N° d'Ordre :
EDSPIC :

UNIVERSITÉ PIERRE ET MARIE CURIE - PARIS VI

ÉCOLE DOCTORALE INFORMATIQUE,
TÉLÉCOMMUNICATIONS ET ÉLECTRONIQUE DE PARIS

THÈSE

présentée par

Aurélien QUESTEL

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : INFORMATIQUE

Conception de réseaux en anneaux-étoiles et programmation mathématique

Soutenue publiquement le 30 mai 2013 devant le jury

MM	A. Ridha MAHJOUB	Président
	Andrea LODI	Rapporteur
	Frédéric SEMET	Rapporteur
	Michel MINOUX	Examinateur
	Eric PINSON	Examinateur
	Pierre FOUILHOUX	Encadrant
	Philippe CHRÉTIENNE	Directeur de Thèse

Résumé

Soit $G = (V,E,A)$ un graphe mixte. V est composé d'un ensemble N de dépôts, d'un ensemble U de clients et d'un ensemble W de sommets Steiner. Un anneau-étoile est un cycle élémentaire de taille bornée passant par l'un des dépôts sur lequel peuvent être affectés des clients via un arc $(i,j) \in A$ où j est un sommet du cycle. On considère le problème de couverture de graphe par des anneaux-étoiles reliant à moindre coût tous les clients de manière à ce que chaque arête appartienne à un nombre limité d'anneaux. Nous montrons que ce problème modélise le problème de conception de réseaux de télécommunications SDH (Synchronous Digital Hierarchy).

Dans un premier temps, nous étudions la version mono-dépôt de ce problème. Nous discutons de la caractérisation d'une solution et démontrons qu'il n'existe pas de formulation linéaire en nombres entiers à 2 indices. Nous proposons donc plusieurs formulations linéaires en nombres entiers à 3 indices pour ce problème avec un nombre polynomial de variables et comparons la valeur de leur relaxation linéaire. Nous introduisons plusieurs familles d'inégalités valides, inspirées de la littérature du TSP et du CVRP, pour renforcer l'une de ces formulations, dite naturelle.

Nous menons par la suite une étude polyédrale sur le dominant de la formulation naturelle défini comme la relaxation de l'obligation de desservir tous les clients. Nous donnons des conditions nécessaires et suffisantes pour que certaines classes d'inégalités de la formulation définissent des facettes du polytope. Nous discutons également d'algorithmes de séparation et montrons que certaines de ces classes peuvent être séparées en temps polynomial. Nous développons un algorithme de Branch-and-Cut basé sur ces résultats pour résoudre des instances générées aléatoirement ou issues d'un réseau réel déployé en région parisienne.

Nous considérons ensuite la version multi-dépôts du problème de conception de réseaux en anneaux-étoiles. Nous proposons pour ce problème une formulation set partitionning, habituellement utilisée pour des problèmes de tournées de véhicules, qui se résout par une méthode de génération de colonnes. Nous discutons du problème auxiliaire et montrons qu'une approche algorithmique classique par programmation

dynamique ne sera pas efficace dans ce cas. Comme ce problème auxiliaire consiste à rechercher un anneau-étoile de coût minimal, nous proposons d'utiliser pour le résoudre l'algorithme de Branch-and-Cut développé pour le cas mono-dépôt.

Une étude de la structure des solutions nous permet ensuite de renforcer notre formulation grâce à plusieurs classes d'inégalités valides issues des polytope du Stable, dont les inégalités de clique et de cycle impair. Ces inégalités sont connues pour leur efficacité mais ne sont pas utilisées habituellement dans les algorithmes de génération de colonnes car il est difficile de prendre en compte les coûts duaux associés à ces contraintes dans le problème auxiliaire. Nous montrons comment l'utilisation d'un algorithme de Branch-and-Cut pour résoudre le problème auxiliaire permet de prendre en compte de manière exhaustive ces familles de contraintes. Cette étude nous permet de développer un algorithme de Branch-and-Cut-and-Price qui est testé sur des instances générées alétoirement.

Mots clefs: couverture de graphe par des anneaux-étoiles, réseau SDH, polytope, facette, problème de séparation, génération de colonnes, Branch-and-Cut-and-Price.

Abstract

Given a graph $G = (V,E,A)$, V is the set of nodes which is partitioned into the set of depots N, the set of customers U, and the set of Steiner nodes W. A ring-star is a bounded elementary cycle going through one depot $v \in N$ along with a subset of arc $(i,j) \in A$ where i is a client and j is a node of the cycle. The ring-star covering problem consists in finding a subset of ring-stars of minimal cost such that every edge belongs to a limited number of cycles. We show that designing SDH networks reduces to solving Ring-Star problems.

First, we consider the single-depot case. We discuss the encoding of a solution, showing there is no 2-indices formulation for our problem. We then present three integer programming formulations and discuss the values of their linear relaxation. For the natural formulation, we describe several families of valid inequalities adapted from the CVRP and TSP literature.

We study from a polyhedral point of view the dominant of the polytope associated to the natural formulation, which corresponds to a relaxed version of the problem where clients may be not served. We give necessary and sufficient conditions for some constraints to be facet defining. We also discuss separation algorithms and show that some of these classes can be separated in polynomial time. We develop a branch-and-cut algorithm based on these results and use this algorithm to solve both random and real instances.

Finally, we study the multi-depot case. We propose a set partitionning formulation for this problem together with a column generation technique. We study the auxiliary problem and show that a classical algorithmic approach would not lead to an efficient pricing procedure. Since the auxiliary problem consists in finding a minimal cost ring-star on G, we propose to use our Branch-and-Cut algorithm for the mono-depot case as a pricer.

We then strengthen our formulation with additional inequalities, including the well-known clique and odd-cycle inequalities. These inequalities are known to be very efficient, meanwhile, their dual costs are hard to be taken into account in the auxiliary

problem. For that reason, there have been very few attempts to incorporate these inequalities in a Branch-and-Price algorithm. We present a method that allows to handle such inequalities using a Branch-and-Cut algorithm to solve the auxiliary problem. Finally, we present an efficient Branch-and-Cut-and-Price algorithm for the multi-depot case.

Mots clefs: ring-star covering, SDH network, polytope, facet, separation problem, column generation, Branch-and-Cut-and-Price.

Table des matières

Introduction **1**

1 Notions préliminaires et définitions **5**
1.1 Rappels sur la théorie des graphes 5
1.2 Eléments de la théorie des polyèdres 6
1.3 Programmation Linéaire en Nombres Entiers 8
1.3.1 Définitions 8
1.3.2 Approche polyédrale, méthode de coupes et branchements . . . 8
1.3.3 Approche par génération de colonnes 10
1.4 Problèmes d'Optimisation Combinatoire 13
1.5 Problème de couverture d'un graphe par des anneaux-étoiles non-disjoints (k-γ-RSP) 15

2 Fiabilité dans les réseaux : état de l'art **19**
2.1 Réseaux fiables 19
2.2 Fiabilité d'un réseau selon le modèle de Winter 21
2.3 Fiabilité dans les réseaux hiérarchiques 24
2.4 Recouvrement d'un graphe par des anneaux (ou des tournées de véhicules) 25
2.4.1 Formulation à 2 indices 26
2.4.2 Formulation Set Partitionning 27
2.4.3 Ajout d'inégalités valides 29
2.5 Recouvrement par des anneaux-étoiles 30

3 Modélisation du problème **33**
3.1 Présentation des réseaux SDH 33
3.1.1 Couche logique 33
3.1.2 Couche physique 35
3.2 Modélisation 39
3.3 Conclusion 41

4 Formulations linéaires mixtes pour le 1-γ-RSP 43
4.1 Caractérisation d'une solution . . . 43
4.2 Formulations compactes . . . 45
4.2.1 Formulation MTZ . . . 45
4.2.2 Formulation flot à une commodité . . . 47
4.3 Formulation naturelle . . . 48
4.4 Comparaison des formulations . . . 49
4.5 Inégalités valides . . . 50
4.5.1 Contraintes de connexité simple . . . 50
4.5.2 Contraintes de capacité fractionnaire . . . 51
4.5.3 Contraintes de capacité arrondie . . . 51
4.6 Conclusion . . . 52

5 Étude polyédrale et algorithme de Branch-and-Cut pour le 1-γ-RSP 53
5.1 Étude polyédrale d'un dominant de la formulation naturelle . . . 53
5.1.1 Propriétés basiques . . . 54
5.1.2 Contraintes de connexité . . . 57
5.2 Algorithme de Branch-and-Cut . . . 61
5.2.1 Algorithmes exactes et approchés de séparation . . . 63
5.2.2 Règle de branchement . . . 68
5.2.3 Heuristique primale . . . 69
5.3 Résultats expérimentaux . . . 72
5.4 Conclusion . . . 78

6 Génération de colonnes et inégalités valides pour le k-γ-RSP 79
6.1 Problème maître . . . 79
6.2 Problème auxiliaire . . . 80
6.2.1 Définition . . . 80
6.2.2 Résolution algorithmique . . . 81
6.2.3 Résolution par un algorithme de Branch-and-Cut . . . 83
6.3 Inégalités valides . . . 85
6.3.1 Contraintes de capacité . . . 85
6.3.2 Contraintes issues du polytope du Stable . . . 86
6.4 Conclusion . . . 89

7 Algorithme de Branch-and-Cut-and-Price pour le k-γ-RSP 91
7.1 Algorithme Branch-and-Cut-and-Price . . . 91
7.1.1 Initialisation . . . 91
7.1.2 Contraintes de branchement . . . 92
7.1.3 Séparation des contraintes additionnelles du Problème Maître . 93

7.1.4 Gestion des coûts duaux dans le problème auxiliaire 95
7.1.5 Algorithme de Branch-and-Cut pour le problème auxiliaire . . . 97
7.2 Résultats expérimentaux 98
7.2.1 Comparaison des approches pour le 1-γ-RSP 100
7.2.2 Résultats expérimentaux pour le k-γ-RSP 102
7.3 Conclusion 108

Conclusion **111**

Bibliographie **113**

Table des figures

1.1 Deux anneaux-étoiles non-disjoints 17

2.1 Quatre catégories de réseaux fiables 21

3.1 Reroutage en cas de rupture d'une connexion 35
3.2 Fourreau, câble et modules 36
3.3 Boîte de jonction placée dans une niche 37
3.4 Boîte de jonction ouverte 38
3.5 Cassette de raccordement 38
3.6 Illustration a) de la simple adduction, b) de la double adduction. 39
3.7 a) Un exemple de réseau SDH - b) Le cheminement des fibres 42
3.8 a) Une instance du k-γ-RSP - b) Une solution 42

4.1 Un point fractionnaire typique 50

5.1 Illustration des solutions de la preuve 5.1.2 62

6.1 Le graphe construit à partir de la figure 3.8-a pour le problème auxiliaire 84
6.2 Exemple de dominance des contraintes (6.12) sur les contraintes (6.13) 86

Liste des tableaux

5.1 Détail sur la densité des instances aléatoires pour $|N| = 1$ 73
5.2 Comparaison des relaxations linéaires des formulations flot et naturelle 74
5.3 Efficacité de l'heuristique primale 75
5.4 Impact des contraintes additionnelles sur la formulation naturelle . . . 76
5.5 Résultats expérimentaux sur les instances aléatoires 77
5.6 Résultats expérimentaux sur les instances réalistes 78

7.1 Comparaison entre le Branch-and-Cut et le Branch-and-Cut-and-Price . 98
7.2 Détail sur la densité des instances aléatoires pour $|N| = 2$ 99
7.3 Détail sur la densité des instances aléatoires pour $|N| = 3$ 100
7.4 Détail sur la densité des instances aléatoires pour $|N| = 4$ 101
7.5 Impact des contraintes additionnelles pour $|N| = 1$ 102
7.6 Impact des contraintes additionnelles pour $|N| = 2$ 103
7.7 Impact des contraintes additionnelles pour $|N| = 3$ 104
7.8 Impact des contraintes additionnelles pour $|N| = 4$ 105
7.9 Résolution exacte pour $|N| = 1$ 106
7.10 Résolution exacte pour $|N| = 2$ 107
7.11 Résolution exacte pour $|N| = 3$ 108
7.12 Résolution exacte pour $|N| = 4$ 109

Introduction

Avoir accès en permanence au réseau internet devient une nécessité pour les entreprises comme pour de plus en plus d'individus. Il est ainsi crucial pour les pourvoyeurs de réseaux de proposer une technologie fiable, c'est-à-dire qui maintient une connexion en cas de panne d'un équipement du réseau. La conception de réseaux de télécommunications fiables consiste ainsi à déterminer comment positionner les câbles d'un réseau de manière à assurer une fiabilité au moindre coût.

