu. au Di.	p_7	$\ell_1 - \ell_6$	4	3	4

été utilisés afin de les résoudre. La relation entre les quarts et les périodes est définie au tableau A.2.

Enfin, comparativent aux infirmières du centre des naissances, la plupart des infirmières du présent jeux de données possèdent de 20 à 30 préférences ou aversions.

Tableau 5.3: Données générales de l'urgence.

Taille de l'horizon en semaines	6
Nombre d'infirmières	41
Nombre de périodes	7
Nombre de quarts de travail	4
Nombre de contraintes de quotas	49

Tableau 5.4: Description des contraintes de quotas de l'urgence.

Jour	Période	Compétence	Quota cible	Déficit max.	Surplus max.
Lu. au Di.	$p_1 - p_6$	$\ell_2 - \ell_5$	8	4	4
Lu. au Di.	p_7	$\ell_2 - \ell_5$	4	3	5

5.2.3 Données de l'urgence de Santa Gabrini (médecins)

Le présent jeux de données a été fourni par l'urgence de l'hôpital Santa Gabrini et il s'applique aux médecins de cette unité, contrairement aux deux précédents jeux de données où les employés considérés étaient des infirmières. Toutefois, ce jeux de données peut être traité par Iris car il demeure dans les balises de la définition des problèmes pouvant être résolus par ce programme.

Nous montrons ci-dessous les principales particularités du présent jeux de données, qui s'applique aux médecins, afin de bien comprendre les différences entre un tel jeux de données et un jeux de données qui s'applique aux infirmières.

1. Tous les quarts de travail couvrent une seule période et toutes les périodes sont couvertes par un seul quart. En d'autres termes, la notion de période est désuete.
2. Toutes les contraintes de quotas ont un quota cible de 1 et n'acceptent ni déficit ni surplus.
3. Tous les médecins n'ont aucune préférence ni aversion.

Par cause des deux dernières particularités mentionnées ci-haut, il est important de noter au lecteur que tous les horaires généraux réalisables du présent problème ont tous une valeur identique de 0. Une autre différence entre ce jeux de données et les précédents est la grandeur du problème à résoudre. En effet, celui ci est de grandeur empirique inférieure car il possède 17 médecins, soit moins de la moitié du nombre d'infirmières des autres problèmes, et la taille de l'horizon est de 4 semaines, contrairement à 6 pour les autres problèmes.

Nous pouvons voir au tableau 5.6 toutes les contraintes de quotas du présent jeux de données. Il est important de préciser que les contraintes de quotas définies à tous

Tableau 5.5: Données générales du problème de médecin.

Taille de l'horizon en semaines	4
Nombre de médecins	17
Nombre de périodes	8
Nombre de quarts de travail	8
Nombre de contraintes de quotas	56

Tableau 5.6: Description des contraintes de quotas du problème de médecin.

Jour	Période	Compétence	Quota cible	Déf. max.	Surplus max.
Lu. au Di.	$p_1 - p_5, p_7, p_8$	ℓ_1, ℓ_2	1	0	0
Lu. au Ve.	p_6	ℓ_1, ℓ_2	1	0	0
Sa. et Di.	p_6	ℓ_1, ℓ_2	0	0	0

les samedi et à tous les dimanche de l'horizon et à la période p_6 n'apportent aucune difficulté au problème. Ceci est dû aux deux assignations interdites à tous les 17 médecins. En effet, ces deux contraintes, qui sont intrinsèques à tous les médecins et qui sont définies à même les données, portent sur tous les samedi et tous les dimanches de l'horizon et sur l'unique quart de travail qui couvre la période p_6. Par conséquent, tous les horaires de tous les médecins n'auront aucune de ces deux assignations.

Finalement, après avoir décrit les 3 jeux de données utilisés pour tester Iris, nous allons maintenant montrer les résultats obtenus par ce programme et l'analyse que nous avons faits sur ces résultats.

5.3 Résultats et analyse

Au chapitre 2, nous avons présentés les deux parcours possibles de l'arbre de recherche et, au chapitre 3, nous avons définis les deux règles de branchement qui peuvent être utilisées lors de ces deux algorithmes. Ces parcours sont le DFS et le BSFS et ces deux méthodes de séparation sont le branchement binaire et le branchement Ryan-Foster.

De plus, à la section 2.6.1, nous avons expliqués comment obtenir un arbre de recherche exhaustif lorsque les deux méthodes de séparation sont permises, soit grâce à l'application d'un branchement binaire lorsqu'aucun branchement Ryan-Foster n'est possible. Alors, nous obtenons les 4 versions décrites ci-dessous pour notre algorithme de branch and price.