Une des techniques les plus efficaces pour concevoir de tels réseaux est l'utilisation de la Programmation Mathématique et en particulier de la Programmation Linéaire en Nombres Entiers (PLNE). Nous nous intéressons ici aux deux techniques algorithmiques qui ont permis de résoudre optimalement des instances de grandes tailles pour plusieurs problèmes célèbres de Recherche Opérationnelle : les algorithmes de coupes et les algorithmes de génération de colonnes. Les performances calculatoires de ces algorithmes sont principalement dues à la proximité de la PLNE avec l'Optimisation Combinatoire au travers de l'étude approfondie de la structure discrète du problème. Une des techniques les plus efficaces d'analyse des propriétés combinatoires d'un PLNE repose sur les Approches Polyédrales qui permettent une analyse fine de cette structure.

La littérature en programmation mathématique autour de la conception de réseaux fiables s'est jusqu'ici surtout intéressée à des problèmes d'optimisation combinatoire académiques. Dans cette thèse, nous étudions plus particulièrement la technologie SDH (Synchronous Digital Hierarchy, aussi appelée SONET aux États-Unis) qui est utilisée par la plupart des opérateurs télécom désirant offrir à leurs clients une connexion de haut-débit capable d'être maintenue sans perte de qualité en cas de panne. Elle repose sur un réseau de télécommunications en fibres optiques formant des anneaux permettant de relier par câbles distincts un petit groupe de clients à un centre technique appelé dépôt. Nous montrons que la conception de réseaux SDH se ramène au problème de couverture de sommets d'un graphe par des anneaux-étoiles non-disjoints passant par au moins l'un des k dépôts et en utilisant au plus γ fois chaque arêtes. Un anneau-étoile est un couple composé d'un cycle de taille limitée et d'un ensemble

d'arcs incidents aux sommets de ce cycle. On note ce problème le k-γ-RSP.

Le problème de couverture par un seul anneau-étoile ou par des anneaux-étoiles disjoints a déjà été abordé dans la littérature par l'utilisation de PLNE. Il est à noter que le k-γ-RSP est très proche du problème de tournées de véhicules à capacité limitée (CVRP) pour lequel de nombreux travaux ont été menés également en PLNE.

Dans un premier temps, nous étudions la version mono-dépôt de ce problème, le 1-γ-RSP. Nous discutons de la caractérisation d'une solution à ce problème et démontrons que, contrairement au problème CVRP, il n'existe pas de formulation PLNE à 2 indices. Nous proposons plusieurs formulations PLNE à 3 indices pour ce problème avec un nombre polynomial de variables et comparons la valeur de leur relaxation linéaire. Nous introduisons plusieurs familles d'inégalités valides, inspirées de la littérature du TSP et du CVRP, pour renforcer l'une de ces formulations, dite naturelle.

Nous menons par la suite une étude polyédrale sur le dominant de la formulation naturelle défini comme la relaxation de l'obligation de desservir tous les clients. Nous donnons des conditions nécessaires et suffisantes pour que certaines classes d'inégalités de la formulation définissent des facettes du polytope. Nous discutons également d'algorithmes de séparation et montrons que certaines de ces classes peuvent être séparées en temps polynomial. Nous développons un algorithme de Branch-and-Cut basé sur ces résultats pour résoudre des instance du 1-γ-RSP générées aléatoirement ou issues d'un réseau réel déployé en région parisienne.

Nous considérons ensuite la version multi-dépôts du problème de conception de réseaux en anneaux-étoiles, le k-γ-RSP. Nous proposons pour ce problème une formulation proche de celle utilisée pour le CVRP, appelée fréquemment formulation set partitionning. Une telle formulation possède un nombre exponentiel de variables et doit donc être résolue par une méthode de génération de colonnes. Nous discutons du problème de génération de colonnes associé, appelé problème auxiliaire, et montrons qu'une approche algorithmique classique par programmation dynamique ne sera pas efficace dans ce cas. Comme le problème auxiliaire consiste à rechercher un anneau-étoile de coût minimal, nous proposons d'utiliser pour le résoudre l'algorithme de Branch-and-Cut développé pour le cas mono-dépôt. Une étude de la structure des solutions du k-γ-RSP nous permet ensuite de proposer plusieurs classes d'inégalités valides, issues des polytopes du Stable et du CVRP, permettant de renforcer notre formulation.

En particulier, les inégalités de clique et de cycle impair, issues des polytope du Stable, sont connues pour leur efficacité mais ne sont pas utilisées habituellement dans les algorithmes de génération de colonnes. En effet, il est difficile de prendre en compte les coûts duaux associés à ces contraintes dans le problème auxiliaire. Nous montrons comment l'utilisation d'un Branch-and-Cut pour résoudre le problème auxiliaire per-

met, pour la première fois, de prendre en compte de manière exhaustive ces familles de contraintes. Cette étude nous permet de développer un algorithme de Branch-and-Cut-and-Price qui est testé sur des instances du k-γ-RSP générées aléatoirement.

Le plan de ce document sera le suivant : dans le premier chapitre nous introduisons quelques notions de base de la programmation mathématique et de l'approche polyédrale, ainsi que certaines définitions et notations. Dans le chapitre 2 nous présentons un état de l'art pour le problème de conception de réseaux fiables et présentons les techniques de résolution utilisées pour les problèmes de tournées de véhicules. Dans le chapitre 3, nous montrons comment le problème de conception de réseaux SDH se réduit au k-γ-RSP. Les chapitres 4 et 5 sont dédiés au cas mono-dépôt. Dans le chapitre 4 nous étudions la caractérisation d'une solution du problème et proposons plusieurs formulations PLNE. Le chapitre 5 présente une étude polyédrale et un algorithme de Branch-and-Cut pour le 1-γ-RSP ainsi que des résultats expérimentaux. Nous présentons dans le chapitre 6 une formulation à nombre exponentiel de variables pour le k-γ-RSP, renforcée par l'ajout d'inégalités valides. Le chapitre 7 détaille la mise en oeuvre d'un algorithme de Branch-and-Cut-and-Price ainsi que des résultats expérimentaux pour le k-γ-RSP.

Chapitre 1

Notions préliminaires et définitions

Dans ce chapitre, nous donnons quelques notions de base sur la théorie des graphes et la programmation mathématique. Nous donnons également des définitions et des notations qui seront utilisées tout au long de ce mémoire.

1.1 Rappels sur la théorie des graphes

Les graphes que nous considérons sont finis, mixtes et sans arête ni arc multiple. Nous notons un graphe par $G = (V,E,A)$ où V est l'*ensemble des sommets*, E l'*ensemble des arêtes* et A l'*ensemble des arcs* de G. Si $e \in E$ est une arête d'extrémités u et v, l'arête e peut être également notée $\{u,v\}$. Si $(i,j) \in A$ est un arc allant de i à j nous dirons que (i,j) est un arc *sortant* de i et *entrant* en j.

Soit $G = (V,E,A)$ un graphe mixte. Un *sous-graphe* $H = (W,F,B)$ de G est un graphe tel que $W \subseteq V$, $F \subseteq E$ et $B \subseteq A$. On appelle *sous-graphe partiel* de G un sous-graphe $G' = (V,E',A')$ avec $E' \subset E$ et $A' \subset A$.

Si $F \subset E$, alors $V(F)$ désigne l'ensemble des sommets de V qui sont extrémités des arêtes de F. Si $W \subset V$, alors $\delta(W)$ est l'ensemble des arêtes ayant une extrémité dans W et l'autre extrémité dans $V \setminus W$. L'ensemble $\delta(W)$ est appelé une *coupe*. On note $\delta(v)$ à la place de $\delta(\{v\})$ pour $v \in V$ et on appelle $\delta(v)$ l'*étoile* de v.

Une *chaîne* P dans $G = (V,E)$ est une séquence d'arêtes $e_1,e_2,\ldots,e_k$ telles que $e_1 = \{v_0,v_1\},e_2 = \{v_1,v_2\},\ldots,e_k = \{v_{k-1},v_k\}$. P est dite *chaîne élémentaire* si elle passe au plus une fois par le même sommet. Les sommets v_0 et v_k sont les extrémités

de P et on dit que P relie v_0 à v_k. Le nombre k d'arêtes de P est appelé la *taille* de P. Si $P = e_1,e_2,\ldots,e_k$ est une chaîne (*resp.* chaîne élémentaire) reliant v_0 à v_k et $e_{k+1} = \{v_0,v_k\} \in E$, alors la séquence $e_1,e_2,\ldots,e_k,e_{k+1}$ est dite un *cycle* (*resp. cycle élémentaire*) de taille $k+1$. Lorsqu'il n'y a pas d'ambiguïté, on utilisera chaîne (*resp.* cycle) pour désigner une chaîne (*resp.* un cycle) élémentaire. Un cycle (chaîne) est dit *impair* si la taille est impaire, autrement, il est dit *pair*. Si P est un cycle ou une chaîne et $\{u,v\}$ une arête de $E \setminus P$ avec $u,v \in V(P)$, alors $\{u,v\}$ est appelé une *corde* de P. Un *trou* de G est un cycle sans corde.

Un graphe $G = (V,E)$ est *connexe* si, pour toute paire de sommets u,v de V, il existe au moins une chaîne entre u et v. Une *composante connexe* d'un graphe est un sous-graphe connexe qui est maximal par rapport à cette propriété.

Un graphe $G = (V,E)$ est dit *complet* si toute paire de sommets de V est jointe par une arête de E. On appelle *clique* de G un sous-graphe complet de G. Un *stable* d'un graphe est un ensemble de sommets deux à deux non-adjacents.

Un graphe est dit *planaire* si on peut le dessiner sur un plan de manière à ce qu'aucune arête n'en croise une autre.

1.2 Eléments de la théorie des polyèdres

Dans cette section, nous allons introduire quelques définitions et propriétés de la théorie des polyèdres. Pour plus de détails, on peut se référer à Pulleyblank [92] ou Schrijver [97].

Soit $x \in I\!R^n$. On dit que x est une *combinaison linéaire* des points $x_1,\ldots,x_k \in I\!R^n$ s'il existe k scalaires $\lambda_1,\lambda_2,\ldots,\lambda_k \in I\!R$ tels que $x = \sum_{i=1}^k \lambda_i x_i$. Si, de plus, $\sum_{i=1}^k \lambda_i = 1$, alors on dit que x est une *combinaison affine* de ces points. De même, si $\lambda_i \geq 0$ pour tout $i \in \{1,\ldots,k\}$ avec $\sum_{i=1}^k \lambda_i = 1$, x est alors dit *combinaison convexe* de ces points.

Des points $x_1,\ldots,x_k \in I\!R^n$ sont dits *linéairement indépendants* (*resp. affinement indépendants*) si le système

$$\sum_{i=1}^k \lambda_i x_i = 0 \qquad \left(\textit{resp.} \ \sum_{i=1}^k \lambda_i x_i = 0 \ \text{ et } \ \sum_{i=1}^k \lambda_i = 0 \right)$$

admet une solution unique, $\lambda_i = 0$ pour $i = 1,...,k$.

Soit S un ensemble non vide de points de $\mathbb{R}^n$. L'*enveloppe convexe* des points de S, notée *conv(S)*, est l'ensemble de tous les points de $\mathbb{R}^n$ qui peuvent s'écrire comme combinaison convexe de points de S.

Un *polyèdre* P est un ensemble de points de $\mathbb{R}^n$ engendré par l'intersection d'un nombre fini de demi-espaces de $\mathbb{R}^n$. D'une manière équivalente, P est l'ensemble des solutions d'un système d'inégalités linéaires, c'est-à-dire $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, où A est une matrice à m lignes et n colonnes, et b un vecteur à m composantes. Nous dirons alors que le système $Ax \leq b$ définit (ou caractérise) le polyèdre P. Un *polytope* est un polyèdre borné. Par exemple, l'enveloppe convexe d'un ensemble fini de points est un polytope.

Un polyèdre P de $\mathbb{R}^n$ est de *dimension* d si le nombre maximum de points de P affinement indépendants est égal à $d + 1$. On écrit alors $dim(P) = d$. Un polyèdre P est dit de *pleine dimension* si $dim(P) = n$. Un point $x \in \mathbb{R}^n$ est un *point extrême* d'un polyèdre P s'il ne peut pas être écrit comme combinaison convexe d'autres points de P.

Une inégalité (ou *contrainte*) $ax \leq \alpha$ est dite *valide* pour un polyèdre P si elle est vérifiée par tous les points de P. Soit $ax \leq \alpha$ une inégalité valide pour P. Le sous-ensemble $F = \{x \in P : ax = \alpha\}$ est appelé *face* de P. On dit aussi que F est la face définie par $ax \leq \alpha$. Si $F \neq \emptyset$ et $F \neq P$, on dit que la face F est *propre*. Si F est propre et $dim(F) = dim(P) - 1$, alors F est appelée *facette* de P.

Une inégalité est dite *essentielle* pour P si elle définit une facette du polyèdre P. En revanche, une inégalité est dite *redondante* dans un système $Ax \leq b$ définissant un polyèdre P, si le sous-système, obtenu à partir de $Ax \leq b$ en supprimant cette inégalité, définit le même polyèdre P. Soit $x^* \in \mathbb{R}^n$. Nous dirons que l'inégalité $ax \leq \alpha$ est *serrée* (*resp. violée*) par x^* si $ax^* = \alpha$ (*resp.* $ax^* > \alpha$).

1.3 Programmation Linéaire en Nombres Entiers

1.3.1 Définitions

Un problème de programmation linéaire (P) consiste à minimiser (ou maximiser) une fonction linéaire sous contraintes linéaires, c'est-à-dire :

$$\begin{aligned} (P) \quad & \min cx \\ s.t. \quad & Ax \leq b \\ & x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n \end{aligned}$$

où n est le nombre de variables, m le nombre de contraintes, A une matrice réelle $n \times m$, $c = (c_1, c_2, \ldots, c_n)$ le vecteur-ligne coût et $b = (b_1, b_2, \ldots, b_m)^T$ le vecteur-colonne des seconds membres.

$x = (x_1, \ldots, x_n)^T$ est une *solution* de (P) si et seulement si $Ax \leq b$.
$x = (x_1, \ldots, x_n)^T$ est une *solution réalisable* de (P) si et seulement si $Ax \leq b$ et $x \geq 0$.
Une *solution optimale* de (P) est une solution réalisable qui donne à la fonction $f(x) = cx$ une valeur minimale.

La programmation linéaire est un problème d'optimisation combinatoire polynomial (Khachiyan, 1979). Un programme linéaire peut être résolu efficacement grâce à l'algorithme du *simplexe* (Dantzig, 1947) ou par les méthodes de points intérieurs (Karmarkar, 1984).

Si les variables x prennent leur valeur dans un espace discret, on parle de *Programme Linéaire en Nombres Entiers* (PLNE). La résolution d'un PLNE est NP-difficile et ce problème est couramment résolu par une *méthode de branchement et évaluation* (méthode de *Branch-and-Bound*).

1.3.2 Approche polyédrale, méthode de coupes et branchements

Soient $\mathcal{P}$ un problème d'optimisation combinatoire (*cf.* la section 1.4) et $\mathcal{S}$ l'ensemble de ses solutions. Le problème $\mathcal{P}$ s'écrit alors

$$\max \{cx : x \in \mathcal{S}\},$$

où c est une fonction coût associée aux variables du problème. Considérons l'enveloppe convexe $conv(\mathcal{S})$ des solutions de $\mathcal{P}$. Le problème $\mathcal{P}$ est alors équivalent au programme

linéaire

$$\max \{cx : x \in conv(\mathcal{S})\}.$$

Par conséquent, si nous caractérisons le polyèdre $conv(\mathcal{S})$ par un système d'inégalités linéaires, alors nous ramenons le problème $\mathcal{P}$ à la résolution d'un programme linéaire.

L'*approche polyédrale*, introduite par Edmonds [37] dans le cadre du problème du couplage, consiste à étudier le polytope $conv(\mathcal{S})$ afin de pouvoir résoudre $\mathcal{P}$ comme un programme linéaire. La caractérisation complète du polytope $conv(\mathcal{S})$ est généralement difficile à obtenir. Par ailleurs, une description complète du polyèdre peut comporter un nombre exponentiel d'inégalités. Cependant, un nombre réduit de ces inégalités peut être suffisant pour résoudre le problème à l'aide d'une *méthode de coupes*. Cette méthode permet de résoudre un problème $\mathcal{P}$ comme une séquence de programmes linéaires, chacun contenant un nombre raisonnable de contraintes. Pour pouvoir utiliser une telle méthode, nous devons déterminer des classes d'inégalités valides (et non redondantes) pour $conv(\mathcal{S})$, pour lesquelles le problème suivant, appelé *problème de séparation*, peut être résolu efficacement.

Problème Soit $\mathcal{C}$ une classe d'inégalités valides pour $conv(\mathcal{S})$. Etant donné un point $x \in I\!R^n$, le *problème de séparation* associé à $\mathcal{C}$ et x consiste à déterminer si x satisfait toutes les inégalités de $\mathcal{C}$ et sinon de trouver une inégalité de $\mathcal{C}$ violée par x.

Le problème de séparation est la clef de voûte d'une méthode de coupes. En effet, Grötschel, Lovász et Schrijver [56] ont montré qu'un problème d'optimisation combinatoire sur un ensemble de contraintes $\mathcal{C}$ peut être résolu en temps polynomial si et seulement si le problème de séparation associé à $\mathcal{C}$ peut être résolu en temps polynomial. Ainsi, une méthode de coupes permet de résoudre un problème d'optimisation combinatoire en temps polynomial si on sait résoudre en temps polynomial le problème de séparation pour un système d'inégalités caractérisant le polyèdre associé.

Cependant, comme il a déjà été noté auparavant, il y a peu d'espoir de connaître la caractérisation complète du polyèdre des solutions du problème. D'autre part, le problème de séparation sur certaines classes d'inégalités valides peut être lui-même NP-difficile. Dans ce cas, on ne peut disposer que de techniques de séparation approchées. Ainsi, une méthode de coupes seule peut ne fournir que des solutions fractionnaires (non optimales). Dans ce cas, on exécute une étape de branchement qui peut consister à choisir une variable fractionnaire x_i dans la solution et à considérer deux sous-problèmes du problème courant en fixant x_i à 0 pour l'un et à 1 pour l'autre. On applique alors la méthode de coupes pour les deux sous-problèmes. La solution optimale du problème sera donc la meilleure entre les deux solutions entières obtenues pour les deux sous-problèmes. Cette phase de branchement est répétée de manière ré-

cursive jusqu'à l'obtention d'une solution entière optimale. Cette combinaison de la méthode de branchements et d'une méthode de coupes au niveau de chaque noeud de l'arbre de branchement est appelée *méthode de coupes et branchements* (ou méthode de *Branch-and-Cut*). Cette méthode s'est révélée très efficace pour la résolution de problèmes d'optimisation combinatoire réputés pour être difficiles tels que le problème du voyageur de commerce [2] ou celui de la coupe maximum [12, 13].

1.3.3 Approche par génération de colonnes

1.3.3.1 Définitions

Il arrive parfois qu'une modélisation mène à une formulation linéaire ayant un nombre exponentiel de variables. C'est le cas lorsque les *objets* de la modélisation sont les chemins ou les cycles d'un graphe par exemple. Soit Ω l'ensemble des objets r, indicés par $k \in \{1,2,\ldots,|\Omega|\}$, de la formulation et m le nombre de contraintes. La formulation que nous appellerons *Problème Maître* sera de la forme suivante :

$$\begin{array}{lll} (MP(\Omega)) & \min \sum_{r_k \in \Omega} c_k \theta_k & \\ s.t. & \sum_{r_k \in \Omega} a_i^k \theta_k \leq b_i, & \forall i = 1, \ldots, m \\ & \theta_k \geq 0 & \forall r_k \in \Omega \end{array}$$

dans laquelle c_k est le coût de l'objet r_k, a_i^k est le coefficent de la variable r_k dans la contrainte i, b_i est le membre droit de la contrainte i et θ_r est une variable positive associée à l'objet r_k.

Soit λ_i la *variable duale* non-positive associée à la contrainte i du problème maître. Le *dual* du problème maître $MP(\Omega)$, noté $D(\Omega)$, est donné par le programme linéaire suivant :

$$\begin{array}{lll} (D(\Omega)) & \max \sum_{i=1}^{m} b_i \lambda_i & \\ s.t. & \sum_{i=1}^{m} a_i^k \lambda_i \geq c_k, & \forall r_k \in \Omega \\ & \lambda_i \leq 0 & \forall i = 1, \ \ldots, \ m \end{array}$$

Soient θ^* une solution réalisable de $MP(\Omega)$ et λ^* le vecteur de coût dual associé à la solution θ^*, le *coût réduit* $\tilde{c_k}$ d'une variable θ_k^* est donné par la formule suivante :

$$\tilde{c_k} = c_k - \sum_{i=1}^{m} a_i^k \lambda_i^*.$$

La taille importante de Ω ne permet pas de résoudre directement le problème maître. Cependant, un nombre réduit de variables peut être suffisant pour résoudre le problème à l'aide d'une méthode de *génération de colonnes*. Cette méthode permet de résoudre un tel problème comme une séquence de programmes linéaires, chacun contenant un nombre raisonnable de variables.

Soit $\Omega_1 \subset \Omega$ un sous-ensemble de variables tel que la formulation limitée aux variables Ω_1 ait une solution réalisable. Nous noterons $(MP(\Omega_1))$ cette formulation et l'appellerons le *Problème Maître Restreint*. Le dual associé à $MP(\Omega_1)$ sera noté $D(\Omega_1)$.

$$\begin{array}{lll} (MP(\Omega_1)) & \min \sum_{r_k \in \Omega_1} c_k \theta_k & \\ s.t. & \sum_{r_k \in \Omega_1} a_i^k \theta_k \leq b_i, & \forall i = 1, \ldots, m \\ & \theta_k \geq 0 & \forall r_k \in \Omega_1 \end{array}$$

Propriété Soient $\Omega_1 \subset \Omega$ et λ^* une solution optimale de $D(\Omega_1)$. Si λ^* est réalisable pour $D(\Omega)$, alors λ^* est optimale pour $D(\Omega)$.

Conséquemment, soit θ_1^* une solution optimale de $MP(\Omega_1)$ et λ_1^* la solution duale optimale associée à θ_1^*. Si λ_1^* est réalisable pour $D(\Omega)$ alors $D(\Omega)$ et $MP(\Omega)$ sont résolus optimalement. Sinon, une ou plusieurs contraintes de $D(\Omega)$ sont violées par λ_1^*. Le principe de la génération de colonnes est d'identifier une ou plusieurs de ces contraintes, en résolvant un problème d'optimisation approprié appelé *Problème Auxiliaire*, de manière à intégrer les variables correspondantes à l'ensemble Ω_1. Ainsi, en résolvant itérativement $MP(\Omega_1)$ et le problème auxiliaire, nous convergeons vers la réalisabilité du problème dual $D(\Omega)$. La terminaison d'un tel algorithme est garanti puisque l'ensemble Ω est fini.

D'un certain point de vue, la génération de colonnes reproduit l'algorithme du simplex puisqu'une contrainte duale violée correspond exactement à une variable primale ayant un coût réduit négatif dans MP quand la solution courante est optimale pour $MP(\Omega_1)$. Étant donné une solution θ_1^* optimale pour $MP(\Omega_1)$, le problème auxiliaire revient donc à rechercher un objet $r_k \in \Omega \setminus \Omega_1$ tel que le coût réduit de la variable θ_k est strictement négatif. Il est équivalent, et généralement plus simple, d'étendre cette recherche à l'ensemble Ω. En effet, θ_1^* étant optimale pour $(MP(\Omega_1))$, tous les objets de Ω_1 satisfont la condition précédente. Enfin, notons que la nature et la difficulté du problème auxiliaire varie selon la nature des objets contenus dans l'ensemble Ω. Une technique de génération de colonnes ne pourra être mise en oeuvre que si le problème auxiliaire peut être résolu efficacement.

1.3.3.2 Mise en oeuvre

• *Initialisation*

Pour pouvoir se lancer, un algorithme de génération de colonnes doit disposer d'un premier ensemble $\Omega_0 \subset \Omega$ tel que le problème restreint à Ω_0 admette une solution réalisable. Selon les problèmes, il peut être difficile de déterminer un tel ensemble. Pour contourner la difficulté, il est possible d'introduire un ensemble de variables artificielles $\mathcal{S} = \{s_1, \ldots, s_m\}$ où chaque variable s_i n'apparait que dans la contrainte i et tel que le problème restreint à $\mathcal{S}$ admette une solution réalisable. En donnant à chacune de ces variables un coût suffisamment grand, nous aurons l'assurance qu'elles seront écartées dès qu'une solution réalisable sera rencontrée.