1. DFS-1 : parcours DFS avec l'application de branchements binaire seulement.
2. DFS-2 : parcours DFS avec l'application de branchements Ryan-Foster et de branchements binaire.
3. BSFS-1 : parcours BSFS avec l'application de branchements binaire seulement.
4. BSFS-2 : parcours BSFS avec l'application de branchements Ryan-Foster et de branchements binaire.

Premièrement, pour chacun des 3 jeux de données présentés à la section précédente, nous allons décrire les résultats que nous avons obtenus par l'une des 4 versions de l'algorithme de branch and price énumérées ci-dessus. Aussi, nous allons donner la ou les raisons pour lesquelles cette version de Iris a été choisie pour résoudre le jeux de données considéré.

Ensuite, pour chacun des jeux de données, nous allons présenter les résultats obtenus à l'aide de deux tableaux.

Le premier tableau donne un aperçu de l'arbre de recherche résultant de l'algorithme de branch and price. Il est composé des données décrites ci-dessous.

1. Le nombre de noeuds
   - explorés (pour lesquels au moins un des deux fils a été défini et résolu),
   - coupés (voir les conditions décrites à la section 2.6.2),
   - feuilles (pour lesquels la solution optimale est entière) [2],
   - total.
2. Le nombre de branchements binaire appliqués (idem pour Ryan-Foster).

Le deuxième tableau donne quelques informations sur la première solution trouvée et quelques-unes des meilleures solutions trouvées durant l'exécution du programme Iris. Pour chaque solution, les informations suivantes sont données :

1. son ordre (première, deuxième, etc.),
2. la profondeur et l'ordre du noeud où elle a été trouvée,
3. le temps durant lequel elle a été trouvée,
4. sa valeur,
5. si elle est entière (oui ou non),
6. si c'est une solution optimale (oui ou non).

De plus, nous donnons la valeur de la relaxation continue du noeud racine, notée par $z^*_{u^0}$. Rappelons que $\lceil z^*_{u^0} \rceil$ est une borne inférieure sur la valeur optimale du problème (voir la section 2.6.2).

Pour chaque exécution du programme Iris, un temps maximal de 8 heures est imposé. Ce temps inclus le pré-traitement nécessaire à la construction des graphes

[2] À ne pas confondre avec la définition d'un noeud feuille de la section 2.6.1.

G_k et à leur application à $PA_{\phi_1}(G_k)$. Pour les 3 jeux de données, le temps requis pour ce pré-traitement est d'environ 8 minutes. Notons aussi que la profondeur et l'ordre du noeud racine, noté par u^0, sont tous les deux de valeur 0. De plus, précisons qu'une solution qui n'est pas entière est générée par la méthode d'arrondi (voir la section 2.5.2).

5.3.1 Données du centre des naissances

Ce jeux de données a été utilisé afin de comparer les 4 versions DFS-1, DFS-2, BSFS-1 et BSFS-2 du programme Iris. Les meilleurs résultats ont été obtenus par la version BSFS-1, qui est d'ailleurs la seule à avoir trouvé une solution de valeur optimale et à avoir prouvé cette optimalité à l'intérieur du délai de 480 minutes (8 heures), soit en un temps total de 225 minutes. Au tableau 5.7, nous pouvons voir pour chacune des 4 versions leur meilleure solution trouvée et leur temps total.

Tableau 5.7: Comparaison de la meilleure solution trouvée et du temps total de chacune des 4 versions de Iris.

Version	Meilleure solution trouvée				temps
de Iris	temps	valeur	entière	optimale	total
DFS-1	303	186	oui	non	480
DFS-2	3	211	non	non	480
BSFS-1	225	176	oui	oui	225
BSFS-2	461	181	non	non	480

Maintenant, nous allons avancer deux hypothèses afin d'expliquer pourquoi la version BSFS-1 prouve l'optimalité de la valeur 176 en seulement 225 minutes alors que les 3 autres versions n'ont trouvée aucune solution de cette valeur après un temps total de 480 minutes. Afin de nous aider dans cette démarche, nous donnons

au tableau 5.8 quelques données sur les noeuds et les branchements de l'arbre de recherche de chacune des 4 versions de Iris.

Tableau 5.8: Données des arbres obtenus par les 4 versions de Iris.