• *Arbre de branchement*

Généralement, les variables du problème maître sont binaires alors que la solution obtenue à la fin de la procédure de génération de colonnes n'est pas entière. Dans ce cas, on exécute une phase de branchement qui consiste à considérer deux sous-problèmes du problème courant. On applique la méthode de génération de colonnes pour les deux sous-problèmes. La solution optimale du problème sera donc la meilleure entre les deux solutions obtenues par les deux sous-problèmes. Cette phase de branchement est répétée de manière récursive jusqu'à l'obtention d'une solution entière optimale. Cette combinaison de la méthode de branchement et d'une méthode de génération de colonnes est appelée méthode de *génération de colonnes et branchement* (ou métode de *Branch-and-Price*).

• *Gestion de l'infaisabilité*

Les différentes décisions prises pendant la phase de branchement peuvent nous amener à une situation où le sous-problème n'a plus de solution réalisable. Pour prouver qu'un sous-problème n'est pas réalisable, nous utilisons le fait que son dual n'est pas borné. Cette assertion est formalisée par le Lemme de Farkas qui dit que soit le problème $A\theta = b, \theta \geq 0$ est réalisable, soit il existe un vecteur π avec $\pi A \leq 0$ et $\pi b > 0$. Un tel vecteur π prouve la non-réalisabilité du problème maître puisque $\pi A\theta = \pi b$ ne peut être vérifié. L'idée est donc d'ajouter une nouvelle variable au programme maître avec un vecteur coefficient a tel que $\pi a > 0$ et ainsi détruire cette preuve de non-réalisabilité. Une telle variable peut être trouvée en résolvant

$$max_{r_k \in \Omega}\{\pi^* a(\theta_k)\} = min_{r_k \in \Omega}\{-\pi^* a\theta_k\}$$

ce qui est exactement le problème auxiliaire classique avec un vecteur de coût nul $c(\theta) = 0$. Si une variable ne peut être trouvée, l'infaisabilité du sous-problème a été prouvée et l'arbre de branchement peut être élagué.

1.3.3.3 Algorithme de coupes et génération de colonnes

Si l'on considère un programme linéaire (ou un PLNE) contenant à la fois un nombre exponentiel de variables et un nombre exponentiel de contraintes, il est nécessaire d'utiliser dans un même algorithme des techniques de génération de coupes et de colonnes pour le problème maître. Un tel algorithme s'appelle alors un algorithme de coupes et génération de colonnes, souvent couplé à une méthode de branchement (Branch-and-Cut-and-Price).

Une telle méthode demande de prendre en compte finement les coûts réduits des contraintes générées par séparation lors de la génération de colonnes. De même, les variables nouvellement générées doivent être correctement insérées dans les contraintes déjà ajoutées. L'ensemble de ces difficultés rend ardue la mise en oeuvre d'un tel algorithme.

1.4 Problèmes d'Optimisation Combinatoire

Dans cette section nous donnons les définitions de plusieurs problèmes d'optimisation combinatoire qui seront évoqués dans ce document.

Un problème d'optimisation combinatoire (OC) consiste à déterminer un plus grand (petit) élément dans un ensemble fini valué. En d'autres termes, étant donnés une famille $\mathcal{F}$ de sous-ensembles d'un ensemble fini $E = \{e_1, \ldots, e_n\}$ et un système de poids $w = (w(e_1), \ldots, w(e_n))$ associé aux éléments de E, un problème d'optimisation consiste à trouver un ensemble $F \in \mathcal{F}$ de poids $w(F) = \sum_{e \in F} w(e)$ maximum (ou minimum), *i.e.*

$$\text{max ou min } \{w(F) : F \in \mathcal{F}\}.$$

La famille $\mathcal{F}$ représente donc les solutions du problème. Elle peut correspondre à un ensemble de très grande taille que l'on ne connaît que par des descriptions ou des propriétés théoriques qui ne permettent pas facilement son énumération.

• *Le Problème de Voyageur de Commerce (TSP)*

Soit $G = (V,E)$ un graphe connexe non-orienté. Un circuit *hamiltonien* est un circuit passant une et une seule fois par chacun des sommets de G. Soit d un vecteur associant une longueur l_e à chaque arête e dans E. Le Problème de Voyageur de Commerce (Traveling Salesman Problem) revient à rechercher un cycle hamiltonien de longueur minimale dans le graphe G.

• *Le Problème de Tournées de Véhicules à Capacité Limitée (CVRP)*

Soit $G = (V,E)$ un graphe complet non orienté où V est un ensemble de $n+1$ sommets $V = \{0,1,\ldots,n+1\}$. Notons $V_c = \{1,\ldots,n+1\}$ l'ensemble des n clients et 0 le dépôt. Un coût d_{ij} est associé à chaque arête $\{i,j\}$ de E et tel que la matrice $[d_{ij}]$ satisfait les inégalités triangulaires, *i.e.*

$$w_{ij} + w_{jk} \leq w_{ik},\ \forall \{i,j,k\} \subset V.$$

Considérons une flotte de m véhicules identiques de capacité Q, stationnés au dépôt 0. Chaque client i de V_c demande une quantité d_i de produits stockés au dépôt 0 ($0 < d_i \leq Q$). Une *route* est un cycle élémentaire de G passant par le dépôt 0 tel que la demande cumulée des clients visités ne dépasse pas Q. Le coût d'une route sera la longueur totale du cycle.

Le Problème de Tournées de Véhicule à Capacité Limitée (Capacitated Vehicle Routing Problem) revient à déterminer au plus m routes telles que tous les clients soient desservis et telles que le coût total des routes soit minimal. Notons qu'en général m est une donnée du problème (sinon une valeur m peut être facilement déterminée).

• *Le Problème du Set Partitionning*

Soit ξ une collection de sous-ensembles d'un ensemble fini S d'éléments. Un coût c_E est associé à chaque sous-ensemble $E \in \xi$. Le problème du Set Partitionning revient à déterminer un sous-ensemble $C \subset \xi$ de coût $\sum_{E \in C} c_E$ maximum (ou minimum) tel que chaque élément de S appartienne à exactement un élément de C, *i.e.*

$$\forall x \in S,\ |\{E \in C : x \in E\}| = 1.$$

• *Le Problème du Stable*

Soit $G = (V,E)$ un graphe non-orienté où les sommets u sont munis d'un poids c_u. Le problème du stable revient à rechercher un stable de poids maximum. Le problème de recherche d'un stable dans un graphe est très proche du problème du Set Partionning. En fait, le polyèdre du Set Partitionning est une face du polyèdre associé à la recherche d'un stable.

• *Le Problème de Plus Court Chemin avec Contraintes de Ressources (ESPPRC)*

Soient $G = (V,E)$ un graphe non orienté et une ressource limitée à une quantité Q. V est un ensemble de $n+2$ sommets $V = \{s,t,1,\ldots,n\}$ où l'ensemble $V_c = \{1,\ldots,n\}$ représente un ensemble de n clients. Un coût d_{ij} est associé à chaque arête $\{i,j\}$ de E.

On note d_i la *demande* d'un client i de V_c qui se traduit par la consommation d'une quantité d_i de la ressource ($0 < d_i \leq Q$) lorsque le chemin passe par le client i. Le Problème de Plus Court Chemin avec Contraintes de Ressources (Elementary Shortest Path Problem with Ressource Constraints) revient à rechercher le chemin élémentaire de longueur minimale allant de s à t tel que la demande cumulée des clients visités soit inférieure ou égale à Q.

1.5 Problème de couverture d'un graphe par des anneaux-étoiles non-disjoints (k-γ-RSP)

Dans ce document, nous nous intéresserons au problème d'optimisation combinatoire suivant qui n'a pas été étudié dans son cas général dans la littérature et que nous définissons ici.

Soit un graphe mixte $G = (V,E,A)$. L'ensemble V des sommets du graphe est partitionné entre l'ensemble N des *dépôts*, l'ensemble U des *clients* et l'ensemble W des *sommets d'interconnexion* (aussi connus sous le nom de sommets *Steiner*) : $V = N \cup U \cup W$. On notera $n = |V|$ le nombre de sommets du graphe et $n_u = |U|$ le nombre de clients.

Nous associons à chaque client $i \in U$ une *demande* d_i. L'ensemble des arcs A est égal à $\cup_{i \in U} L_i$, où L_i est l'ensemble des *affectations* potentielles du client $i \in U$, $L_i \subset \{(i,j),\ j \in V \setminus N\}$. Les sommets ayant un arc entrant seront appelés des sommets *connecteurs*. Chaque arête $e \in E$ possède une *capacité* γ_e. À chaque arc $(i,j) \in A$ est associé un *coût d'affectation* $c_{ij} \in \mathbb{R}^+$, et à chaque arête $e \in E$ est associé un *coût de routage* $l_e \in \mathbb{R}^+$.

Un couple $R = (E(R),L(R))$ sera appelé *anneau-étoile* si $E(R)$ est un cycle élémentaire de G passant par au moins un dépôt $d \in N$; et si $L(R)$ est un sous-ensemble d'arcs tel que pour chaque arc (i,j) de $L(R)$, j est un sommet du cycle $E(R)$. Les clients de l'ensemble $U(R) = \{i \in U : \exists (i,j) \in L(R)\}$ seront dits *desservis* par l'anneau-étoile R. De plus, un client de $U(R)$ sera affecté à exactement un sommet connecteur du cycle $E(R)$. Étant donné un anneau-étoile R, les sommets connecteurs j ayant un arc incident dans R seront appelés *connecteurs actifs*.

Pour un anneau-étoile R donné, on peut remarquer qu'un client peut être desservi par R sans appartenir au cycle $E(R)$: il est alors relié à un sommet connecteur de $E(R)$ par un arc unique de $L(R)$. Lorsqu'un client appartient à $E(R)$, il n'est pas nécessairement desservi par R. Dans le cas où le client i est à la fois desservi par R et appartient à $E(R)$, alors l'arc (i,i) appartient à $L(R)$.

La figure 1.1 donne un exemple avec deux anneaux-étoiles, le premier est représenté par un tracé continu, le second par un tracé pointillé. On voit que l'une des arêtes du graphe est utilisée par les deux anneaux-étoiles. Le graphe ne contient qu'un seul dépôt, représenté par un carré. Les sommets Steiner sont représentés par des triangles et les clients par des cercles. Les connecteurs actifs sont remplis en gris, les clients desservis sont hachurés. On peut remarquer que le client i est affecté à lui-même car il est à la fois un client desservi par l'anneau et un connecteur actif de l'anneau. En revanche, le client j appartient à un anneau-étoile sans pour autant être desservi par ce dernier.

Étant donné une quantité $Q \in \mathbb{R}^+$, le k-γ-RSP revient à rechercher un ensemble d'anneaux-étoiles tel que :

- la demande cumulée de chaque anneau-étoile soit inférieure ou égale à Q ;
- chaque arête $e \in E$ appartienne à au plus γ_e anneaux-étoiles ;
- chaque client soit desservi par exactement un anneau-étoile ;
- et tel que la somme des coûts d'affectation et de routage soit minimale.

Il est utile d'un point de vue pratique de connaître une borne inférieure du nombre d'anneaux-étoiles nécessaires pour desservir tous les clients de U. On notera m cette borne. Par exemple on peut utiliser la valeur $\frac{\sum_{i \in U} d_i}{Q}$. m étant un entier, la quantité précédente peut être arrondie à l'entier supérieur : $\left\lceil \frac{\sum_{i \in U} d_i}{Q} \right\rceil$. En théorie, m est même supérieure ou égale à la valeur de la solution d'un problème de Bin Packing (*cf.* section 4.1). En pratique, sur les instances qui seront utilisées dans ce document, la valeur $\left\lceil \frac{\sum_{i \in U} d_i}{Q} \right\rceil$ est exactement le nombre d'anneaux-étoiles nécessaires pour desservir tous les clients de U.