Version	Nombre de noeuds				Nombre de branchements	
de Iris	explorés,	coupés,	feuilles,	total.	binaire	Ryan-Foster
DFS-1	1270	1165	11	2447	2396	-
DFS-2	167	36	0	204	0	203
BSFS-1	589	580	10	1179	1168	-
BSFS-2	154	0	0	308	0	307

Premièrement, il est intéressant d'observer que le nombre total de noeuds de l'arbre de recherche de chacune des deux versions DFS-2 et BSFS-2 est de beaucoup inférieur au nombre total de noeuds de l'arbre des deux versions DFS-1 et BSFS-1. De plus, ces deux premières versions n'ont atteint aucun noeud feuille. Ceci est certainement causé par le nombre de combinaison $(k, c) \in \mathcal{K}_{\mathcal{I}_k^u}$ qui est nécessaire de traiter lors de la génération de colonnes CR^- hors bases et qui est égal à $|\mathcal{K}_{\mathcal{I}_k^u}|$ lorsque l'algorithme GC_ϕ prouve l'optimalité de la valeur courante $\dot{z}_u^i$, voir la section 2.3.2.3. De plus, rappelons au lecteur que le nombre de combinaisons propres à une infirmière $k \in \mathcal{K}$ et possibles au noeud u est égal à $|\mathcal{I}_k^u|$ et croît de façon exponentielle selon $|C_k^u|$, qui est le nombre de contraintes de branchement Ryan-Foster définies pour cette même paire d'infirmière et de noeud.

Par conséquent, pour les deux versions DFS-2 et BSFS-2 qui n'ont utilisé que des branchements Ryan-Foster, la relaxation continue (R^u) de chaque noeud u de l'arbre de recherche requiert plus de temps CPU par cause du nombre total de combinaisons qui croît exponentiellement selon les cardinalités $|C_k^u|$ des infirmières $k \in \mathcal{K}$. Il est donc normal de conclure que notre algorithme de branch and price est plus efficace lorsqu'il n'applique que des branchements binaires.

Ensuite, la comparaison des résultats obtenus par les deux versions DFS-1 et BSFS-1, qui n'ont utilisé que des branchements binaires, montre clairement que le parcours BSFS est plus efficace que le parcours DFS. Alors, on peut conclure que dans le parcours BSFS le temps CPU requis pour déterminer lequel des deux fils est le plus prometteur a été profitable pour la recherche d'une solution de valeur optimale.

Aussi, il est important de préciser que la valeur optimale $z^*_{u^0}$ de la relaxation continue (R^{u^0}) est égale à 175.5. Rappelons au lecteur que $\lceil z^*_{u^0} \rceil = 176$ est une borne inférieure sur la valeur optimale de notre problème. C'est d'ailleurs grâce à l'utilisation de cette valeur que la version BSFS-1 de l'algorithme de branch and price conclu de façon immédiate que la solution de valeur 176 est optimale.

Finalement, nous pouvons voir au tableau 5.9 une description de la première et des 4 dernières solutions trouvées par la version BSFS-1 de l'algorithme de branch and price pour le présent jeux de données. Il est intéressant d'observer que seulement 84 minutes ont suffi pour trouver une solution de valeur 177 alors que 225 minutes ont été nécessaires pour atteindre une solution de valeur optimale 176, soit une différence de 141 minutes. Aussi, l'arbre de recherche obtenu par la version BSFS-1 est donné à la figure B.2.

5.3.2 Données de l'urgence

Comme le précédent jeux de données, celui de l'urgence a été testé par les 4 versions DFS-1, DFS-2, BSFS-1 et BSFS-2 du programme Iris. Par contre, contrairement au jeux de données du centre des naissances, toutes ces versions ont obtenu des résultats relativement semblable à l'intérieur d'un délai de 480 minutes (28800

Tableau 5.9: La première et les 4 dernières solutions obtenues par la version BSFS-1 de Iris.

Ordre de la solution	Noeud Prof.	Noeud Ordre	Temps requis (min.)	Valeur	Entière	Optimale
1	0	0	4	248	non	non
16	112	241	10	180	oui	non
17	97	413	56	179	oui	non
18	107	585	84	177	oui	non
19	108	1061	225	176	oui	oui

sec.). De façon plus précise, aucune de ces versions n'a trouvé une solution de valeur optimale, toutes les solutions qu'elles ont générées ont été trouvées par la méthode d'arrondi, qui rappelons le relève du hasard, et leur meilleure solution sont toutes de valeur relativement proche.

Par conséquent, nous avons choisi de façon arbitraire de comparer les données de l'arbre de recherche obtenu par la version BSFS-1 de Iris, voir le tableau 5.10, avec celui obtenu par la même version mais sur le jeux de données du centre des naissances, voir le tableau 5.8. Nous pouvons remarquer que le nombre total de noeuds créés et résolus (et le temps total) par la version BSFS-1 sur les jeux de données du centre des naissances et de l'urgence sont respectivement égale à 1179 (en 13 500 sec.) et 49 (en 28 000 sec.). En d'autres termes, les temps moyens nécessaires pour la résolution d'une relaxation continue (R^u) à un noeud u pour ces deux jeux de données sont respectivement égale à 11 et 587 secondes.

Alors, la question qui se pose à nos yeux est *"pourquoi les relaxations continues du jeux de données de l'urgence sont-elles plus difficiles à résoudre que celles du jeux de données du centre des naissances ?"*. Pour répondre à cette question, nous

Tableau 5.10: Données de l'arbre obtenu par la version BSFS-1 de Iris.

Nombre de noeuds				Nombre de branchements		$z^*_{u^0}$
explorés,	coupés,	feuilles,	total.	binaire	Ryan-Foster	
24	0	0	49	48	-	2677

devons faire un retour sur le contenu de ces deux jeux de données afin de présenter la différence décrite ci-dessous.

- Pour le centre des naissances, toutes les infirmières ne possèdent ni préférence ni aversion.
- Pour l'urgence, presque toutes les infirmières ont de 20 à 30 préférences ou aversions.

Grâce à cette différence majeure entre les deux jeux de données, on peut alors se permettre d'affirmer l'hypothèse suivante concernant une infirmière k_1 de l'unité d'urgence en comparaison avec une infirmière k_2 du centre des naissances : proportionnellement au nombre d'horaires réalisables pour les infirmières k_1 et k_2, qui sont respectivement représentés par les cardinalités $|S_{k_1}|$ et $|S_{k_2}|$, l'infirmière k_2 devrait avoir un plus grand nombre de paires d'horaires de poids identiques que l'infirmière k_1.

Ensuite, considérons les relaxations continues (R^u) du noeud u et (R^v) du noeud v, où u est le noeud père de v et $k \in \mathcal{K}$ est l'infirmière du branchement appliqué au noeud u. De plus, rappelons que cette méthode de séparation consiste à retirer un sous-ensemble de variables appartenant à l'infirmière k de la solution optimale x^n_u du problème (R^u). Alors, si le jeux de données considéré est celui du centre des naissances, peu d'itérations de la méthode de génération de colonnes seront nécessaires afin de trouver des variables CR^- hors base qui vont remplacer celles précédemment retirées, car les variables de l'infirmière k ont des poids similaires et elles sont donc

facilement remplaçables. Par contre, si le jeux de données est celui de l'urgence, la situation inverse se produit étant donné la disparité du poids des horaires. Enfin, ceci explique pourquoi la résolution d'une relaxation continue du jeux de données du centre des naissances requiert moins de temps CPU que la résolution d'une relaxation continue du jeux de données de l'urgence.

Finalement, nous pouvons voir au tableau 5.11 toutes les solutions obtenues par la version BSFS-1 de Iris appliquées aux jeux de données de l'urgence. Remarquons qu'aucune de ces 5 solutions n'est entière et que leur valeur est très loin de la borne inférieure $z^*_{u^0} = 2677$. L'arbre de recherche associé à ces résultats est donné à la figure B.1.

Tableau 5.11: Solutions obtenues par la version BSFS-1 de Iris.

Ordre de la solution	Noeud		Temps requis (min.)	Valeur	Entière	Optimale
	Prof.	Ordre				
1	0	0	51	10370	non	non
2	1	1	82	9531	non	non
3	2	3	100	9149	non	non
4	2	4	110	6645	non	non
5	11	21	257	6640	non	non

5.3.3 Données pour le problème des médecins

Les derniers résultats présentés dans ce mémoire sont ceux obtenus sur le jeu de données des médecins de l'urgence de l'hôpital Santa Gabrini. De tous les problèmes auquel Iris a été confronté, le présent problème a été celui pour lequel ce programme a eu le plus de difficulté à trouver une première solution.