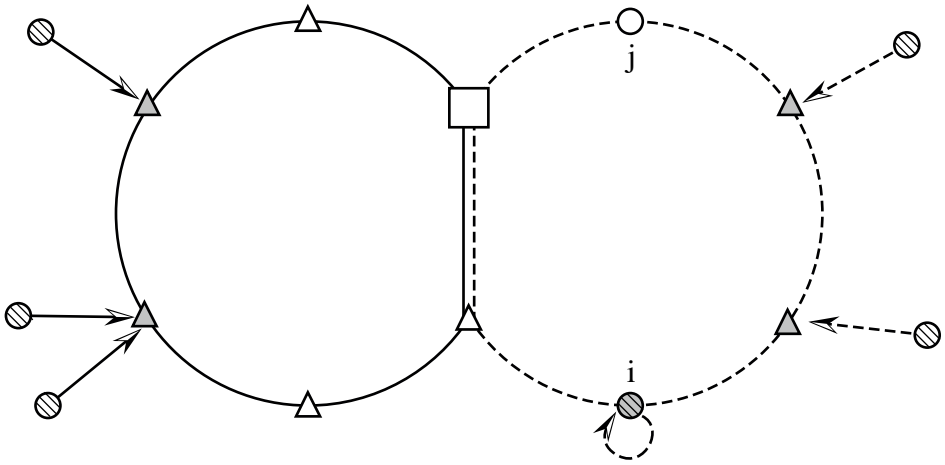


FIG. 1.1 – *Deux anneaux-étoiles non-disjoints*

Le cas particulier où le graphe ne comporte qu'un seul dépôt sera noté 1-γ-RSP. Le cas particulier où toutes les arêtes ont une capacité unitaire sera noté k-1-RSP. Enfin, le cas particulier où les deux conditions précédentes sont vérifiée sera noté 1-1-RSP.

Chapitre 2

Fiabilité dans les réseaux : état de l'art

Dans ce chapitre, nous proposons un état de l'art autour de la fiabilité des réseaux de télécommunications, c'est-à-dire de leur capacité à résister aux pannes. Il s'agit principalement d'aborder les problèmes de conception de réseaux fiables, c'est-à-dire de la construction d'un réseau de moindre coût respectant des contraintes permettant la fiabilité. Cet état de l'art est orienté en direction de la résolution de ces problèmes par programmation mathématique.

Dans un premier temps, nous présentons la littérature consacrée au vaste problème dit de conception de réseaux fiables selon le modèle de Winter, puis à celle des réseaux hiérarchiques. Enfin, nous nous intéressons aux problème de couverture d'un graphe par des anneaux, ce qui nous amène à présenter les problèmes de tournées de véhicules, et en particulier le CVRP dont la structure est proche du k-γ-RSP. Enfin, nous faisons un état de l'art détaillé des problèmes concernant les structures en anneaux-étoiles.

2.1 Réseaux fiables

La conception de réseaux fiables à coûts réduits a une importance cruciale dans l'industrie des télécommunications. La mise en place de la technologie des fibres optiques a permis une augmentation du débit des flux d'informations. En conséquence, une panne dans ces réseaux peut avoir de lourdes conséquences. Le coût d'un réseau est essentiellement basé sur la longueur totale des câbles qui le composent. Ainsi concevoir des réseaux optiques très denses, et donc très fiables, peut être très coûteux pour les opérateurs. Cependant, diminuer la densité d'un réseau diminuera son coût mais aussi sa fiabilité. Prenons l'exemple d'un réseau pour lequel il n'existe qu'un seul chemin

permettant de transférer des information entre deux terminaux du réseau. Si une liaison de ce chemin vient à défaillir, le trafic entre les deux terminaux ne peut alors plus être assuré. Le réseau se retrouve ainsi déconnecté.

La fiabilité d'un réseau s'exprime généralement en termes de connexité : plus un terminal a de l'importance, plus il doit être connecté au reste du réseau. Ainsi, la défaillance d'un lien n'entraine pas l'isolement de ce terminal. Pour cela, on associe à chaque noeud du réseau un *degré de connexité*. Celui-ci exprime le nombre de liaisons minimum qui doivent connecter le noeud au réseau final. Ainsi on assure entre chaque paire de sommets, en fonction de leur importance, un certain nombre de chemins de routage.

Le *problème de conception d'un réseau fiable* est donc de déterminer les liaisons à installer dans le réseau afin de respecter les conditions de fiabilité tout en minimisant le coût total du réseau.

La littérature propose trois grands cadres de topologie de réseaux de télécommunications suivant l'échelle géographique à prendre en compte et nous proposons dans cette thèse un quatrième cadre:

- A l'échelle d'un pays ou d'une région, les points à interconnecter sont des villes qui, selon leur taille respective, ont des degrés de connexité différents : le modèle approprié est alors le modèle de Winter qui consiste à déterminer un réseau minimum en fonction uniquement de ces degrés de connexité. L'exemple de la figure 2.1-a propose par exemple un réseau où toutes les villes ont un degré de connexité 2 (c'est-à-dire où toutes les villes sont reliés au réseau par 2 chemins disjoints).
- A l'échelle d'une ville, on peut désirer concevoir un réseau pour une grande quantité de clients (accès internet pour des particuliers par exemple) : le modèle utilisé est celui des réseaux hiérarchiques qui sont constitués d'un réseau-coeur fiable et coûteux, auquel sont reliés les clients par un réseau d'accès moins fiable et moins coûteux. Par exemple, la figure 2.1-b propose un réseau où le réseau coeur est un anneau auquel les clients sont reliés uniquement par un chemin.
- A l'échelle d'un quartier ou d'une ville, on peut vouloir relier plusieurs clients en prenant en compte, contrairement aux deux cas précédents, leur demande en bande-passante (un réseau reliant des immeubles d'un quartier d'affaire par exemple) : une solution est alors de couvrir les clients par des anneaux disjoints de capacité bornée. Ce problème d'optimisation est alors exactement le CVRP. La figure 2.1-c illustre ce cas par un réseau où plusieurs anneaux passent par un même dépôt.
- A l'échelle d'un quartier ou d'une ville, on peut vouloir relier plusieurs clients en

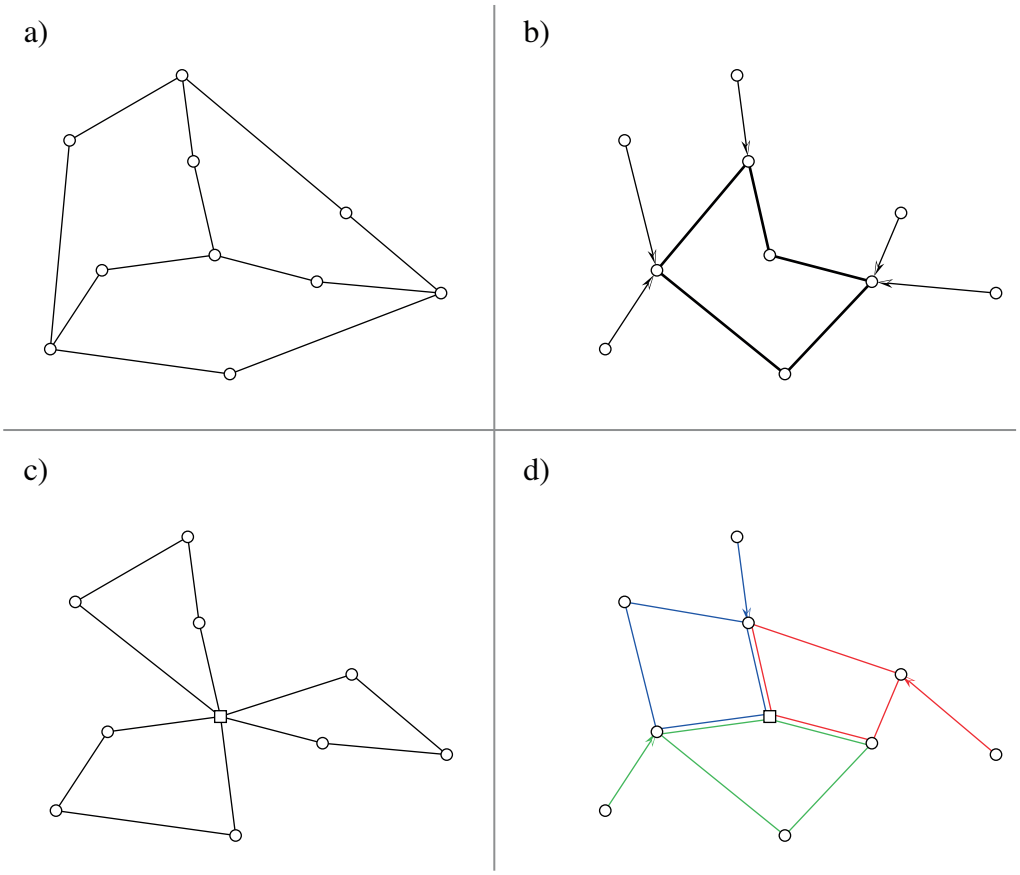


FIG. 2.1 – *Quatre catégories de réseaux fiables*

prenant en compte leur demande en bande passante et leur degré de connexité (un réseau reliant plusieurs immeubles d'affaires disséminés dans une ville par exemple) : la solution utilisée est celle proposée par les réseaux SDH de deuxième génération. Nous montrons dans le chapitre 3 que ce problème se ramène au recouvrement d'un graphe par des anneaux-étoiles à capacité bornée (k-γ-RSP). La figure 2.1-d donne un tel réseau où l'on peut remarquer que les anneaux ne sont pas nécessairement disjoints.

2.2 Fiabilité d'un réseau selon le modèle de Winter

Un réseau peut être représenté par un graphe, orienté ou non, où chaque noeud du réseau correspond à un sommet du graphe et un lien entre deux noeuds du réseau est représenté par une arête ou un arc du graphe. Le problème de conception d'un réseau

fiable, introduit par Winter [103, 101, 102] peut être exprimé de la manière suivante :

Problème Soit $c : E \to \mathbb{R}$ une fonction qui associe à chaque arête e de R son coût de construction $c(e)$. Soit $R = (r_{ij})$ une matrice carrée d'entiers non négatifs de taille $|V|$. Trouver un sous-graphe de G de coût minimum tel qu'entre chaque paire de sommets $(i,j) \in V$, il existe au moins r_{ij} chaînes arête-disjointes (ou sommet-disjointes) entre i et j.

Ce problème a été considéré plus tard dans un cadre légèrement plus restreint par Grötschel, Monma et Stoer [61, 58, 57, 59, 60, 85]. Soient $G = (V,E)$ un graphe et $c : E \to \mathbb{R}$ une fonction coût associée aux arêtes. On associe à chaque sommet $v \in V$ un entier $r(v)$ appelé *type de connexité*. On dit qu'un graphe $H = (V,F)$ vérifie les conditions d'arête-fiabilité (*resp.* sommet-fiabilité) si pour toute paire de sommets $(i,j) \in V$, il existe au moins $r(i,j) = \min(r(i),r(j))$ chaines arête-disjointes (*resp.* sommet-disjointes). Le problème de conception d'un réseau fiable consiste alors en la recherche d'un sous-graphe de G de coût minimum vérifiant les conditions de fiabilité par rapport à r.

Plusieurs cas particuliers de ce problème ont été largement étudiés. Si $r(v) = 1$ pour tout sommet $v \in V$, le problème est équivalent au problème de l'arbre couvrant de poids minimum. Ce dernier peut être résolu en temps polynomial en utilisant un des algorithmes bien connus de Kruskal [74] ou de Prim [91]. Si $r(v) = 1$ pour exactement deux sommets s et t de V et $r(s) = 0$ pour tous les autres, alors le problème n'est rien d'autre que celui du plus court chemin entre s et t. Ce problème est NP-difficile dans le cas général mais peut cependant être résolu en temps polynomial quand les coûts sont non-négatifs avec, par exemple, l'algorithme de Dijkstra [36]. Suurballe et Tarjan [99, 100] ont quant à eux résolu le problème lorsque $r(s) = r(t) = k$ avec $k \geq 2$ et $r(v) = 0$ pour les autres sommets.

Menger [83] a donné la relation entre le nombre de chemins arête-disjoints et la cardinalité des coupes dans un graphe G. Cette relation est donnée par le théorème suivant.