Premièrement, nous avons vu lors de la description du présent jeu de données que toutes ses solutions sont de valeur identiques. D'ailleurs, cette particularité propre à ce problème aide à sa résolution. En effet, étant donné que la valeur optimale $z^*_{u^0}$ de la relaxation continue (R^{u^0}) et la valeur de toutes les solutions sont égales à 0, le problème est résolu dès qu'une première solution est trouvée car elle est obligatoirement de valeur optimale. De plus, notons que la valeur optimale z^*_u des relaxations continues (R^u) sont aussi de valeur optimale égale à 0.

Deuxièmement, l'application de la méthode d'arrondi, qui tente de trouver un horaire général réalisable en choisissant au hasard un horaire par infirmière, n'a été d'aucun apport pour le présent jeux de données. Ceci est dû aux contraintes de quotas qui n'acceptent ni déficit ni surplus et qui doivent donc être satisfaites avec une précision extrême.

Ainsi, l'unique parcours testé est le DFS car le choix du meilleur des deux fils lors du BSFS est inutile étant donné qu'ils sont tous les deux de valeur optimale identique. Enfin, le choix entre les deux versions DFS-1 et DFS-2 s'est portée sur la première. La motivation est que l'application seule de branchement binaire permet d'obtenir plus d'une contrainte de branchement grâce aux déductions logiques décrites à la section 3.2.3. En outre, ceci est vrai pour toutes les applications d'un branchement binaire à tous les noeuds de l'arbre de recherche étant donné que toutes les contraintes de quotas ont une cible de 1 et n'acceptent ni déficit ni surplus.

Rappelons que dans la méthode de génération de colonnes en deux phases, la phase $GC_{\phi 1}$ a l'objectif de fournir une première solution réalisable à la relaxation continue (R^u). Toutefois, pour le présent jeu de données, cette solution est non seulement réalisable à ce problème mais elle est aussi une solution de valeur optimale $z^*_u = 0$.

Tableau 5.12: Données de l'arbre obtenu par la version DSF-1 de Iris.

Avec déductions logiques	Nombre de noeuds explorés,	coupés,	feuilles,	total.
oui	57	0	1	58
non	70	0	1	71

Tableau 5.13: Solutions obtenues par la version DFS-1 de Iris.

Avec déductions logiques	Noeud Prof.	Ordre	Temps requis (min.)	Entière
oui	57	57	346	oui
non	70	70	486	oui

Par conséquent, pour toutes les relaxations continues de l'arbre de recherche, aucune application de la phase GC_{ϕ_2} n'est nécessaire à leur résolution.

Finalement, nous pouvons voir aux tableaux 5.12 et 5.13 des statistiques sur l'arbre de recherche et les solutions obtenues par la version DFS-1 de Iris. Nous montrons ces données pour le cas où les déductions logiques ont été appliquées et le cas où aucune déduction n'a été utilisée. Nous pouvons voir l'apport significatif des déductions logiques dans la version DFS-1.

CONCLUSION

Dans ce mémoire, nous avons proposé un algorithme de branch and price pour la résolution du problème de confection d'horaires d'infirmières (NSP). Les contributions de ce mémoire sur le projet Iris peuvent se classer en trois catégories.

Premièrement, lorsqu'on applique un algorithme pour résoudre un problème donné, il est important de montrer la maîtrise totale de cet algorithme et de ses différentes composantes et de faire ressortir toutes les subtilités émergeant de l'interaction entre celles-ci. C'est pourquoi nous avons décrit de façon aussi détaillée et approfondie l'algorithme de branch and price et l'interaction entre ses éléments : la méthode de génération de colonnes et celles du pricing avec la phase 2 du problème auxiliaire, les deux règles de branchement disjointes et complètes qui définissent soit une séparation en base ou à demi en base, le branchement Ryan-Foster et le branchement binaire donnant naissance au nouveau concept RFRC, etc. Autrement dit, après que le lecteur réfléchi ai lu cette description de notre algorithme (chapitre 2), il pourra conclure par lui même que notre méthode et le programme Iris sont exactes.

Deuxièmement, nous avons aussi étudié l'application des deux méthodes de séparation : branchement Ryan-Foster et branchement binaire. Nous avons tout d'abord montré comment il est possible d'améliorer l'efficacité de branchement binaire par l'application de déductions logiques et nous avons montré l'apport de cette amélioration sur le temps CPU du programme Iris. Aussi, le concept RFRC a été défini pour la première fois en plus de l'algorithme nécessaire à son application. Toutefois, nous avons obtenu des résultats très peu encourageants pour l'application de ce concept. En effet, nous avons conclu que l'application de branchements binaires seule est plus

efficace que l'application des deux branchements (binaire et Ryan-Foster) lorsque le concept RFRC est utilisé.