Théorème 2.2.1 *(Menger) Dans un graphe G, il n'existe pas de coupe de cardinalité inférieure strictement à k déconnectant deux sommets s et t si et seulement s'il existe au moins k chaines arête-disjointes entre s et t.*

Grâce au théorème 2.2.1, le problème de conception de réseaux arête-fiables peut être

décrit par un PLNE. Pour ce faire, introduisons d'abord quelques notations.

$$\begin{aligned} r(W) &= max\{r(u) : u \in W\} && \forall W \subseteq V, \\ con(W) &= max\{r(u,v) : u \in W,\, v \in \bar{W}\} \\ &= min\{r(W),\, r(\bar{W})\} && \forall W \subsetneq V,\, W \neq \emptyset. \end{aligned}$$

Le problème de conception de réseaux arête-fiables est alors équivalent au PLNE suivant.

$$\begin{aligned} &\text{Minimiser } cx \\ &x(e) \geq 0, && \forall e \in E, \\ &x(e) \leq 1, && \forall e \in E, \\ &x(\delta(W)) \geq con(W), && \forall W \subsetneq V,\, W \neq \emptyset, \\ &x(e) \in \{0,1\} && \forall e \in E. \end{aligned} \tag{2.1}$$

Notons que la séparation des contraintes de coupe (2.1) se réduit à un problème de flot maximal dans un graphe [30].

Un graphe $G = (V,E)$ est dit *k-arête (resp. k-sommet) connexe* s'il existe au moins k chaînes arête-(*resp.* sommet-)disjointes entre chaque paire de sommets de V. Par conséquent, si $r(v) = k$ pour tout sommet $v \in V$, alors un sous-graphe k-arête (*resp.* sommet) connexe de G sera arête (*resp.* sommet) fiable.

Dans [25], Chopra étudie le cas où k est impair quand plusieurs copies d'une arête peuvent être utilisées. Il donne la caractérisation du polyèdre associé aux graphes outerplanar. Ce polyèdre a été étudié précédemment par Cornuéjols, Naddef et Fonlupt dans [27]. Ils donnent la caractérisation du polyèdre associé quand le graphe est série-parallèle et $k = 2$. Dans [19], Didi Biha et Mahjoub donne la caractérisation du polyèdre lorsque le graphe est série-parallèle et $k \geq 3$. Dans [43], Fonlupt et Mahjoub étudient les points extrêmes fractionnaires de la relaxation linéaire du polytope du Sous-graphe 2-arête-connexe. Ils obtiennent une caractérisation des graphes pour lesquels la relaxation linéaire du problème est entière. Didi Biha et Mahjoub [20] étendent certains des résultats de Fonlupt et Mahjoub [43] au cas où $k \geq 3$ et introduisent des opérations de réduction de graphe. Dans [17], Bendali et al. proposent de nouvelles inégalités du polyèdre du sous-graphe k-connexe ainsi qu'un algorithme de Branch-and-Cut basé sur des techniques de réduction de graphes.

Comme le type de connexité égal à 2 s'est révélé être une condition de connexité adéquate pour la plupart des réseaux de télécommunications actuels, le problème du sous-graphe 2-arête-(ou 2-sommet-)connexe s'est vu porter beaucoup d'attention. Dans

[4], Baïou et Mahjoub étudient le polytope du Steiner sous-graphe 2-arête-connexe puis généralisent leurs résultats au polytope du Steiner sous-graphe k-arête-connexe avec k pair. Dans [15], Barahona et Mahjoub donnent la caractérisation du polytope du Sous-graphe 2-arête-connexe pour la classe des graphes de Halin. Kerivin et al. [71] décrivent une nouvelle famille d'inégalités qui généralise les inégalités de F-partition introduites dans [81].

Notons également qu'une large variété d'études a aussi été consacrée au cas où la taille des chemins entre deux sommets est contrainte, voir par exemple le survey de Kérivin et Majhoub [70]. Étant donnés un graphe non-orienté $G = (V,E)$, une fonction poids, un ensemble de demandes $D \subseteq V \times V$ et deux entiers $k \geq 2$ et $L \geq 2$, le problème de recherche d'un sous-graphe k-arête-connexe avec contrainte de borne consiste à rechercher un sous-graphe de G de poids minimal tel que pour chaque paire $(s,t) \in D$, il existe au moins k chemins arête-disjoints de longueur au plus L entre s et t. Il s'agit d'une généralisation du problème du sous-graphe k-arête-connexe qui correspond au cas particulier où $L = |V| - 1$ et $D = V \times V$. Dahl et al. [31] analysent le polytope associé au problème du sous-graphe k-arête disjoints où les longueurs des chemins sont limitées. Dans [18], Bendali et al. s'intéressent au cas particulier où $L \leq 3$ et donnent une caractérisation complète du polyèdre dans le cas où l'on se limite à une seule paire (s,t) (ce qui généralise les travaux de [31]). Dernièrement, Botton et al. [23] proposent une décomposition de Benders pour ce problème qui a permis de résoudre des instances de tailles importantes.

En fait, ces contraintes de borne proviennent de la nécessité d'avoir un chemin alternatif de routage qui ne soit pas trop long. Une autre façon de la mettre en œuvre est par exemple possible d'exiger que chaque arête appartienne à un cycle de longueur bornée. Ainsi Fortz et al. [44, 45] étudient le problème de conception d'un réseau 2-sommet-connexe avec des cycles bornés. Ils décrivent plusieurs classes de facettes et proposent un algorithme de Branch-and-Cut. Fortz et al. [46] s'intéressent à un problème similaire dans lequel le réseau est lui 2-arête-connexe.

2.3 Fiabilité dans les réseaux hiérarchiques

A l'échelle d'une agglomération, les réseaux ne peuvent être intégralement fiable. En effet, il serait très coûteux de relier chaque particulier avec un degré de connexité 2 par exemple. On leur préfère des réseaux à deux-couches, aussi appelés *réseaux hiérarchiques*. Voir par exemple les surveys [54] et [72]. Dans de tels réseaux, le trafic provenant des terminaux (les clients) est communiqué à travers des *réseaux d'accès*

jusqu'aux concentrateurs (des multiplexeurs) qui sont interconnectés en formant un *réseau cœur*, appelé aussi squelette (*backbone*). Le trafic traverse alors le réseau cœur pour atteindre le réseau d'accès de son terminal de destination. Le réseau cœur transporte donc un large volume de trafic à haut début et prend le rôle principal dans la transmission de données. C'est pourquoi il est essentiel que ce réseau cœur possède une topologie de réseau fiable. Klincewicz [72] propose une classification des problèmes de conception de réseaux basée sur la topologie des réseaux d'accès et des réseaux cœurs. Typiquement la topologie d'un réseau d'accès inclut l'étoile, les arbres et les anneaux. De même, le réseau cœur peut être un anneau ou un graphe fortement maillé. Dans la classification de Klincewicz, on nomme alors les problèmes selon l'écriture topologie du réseau cœur / topologie du réseau d'accès.

Ainsi Pirkul et Nagarajan [90] et Lee et al. [78] ont étudié le problème arbre/étoile alors que Gavish [51] s'est intéressé à la variante étoile/arbre, Chardaire et al. [24] au problème étoile/étoile et Current, Pirkul et al. [29] au chemin/chemin. Néanmoins toutes ces topologies ne désignent en rien des réseaux fiables.

En revanche, s'intéressant aux problèmes à réseau cœur fortement maillé, Labbé et al. [76] considèrent le problème de conception de réseaux complet/étoiles qui peut-être vu comme un problème de localisation. Ils étudient le polyèdre associé et proposent un algorithme de Branch-and-Cut. Considérant plus en avant les aspects réseaux de télécommunications, Fouilhoux et al. [47] étudient le problème du 2-arête-connexe/étoile en proposant également une analyse polyèdrale du problème, des opérations de réduction et un algorithme de Branch-and-Cut. Enfin, le problème anneau/étoile a naturellement été abordé et, ce problème étant fortement en lien avec notre sujet, nous insisterons en détail sur cette littérature dans la section 2.5.

2.4 Recouvrement d'un graphe par des anneaux (ou des tournées de véhicules)

Rappelons que pour le CVRP nous considérons un graphe $G = (V,E)$ non orienté où V est un ensemble de $n+1$ sommets, $V = \{0,1,\ldots,n+1\}$; le sommet 0 est appelé dépôt et $V_c = \{1,\ldots,n+1\}$ est l'ensemble des n clients. Chaque client i de V_c demande une quantité d_i de produits stockés au dépôt 0. Une route réalisable est un cycle élémentaire du graphe G passant par le sommet dépôt 0 et telle que la demande cumulée des clients visités soit inférieure ou égale à la capacité d'un véhicule Q. Nous voulons déterminer au plus m routes telles que tous les clients soient desservis et telles que le coût total des routes soit minimal.

Le CVRP s'inscrit naturellement dans la littérature des problèmes de tournées de véhicules. En fait, il se ramène alors à la couverture d'un graphe par des anneaux disjoints. En effet, une astuce classique pour les problèmes de tournées de véhicules est de travailler sur un graphe complet contenant le même ensemble de sommets mais sur lequel chaque arête représente le plus court chemin entre les deux extrémités. Une fois une solution obtenue, il est très facile de déterminer le chemin réel de chacun des véhicules. Cela implique deux avantages très largement utilisés dans les études dédiées aux tournées de véhicules : d'une part cela permet de réduire le CVRP à une recherche de cycles élémentaires disjoints et d'autre part, cela permet de travailler sur un graphe dont la fonction coût vérifie les inégalités triangulaires.

De même, la recherche d'une couverture de coût minimum d'un graphe par des anneaux disjoints correspond à la recherche d'un réseau de télécommunications fiables où les clients doivent appartenir à des anneaux disjoints de capacité limitée.

En revanche, on peut remarquer que l'astuce évoquée pour le problème de tournées de véhicules ne peut être malheureusement pas être utilisée pour concevoir un réseau en anneaux non nécessairement disjoints. En effet, la topologie du chemin de chaque véhicule n'est pas contrainte et en particulier rien ne garantit qu'elle sera en anneau. À l'inverse, dans le problème de conception de réseaux en anneaux (ou anneaux-étoiles), la topologie en anneau doit être vérifiée dans le graphe d'origine puisqu'elle est essentielle à la fiabilité du réseau. De plus, dans le cas d'anneaux non-disjoints, il faut respecter une contrainte sur la capacité des câbles (qui n'a pas d'équivalent en tournée de véhicules). Ces deux raisons font que pour le problème de conception de réseaux en anneaux-étoiles, nous ne pourrons pas émettre l'hypothèse que notre graphe est complet ou que la fonction de coût vérifie les inégalités triangulaires.

Voici néanmoins la littérature du CVRP qui sera fortement utilisée dans ce document.

2.4.1 Formulation à 2 indices

Dans [77], Laporte et al. proposent une formulation, appelée formulation à 2 indices pour le CVRP. Soit x_e une variable binaire associée à l'arête $e \in E$, la formulation à 2

indices est la suivante :

$$
\begin{aligned}
Min \sum_{e \in E} c_e x_e & \\
\sum_{e \in \delta(i)} x_e = 2 \quad & \forall i \in V_c \qquad (2.2)\\
\sum_{e \in \delta(S)} x_e \geq 2k(S) \quad & \forall S \subset V_c \qquad (2.3)\\
\sum_{e \in \delta(0)} x_e = 2m & \qquad (2.4)\\
x_e \in \{0,1\} \quad & \forall e \in E.
\end{aligned}
$$

Les contraintes (2.4) imposent que m véhicules sortent puis rentrent au dépôt 0 et les contraintes (2.2) assurent que chaque client soit visité. Les contrainte (2.3) sont les *contraintes de capacité* dans lesquelles $k(S)$ est une évalution inférieure du nombre de tournées nécessaires pour desservir tous les clients de S. Dans [77], Laporte et al. définissent $k(S)$ comme la solution d'un problème de bin packing défini sur un ensemble S d'items de taille d_i, $i \in S$ et des bins de capacité Q. Dans [28], Cornuéjols et Harche montrent que la formulation reste valide lorsque l'on exprime $k(S)$ plus simplement en posant $k(S) = \left\lceil \frac{\sum_{i \in S} d_i}{Q} \right\rceil$, on parle alors de contraintes de *capacité arrondie*. Ils montrent également que déterminer la dimension du polyèdre associé à cette formulation est un problème NP-complet. Ils étudient alors un dominant de ce polytope défini comme la relaxation du fait qu'une arête ne puisse être utilisée qu'une fois et donnent des conditions nécessaires et suffisantes pour que les inégalités de la formulation définissent des facettes de ce dominant.

Il existe d'autres formulations à nombre polynomial de variables. Citons par exemple la formulation à 3 indices proposée par Fisher et Jaikylar [42] et par Fischetti et al. dans [41] ou la formulation flot à une commodité de Gavish et Graves [52] étudiée par la suite par Gouveia dans [55]. Une comparaison de la qualité de la relaxation linéaire de ces différentes formulations peut être trouvée dans [80].