Enfin, le dernier apport théorique est celui du développement de l'algorithme qui calcule le nombre d'horaires réalisables pour une infirmière donnée (PA_{ϕ_3}). Cette contribution n'est certainement pas négligeable étant donné l'aspect combinatoire du NSP. En outre, nous avons donné quelques pistes à suivre pour une exploitation constructive de cet algorithme dans le branch and price.

Finalement, les temps de calcul nécessaires à Iris pour prouver l'optimalité d'une solution ou pour trouver une première solution réalisable sont relativement grands. De plus, d'autres méthodes trouvent une première solution en un temps grandement inférieur à celui requis par Iris. Il ne suffit que de mentionner la méthode de Bourdais [27] qui est une méthode basée sur la programmation par contraintes et qui trouve généralement plusieurs solutions en moins de 100 secondes. Évidemment, Iris ne sera jamais aussi performant par cause du pré-traitement nécessaire à la construction des graphes et à leur application à la phase 1. Toutefois, il est possible d'améliorer Iris, tant pour l'aspect de la preuve de l'optimalité d'une solution que pour trouver la première. Nous décrivons ci-dessous les améliorations possibles pour le développement du projet Iris dans un avenir rapproché.

1. Étudier la possibilité d'intégrer les contraintes de branchement Ryan-Foster dans le problème auxiliaire, sans toutefois devoir ré-appliquer la phase 1, afin d'éliminer le concept RFRC.
2. Développer des stratégies de séparation pour le branchement Ryan-Foster et le branchement binaire.
3. Améliorer la stratégie du branchement prématuré.

4. Améliorer la méthode d'arrondi afin d'obtenir une première solution le plus rapidement possible et en considérant la meilleure solution actuellement connue (borne supérieure).
5. Développer des stratégies de génération de colonnes à l'aide de PA_{ϕ_3}.
6. Utiliser PA_{ϕ_3} pour déterminer s'il est possible et avantageux d'énumérer explicitement tous les horaires réalisables d'une infirmière donnée.

BIBLIOGRAPHIE

[1] VOVOR, T. (1997). *Problèmes de chemins bicritères ou avec contraintes de ressource : algorithmes et applications.* Thèse de doctorat, Ecole Polytechnique de Montréal.

[2] JAUMARD, B., SEMET, F., et VOVOR, T. (1998). Generalized linear programming model for nurse scheduling. *European Journal Of Operational Research*, 107(1) :1–18.

[3] LABIT, P. (2000). IRIS : Amélioration d'une méthode de génération de colonnes pour la confection d'horaires d'infirmières. Master's thesis, École Polytechnique de Montréal.

[4] Hôpital Royal Victoria. Convention collective du personnel infirmier.

[5] JÜNGER, M., et THIENEL, S. (1998). The ABACUS System for Branch-and-Cut-and-Price Algorithms in Integer Programming and Combinatorial Optimization. *Software : Practice and Experience*, 30 :1325–1352.

[6] WARNER, D.M. (1976). Scheduling nursing personnel according to nursing preference : A mathematical programming approach. *Operations Research*, 24 :842–856.

[7] DOWSLAND, K.A. (1998). Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operation Research*, 106(2-3) :393–407.

[8] ABDENNADHER, S., et SCHENKER, H. (1999). Nurse scheduling using constraint logic programming. In *Eleventh Annual Conference on Innovative Applications of Artificial Intelligence, IAAI-99, Orlando, Florida.* AAAI Press.

[9] FORD, L.R., et FULKERSON, D.R. (1958). A suggested computation for maximal multi-commodity network flows. *Management Science*, pages 97–101.

[10] DANTZIG, G.B., et WOLFE, P. (1960). Decomposition principle for linear programs. *Operations Research*, pages 101–111.

[11] GILMORE, P.C., et GOMORY, R.E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, pages 849–859.

[12] SOUMIS, F. (1997). Decomposition and column generation. *Les Cahiers du GERAD, G-97-42.*

[13] HOFFMAN, K.L., et PADBERG, M. (1993). Solving airline crew-scheduling problems by branch-and-cut. *Management Science*, 39 :657–682.

[14] BARNHART, C., JOHNSON, E.L., ANBIL, R., et HATAY, L. (1994). A column generation technique for the long-haul crew assignment problem. *Optimization in Industry II, T.A. Cirani and R.C. Leachman eds., Wiley.*

[15] VANDERBECK, F., et WOLSEY, L. (1996). An exact algorithm for IP column generation. *Operations Research Letters*, pages 151–159.