2.4.2 Formulation Set Partitionning

Considérons maintenant l'ensemble Ω des toutes les routes réalisables. La formulation set partitionning (SP) a été proposée par Balinski et Quandt [11]. Soit c_r le coût de la route $r \in \Omega$, et b_i^r un coefficient qui indique si la route r passe par le sommet $i \in V_c$ ($b_i^r = 1$) ou non ($b_i^r = 0$). En associant une variable binaire θ_r à chaque route

$r \in \Omega$ nous obtenons le modèle suivant :

$$\begin{aligned} (SP) \quad & \min \textstyle\sum_{r\in\Omega} c_r\theta_r & & \\ s.t. \quad & \textstyle\sum_{r\in\Omega} b_i^r\theta_r = 1 & \forall i \in V_c, & (2.5) \\ s.t. \quad & \textstyle\sum_{r\in\Omega} \theta_r \leq m & & \\ & \theta_r \in \mathbb{N} & \forall r \in \Omega. & (2.6) \end{aligned}$$

Cette formulation est valide pour tout type de fonction coût, cependant si les coûts vérifient les inégalités triangulaires, la formulation SP peut être convertie en une formulation set covering tout en conservant la réalisabilité des solutions optimales. Dans ce cas les égalités 2.5 peuvent être remplacées par les inégalités suivantes

$$\sum_{r\in\Omega} b_i^r\theta_r \geq 1, \forall i \in V_c.$$

Cette formulation ayant un nombre exponentiel de variables, sa résolution nécessite l'utilisation d'une procédure de génération de colonnes. Pour les problèmes de tournées de véhicules, le problème auxiliaire associé à cette formulation revient à rechercher un plus court chemin élémentaire avec contraintes de ressources (ESPPRC).

Dans [33] Desrochers et al. donnent la première mise en oeuvre d'un algorithme de génération de colonnes pour le VRP avec fenêtre de temps (VRPTW). À cette occasion, ils utilisent le fait que la recherche d'un plus court chemin avec contrainte de ressource admet un algorithme de résolution pseudo-polynomial lorsque la contrainte d'élémentarité est relâchée. Dans ce cas l'ensemble de tournées réalisables Ω contient également les routes non élémentaires et le coefficient b_i^r devient le nombre de fois qu'un sommet v_i est visité dans la route r_k. Malgré cette modification, le modèle fournit toujours une solution optimale pour le VRPTW lorsque la fonction de coût associée aux arêtes vérifie les inégalités triangulaires. Cela permet d'accélérer la procédure de génération de colonnes mais baisse la qualité de la relaxation linéaire de la formulation.

Desrochers et al. vont donc plus loin et interdisent les chemins contenant des 2-cycles, c'est-à-dire des cycles composés de 2 arêtes. Le coût en temps de calcul pour prendre en compte cette interdiction lors de la résolution du problème auxiliaire reste assez faible. Feillet et al. [39] évaluent l'efficacité du maintient de la contrainte d'élémentarité dans le sous-problème. Cela peut être fait en ajoutant de nouvelles ressources et règles de dominances à l'algorithme de Desrochers et al. [33]. Les résultats expérimentaux montrent que si l'amélioration de la relaxation linéaire peut parfois être décisive, elle ne justifie pas toujours le coût supplémentaire en temps de calcul.

Poursuivant l'idée de suppression des 2-cycles, Irnich et Villeneuve [66] étudient la suppression des k-cycles. Ils évaluent l'impact sur la relaxation linéaire pour $k = 3$ et $k = 4$ et concluent que le gap peut être réduit de manière significative à un coût raisonnable en temps de calcul.

Les travaux récents convergent vers un compromis proposé à la fois par Boland et al. [21] et Righni et Salani [93]. Ces auteurs proposent de résoudre dynamiquement l'ESPPRC en ajoutant progressivement la contrainte de chemin élémentaire. Le SPPRC est résolu dans un premier temps. Si l'ensemble de route de coût réduit négatif est non vide mais ne contient que des routes non-élémentaires, une contrainte de visite unique est ajoutée pour un sommet et l'algorithme est relancé. La résolution s'arrête lorsqu'une route élémentaire est trouvée ou quand il est prouvé qu'aucune route de coût réduit négatif n'existe. Une autre proposition par Desaulniers et al. [32] est de rechercher une borne inférieure pour laquelle seul un sous-ensemble de clients, déterminé dynamiquement, est soumis à la contrainte de visite unique. Dans [10] Baldacci et al. introduisent un nouveau type de relaxation, différente de la relaxation de l'élémentarité, appelée *ng*-route.

2.4.3 Ajout d'inégalités valides

L'ajout d'inégalités valides permet d'améliorer la qualité de la relaxation linéaire d'une formulation et plusieurs familles d'inégalités ont été utilisées dans la formulation set partitionning.

Dans [73] Kohl et al. introduisent les *k-path cuts*. Si la séparation de ces contraintes est difficile, les coûts duaux associés sont facilement pris en compte dans le problème auxiliaire car ils peuvent être répercutés sur les coûts des arêtes. Kohl et al. limitent leur étude au cas où $k \leq 2$. Les cas où $k = 3$ a été étudié plus tard par Cook et Rich [26]. Ces contraintes ont été récemment généralisées par Desaulniers et al. [32]

Dans [49], Fukasawa et al. proposent une formulation dite *robuste* qui combine la formulation à 2 indices et la formulation set partitionning grâce aux contraintes suivantes :

$$x_e = \sum_{r \in \Omega} b_e^r.$$

Ainsi, toutes les inégalités valides introduites dans la formulation à 2 indices pour le CVRP peuvent être utilisées dans la formulation set partitionning. Fukasawa et al. ajoutent alors à leur formulation toutes les contraintes présentées dans [62]. De même, dans [6] Baldacci et al. utilisent les contraintes de capacité (2.3) et toutes les contraintes du package CVRSEP [79] à savoir les inégalités de multistar et de multistar partielle. Ils introduisent également en partie des inégalités du polytope du Set partitionning à savoir les contraintes de clique.

Plus récemment, d'autres études ont tenté d'exploiter les inégalités valides du polytope du Set partitionning. Dans [67], Jepsen et al. introduisent les inégalités de subset-row qui sont des inégalités de Chvatal-Gomory de rang 1. Les auteurs montrent que le

problème de séparation de ces contraintes est NP-complet et donnent les modifications à apporter à l'algorithme de programmation dynamique pour les prendre en compte. Dans sa thèse [98], Spoorendonk étudie aussi la possibilité d'utiliser les contraintes de clique dans une telle formulation. Il propose plusieurs représentations d'une clique et les modifications à apporter à l'algorithme de programmation dynamique pour leur prise en compte.

2.5 Recouvrement par des anneaux-étoiles

La première étude polyédrale de la structure en anneau-étoile est attribuée à Labbé et al. [75]. Étant donné un graphe $G = (V,E)$, les auteurs recherchent un cycle élémentaire passant par le dépôt $s \in V$ de manière à minimiser deux coûts : la longueur du cycle et le coût d'affectation des sommets n'appartenant pas au cycle au plus proche sommet du cycle. Dans cette étude, les auteurs déterminent la dimension du polyèdre associé à leur formulation et proposent plusieurs familles d'inégalités valides, principalement issues du polytope des circuits [16]. Ils donnent également les conditions nécessaires et suffisantes pour que leurs inégalités définissent des facettes du polytope. Enfin, ils donnent le détail de l'implémentation d'un algorithme de Branch-and-Cut qui se montre capable de résoudre optimalement des instances allant jusqu'à 300 sommets.

Peu avant, Lee, Chiu et Sanchez [96] avaient défini le Steiner Ring-Star Problem dans lequel seul un sous-ensemble de sommets doit être connecté au cycle. Dans cette étude ils développent un algorithme de Branch-and-Cut qui est testé sur des instances allant jusqu'à 100 sommets.

Poursuivant l'étude de Labbé et al., Kedad-Sidhoum et Nguyen [69] proposent une nouvelle formulation basée sur les chaines. Pour ce faire ils définissent à partir de $G = (V,E)$ un graphe $G' = (V \cup \{t\},E')$ où t est une copie du sommet s, et $E' = E \cup \{s,t\} \cup \{tu : su \in E\}$. À chaque cycle de G passant par s correspond alors une st-chaine dans G' et réciproquement. Ils mènent une étude sur le polyèdre associé à leur formulation et en donnent la dimension. Ils montrent les correspondances entre leur formulation et celle de Labbé et al., puis introduisent les inégalités de st-chain-blossom et donnent des conditions nécessaires et suffisantes pour que ces dernières définissent des facettes du polyèdre. Enfin, ils mènent une campagne expérimentale indiquant que leur formulation donne de meilleurs résultats que celle de Labbe et al.[75] grâce à l'ajout des inégalités de st-chain-blossom.

Le 1-1-*RSP* est introduit par Baldacci et al. dans [8] sous le nom de *Capacitated m-ring-star problem*. Ils proposent deux formulations inspirées de travaux précédents pour le TSP ou le CVRP : la formulation à 2 indices et la formulation flot à 2 com-

modités. Ils montrent que ces deux formulations sont incomparables en terme de qualité de relaxation linéaire. Ils proposent ensuite des améliorations des contraintes de capacité présentes dans la formulation à 2 indices, ainsi qu'une nouvelle famille de contraintes pour la formulation flot. Ils donnent ensuite une description détaillée de deux algorithmes de Branch-and-Cut développés pour ces formulations. Ils détaillent les procédures, exactes ou heuristiques, de séparation des contraintes, ainsi que la règle de branchement et d'exploration de l'arbre. Ils comparent alors les deux formulations sur des instances générées aléatoirement, montrant que les meilleures performances sont obtenues avec la formulation flot. Ils parviennent ensuite à résoudre des instances réelles du problème allant jusqu'à 137 clients.

Hoshino et al. [63] proposent la première formulation à nombre exponentiel de variables pour le 1-1-RSP. L'élément clé de l'implémentation est le problème auxiliaire qui consiste à rechercher un anneau-étoile de coût minimal dans un graphe à coûts quelconques. Ce problème étant NP-complet, ils en étudient plusieurs relaxations consistant à autoriser la répétition de sommets, que ce soit au sein du cycle ou au sein des affectations. Un *anneau-étoile relâché* est alors un couple $R = (E(R),L(R))$ où

- $E(R)$ est un cycle non nécessairement élémentaire de G passant par le dépôt,
- $L(R)$ est un sous-ensemble d'arcs tel que pour chaque arc (i,j) de $L(R)$, j est un sommet du cycle $E(R)$,
- $U(R) = \{i \in U : \exists (i,j) \in L(R)\}$ est l'ensemble de clients desservis par R,
- un sommet de $U(R)$ peut être affecté à plusieurs sommets connecteurs du cycle $E(R)$.

Reprenant les idées appliquées au CVRP [33, 66], ils montrent que le problème de recherche d'un anneau-étoile relâché admet un algorithme de résolution pseudo-polynomial. Une remarque extrêmement importante est que la validité de l'algorithme est conditionnée par le respect des inégalités triangulaires dans le graphe. En effet, si les inégalités triangulaires sont satisfaites, nous avons l'assurance qu'une solution optimale de la formulation ne contiendra que des anneaux-étoiles réalisables. Dans le cas contraire cependant, la solution fournie par l'algorithme peut contenir des cycles non-élémentaires.

Dans [65], Hoshino et al. améliorent l'efficacité de leur algorithme en y ajoutant un certain nombre d'inégalités valides : les inégalités de 2-connexité, les inégalités multi-star et les inégalités de capacité arrondie. Toutes ces contraintes peuvent facilement être prises en compte lors de la résolution du problème auxiliaire car les coûts duaux associés à ces dernières peuvent être répercutés sur le coût des arcs et des arêtes du graphe. Ils comparent ensuite leur algorithme avec celui de Baldacci et al. sur les instances proposées dans [8]. L'ajout d'inégalités valides permet d'améliorer l'efficacité de l'algorithme de Branch-and-Price [63] et l'algorithme de Branch-and-Cut-and-Price

résultant est le plus efficace à ce jour pour le 1-1-RSP.