[16] BARNHART, C., JOHNSON, E.L., NEMHAUSER, G.L., SAVELSBERGH, M.W.P., et VANCE, P.H. (1998). Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, pages 316–329.

[17] VANCE, P.H., BARNHART, C., JOHNSON, E.L., et NEMHAUSER, G.L. (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, pages 111–130.

[18] VANCE, P.H., ATAMTÜRK, A., BARNHART, C., GELMAN, E., JOHNSON, E.L., KRISHNA, A., MAHIDHARA, D., NEMHAUSER, G.L., et REBELLO, R. (1997). A heuristic branch-and-price approach for the airline crew pairing problem.

[19] DESAULNIERS, G., DESROSIERS, J., et SOLOMON, M. (1999). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. *Les Cahiers du GERAD.*

[20] CHVÁTAL, V. (1983). *Linear Programming.* Freeman.

[21] Ilog Inc. *ILOG CPLEX.* http ://www.ilog.com/products/cplex/.

[22] NEMHAUSER, G.L., et WOLSEY, L.A. (1988). *Integer and Combinatorial Optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience.

[23] RYAN, D.M., et FOSTER, J.C. (1981). An interger programming approach to scheduling. *Computer Scheduling of Public Transport*, pages 269–280.

[24] DESROCHERS, M., et SOUMIS, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23 :1–13.

[25] Ilog Inc. *ILOG Concert Technology.*
http ://www.ilog.com/products/optimization/tech/concert.cfm.

[26] GALINIER, P., JAUMARD, B., et LABIT, P. (2000). Horaires d'infirmières : description des formats d'entrées/sorties.

[27] BOURDAIS, S. (2004). Génération automatique d'horaires en milieu hospitalier. Master's thesis, École Polytechnique de Montréal.

ANNEXE A : RELATION ENTRE LES QUARTS ET LES PÉRIODES DES FICHIERS DU CUSM

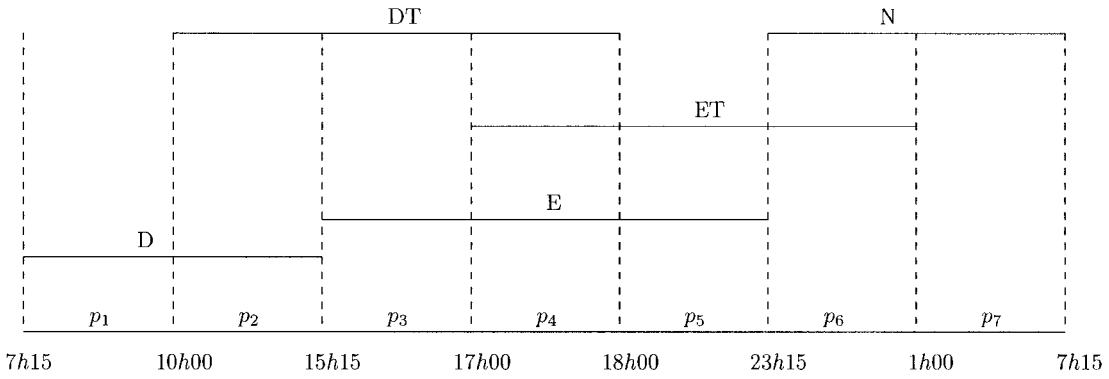


Figure A.1: Relation entre les quarts et les périodes du centre des naissances.

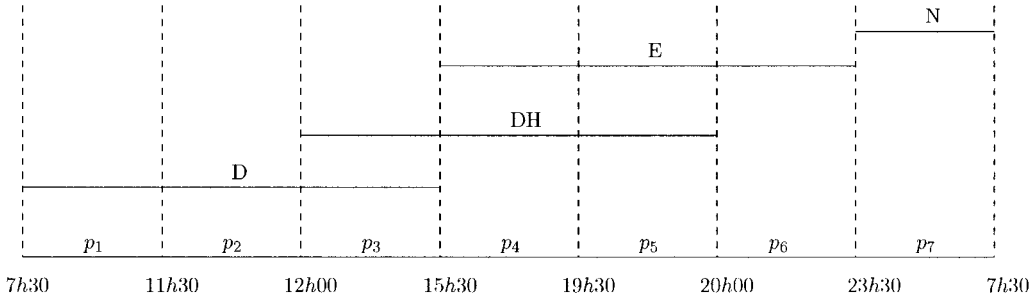


Figure A.2: Relation entre les quarts et les périodes de l'urgence.

ANNEXE B : ARBRES DE RECHERCHE OBTENUS PAR IRIS

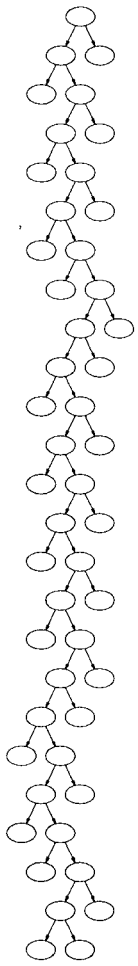


Figure B.1: Arbre obtenu par l'application de la version BSFS-1 de Iris aux données de l'urgence du CUSM.

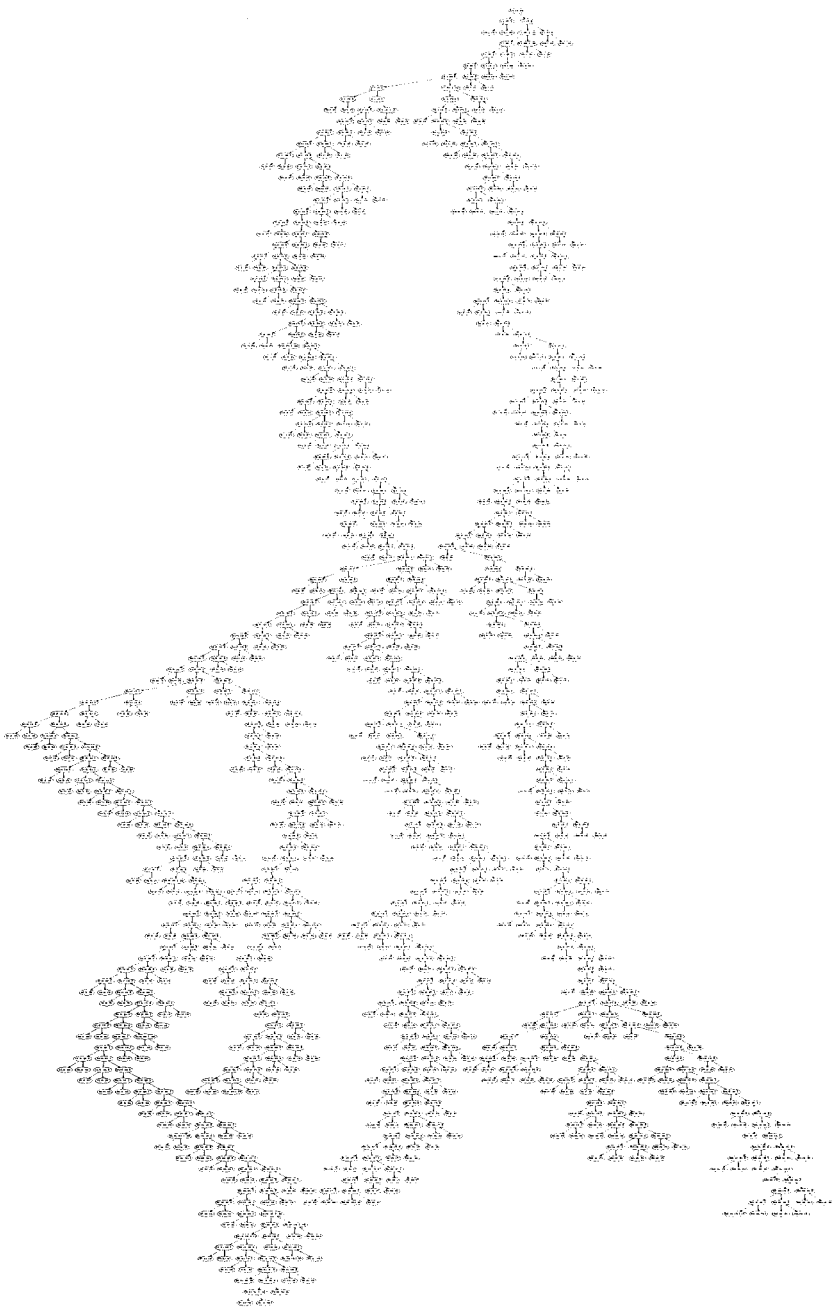


Figure B.2: Arbre obtenu par l'application de la version BSFS-1 de Iris aux données du centre des naissances du CUSM.