Les méthodes exactes de résolution ne permettant pas de résoudre des instances correspondant à des réseaux de très grande taille, plusieurs méthodes de résolution heuristiques ont été proposées. Dans [35], Dias et al. proposent une méta-heuristique pour le RSP. Ils définissent plusieurs voisinages qui sont ensuite utilisés dans une recherche par voisinages variables généralisée (GVNS) lancée à partir d'une première solution obtenue par une procédure GRASP [40]. Leurs travaux se terminent par une importante campagne expérimentale et se comparent avec succès avec une méthode pourtant plus élaborée proposée par Morena et al. [89].

Dans [82], Mauttone et al. proposent la première méta-heuristique pour le 1-1-RSP. Ils combinent une procédure GRASP et une méthode taboue. Ils mènent une campagne expérimentale sur les instances de Baldacci et al. [8] sur lesquelles leur heuristique montre de bons résultats à l'exception de quelques instances pour lesquelles le gap peut monter jusqu'à 5,35%.

Dans [95], puis dans [86], Salari et al. proposent une *matheuristique* [22] pour le 1-1-RSP. Il s'agit d'une méta-heuristique par voisinages variables généralisée (VNS) à laquelle est ajoutée une phase de déconstruction / reconstruction de solutions. La reconstruction d'une solution est un problème secondaire qui est résolu par programmation linéaire.

Baldacci et Dell'Amico [7] ont mené la seule étude connue du cas multi-dépôts k-1-RSP. Ils donnent une formulation linéaire en nombres entiers adaptée de leurs travaux pour le cas mono-dépôt [8], ce qui leur permet de fournir une borne inférieure pour ce problème. Ils proposent ensuite une heuristique pour le cas mono-dépôt ainsi que deux procédures d'amélioration locale. Ces algorithmes leur permettent ensuite d'implémenter deux méta-heuristiques pour le cas multi-dépôts. La première consiste à identifier les différents dépôts en un dépôt artificiel, puis de résoudre l'instance mono-dépôt résultante. Dans une deuxième phase, les anneaux-étoiles construits sont affectés aux dépôts. La seconde heuristique consiste à d'abord définir pour chaque dépôt l'ensemble de clients qu'il desservira, puis à résoudre les k instances mono-dépôt résultantes. Enfin, ils appliquent une méthode taboue [53] sur les solutions obtenues grâce à trois voisinages dédiés.

Nous notons que dans toutes les études citées précédemment, le cas où les structures ne sont pas nécessairement disjointes n'a pas traité de manière exacte. À cela plusieurs raisons, d'une part la plupart des études étaient plus orientées sur des problèmes académiques, et d'autre part, comme nous le verrons dans le chapitre 4, les formulations pour le cas non-disjoint sont plus difficiles à manier et souvent moins efficaces.

Chapitre 3

Modélisation du problème

L'objectif de ce chapitre est de donner la description d'un réseau de fibres optiques utilisant la technologie dite à Hiérarchie Numérique Synchrone (Synchronous Digital Hierarchy), puis de donner une réduction du problème de conception d'un réseau SDH au problème de couverture d'un graphe par des anneaux-étoiles.

3.1 Présentation des réseaux SDH

Les réseaux SDH sont constitués d'une couche physique (appelée parfois *réseau passif*) et d'une couche logique (le *réseau actif*). La couche physique correspond à l'infrastructure : câbles, fibres optiques et boîtiers de raccordement. La couche logique est constituée des équipements électroniques et des protocoles de gestion des données circulant sur le réseau.

La section 3.1.1 donne un bref aperçu de la composante logique, la section 3.1.2 donne une description détaillée de la couche physique. Le but de ces deux sections est d'introduire le problème de conception de réseaux SDH.

3.1.1 Couche logique

Les réseaux de fibres optiques supportent de nombreuses technologies. Celle qui nous intéresse est la *hiérarchie numérique synchrone* ou *SDH*. Elle correspond au protocole *SONET* aux États Unis. L'intérêt de la SDH est la richesse des fonctions de gestion,

de surveillance, d'alarmes et d'auto-cicatrisation. Nous nous intéressons ici aux équipements de seconde génération qui permettent d'instancier la technologie SDH sur des anneaux mono-fibre. En effet, la première génération d'équipement utilisait deux fibres d'un même câble pour chaque anneau. Pour la seconde génération, deux fréquences d'émission sur une fibre optique sont utilisées (*1300 nm* et de *1550 nm*). L'une de ces fréquence est réservée à l'émission des données, l'autre à la réception, permettant ainsi la création de deux canaux distincts sur une même fibre.

3.1.1.1 Équipements électroniques

Le transfert des données se fait grâce à des trames, notées *STM-n*, dont le débit se calcule de la manière suivante :

$$D = (n * 270 * 9 * 8bits)/125\mu s = n * 155{,}520Mbits/s.$$

Par exemple, il est possible d'utiliser des trames allant de STM-1 (155 Mbits/s) à STM-16 (2.5 Gbits/s).

Les signaux à transporter d'une même trame proviennent de sources différentes, on parle alors de *service*. Un service est un flux de données à transporter d'un point à un autre. Pour faciliter leur transport ils sont accumulés dans un conteneur virtuel *VC-x* (*Virtual Container*). Les principaux conteneurs sont VC-12 (2 Mb/s), VC-3 (34 ou 45 Mb/s) et VC-4 (140 Mb/s).

Les équipements permettant le routage des données sont les *Multiplexeurs* (ou *MUX*). Ils permettent d'intégrer différents groupes de données dans un seul flux agrégé, mais également le processus inverse, c'est à dire retirer un ou plusieurs groupes de données du flux agrégé pour les rediriger. Ils peuvent être situés chez le client ou dans les dépôts. Chez le client, ils servent à retirer les données du flux de l'anneau pour les rediriger sur les équipements clients, ou à l'inverse à insérer les données du client dans le flux principal. Lorsqu'ils sont situés dans les dépôts ils assurent les échanges de données entre les différents anneaux.

3.1.1.2 Protection des connexions

La principale protection est induite par la structure en anneau du réseau. Les données transitent dans un sens, néanmoins, si la connexion vient à se rompre, la technologie SDH permet de rediriger de manière quasi-immédiate les données dans l'autre sens. La

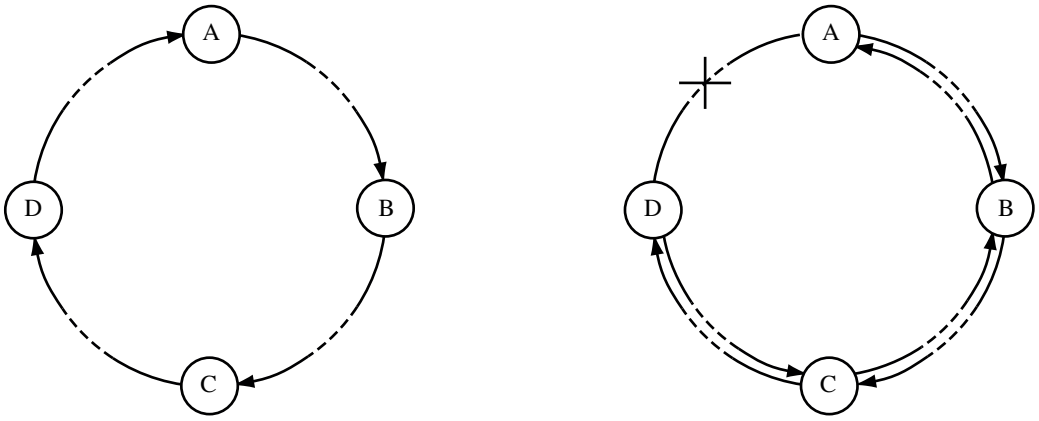


FIG. 3.1 – *Reroutage en cas de rupture d'une connexion*

structure en anneau implique qu'une rupture en un endroit n'isole pas totalement l'un des éléments (*cf.* figure 3.1).

3.1.2 Couche physique

3.1.2.1 Définitions

L'infrastructure est composée d'un ensemble de *fourreaux*, placés dans la chaussée, dans les égouts ou le métro, parfois même dans les conduites de gaz. Un fourreau est un tube (différents diamètres étant possibles) contenant un ensemble de *câbles*. Les câbles contiennent eux mêmes un ensemble de *modules* (aussi appelé parfois tubes). Chaque module est un tube contenant de deux à douze *fibres optiques* (*cf.* figure 3.2).

Un réseau de télécommunications est souvent décrit comme la somme de deux réseaux disctincts : le réseau de *transport* et le réseau de *distribution* (aussi appelé réseau d'accès). Le réseau de transport est l'ensemble des artères principales du réseau et est constitué de câbles de capacité très importante en nombre de fibres. Le réseau de distribution est constitué de câbles de capacité inférieure, déployés pour connecter les différents clients. Cependant, pour des raisons pratiques, la séparation n'est pas toujours aussi nette que cela. Il peut y avoir des clients reliés directement au réseau de transport, ou des câbles de distribution servant au transport.

L'ensemble de ce réseau constitue donc un maillage complexe, dont les centres névralgiques sont les *dépôts* (ou *nodes*). Il s'agit de bâtiments dans lesquels se trouvent les équipements électroniques à partir desquels le réseau sera géré et monitoré.

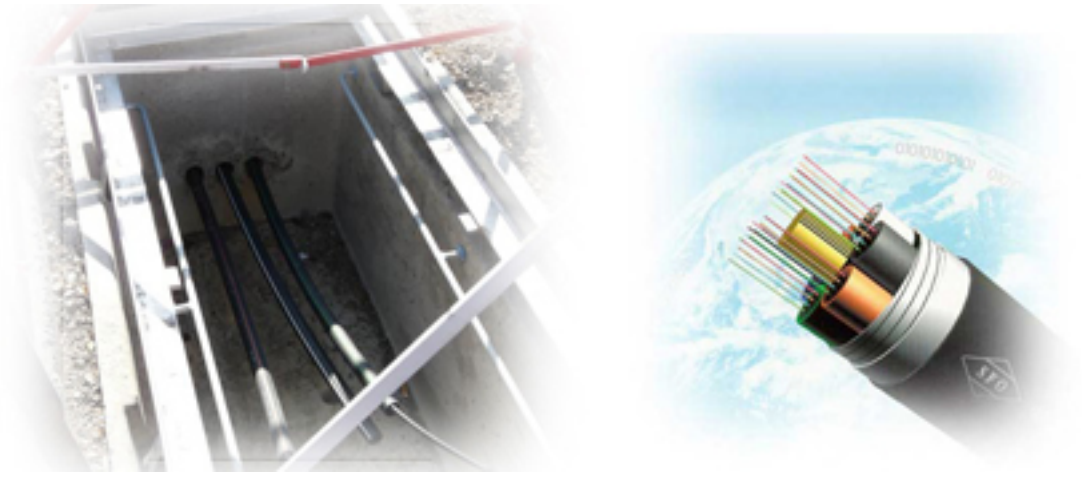


FIG. 3.2 – *Fourreau, câble et modules*

3.1.2.2 Câbles

La technologie utilisée pour les fourreaux et câbles diffère suivant l'endroit où ils sont placés. Un câble placé en égout aura besoin de protection plus importante qu'un câble déployé dans un bâtiment. A l'inverse, un câble déployé dans un bâtiment doit répondre à certaines normes (particulièrement de sécurité) inutiles pour un câble déployé en chaussée.

Les câbles sont découpés en *segments* de deux cents ou quatre cents mètres. Entre chaque segment se trouve une *boîte de jonction* (*cf.* figure 3.3) ou une longueur de câble supplémentaire (une *loop*) qui permettra l'installation d'une boîte de jonction si nécessaire. Ces boîtes permettent la liaison d'un segment à l'autre. Elles sont situées dans des *niches* (si elles se trouvent dans les égouts) ou *chambres* (si elles se trouvent dans la chaussée). Il s'agit de petits espaces aménagés permettant aux techniciens d'accéder aux boîtes de jonction et de les remonter à la surface en cas d'intervention. C'est au niveau de ces boîtes que se font tous les raccordements, que ce soit d'un segment à l'autre, ou d'un câble à un autre (du réseau de transport au réseau de distribution, ou du réseau de distribution au client par exemple).

Chaque boîte de jonction possède sept entrées pour les câbles dont deux réservées à la continuité du câble principal. Les emplacements restants peuvent servir à relier une boîte de jonction à une autre située dans la même niche pour assurer le raccordement d'un câble principal à un autre. Ils peuvent également permettre le raccordement d'un bâtiment au réseau. Un câble part alors de la boîte de jonction jusqu'au *Building Flexibility Point* (*BFP*). Il s'agit d'un boîtier situé à l'extérieur du bâtiment à relier.

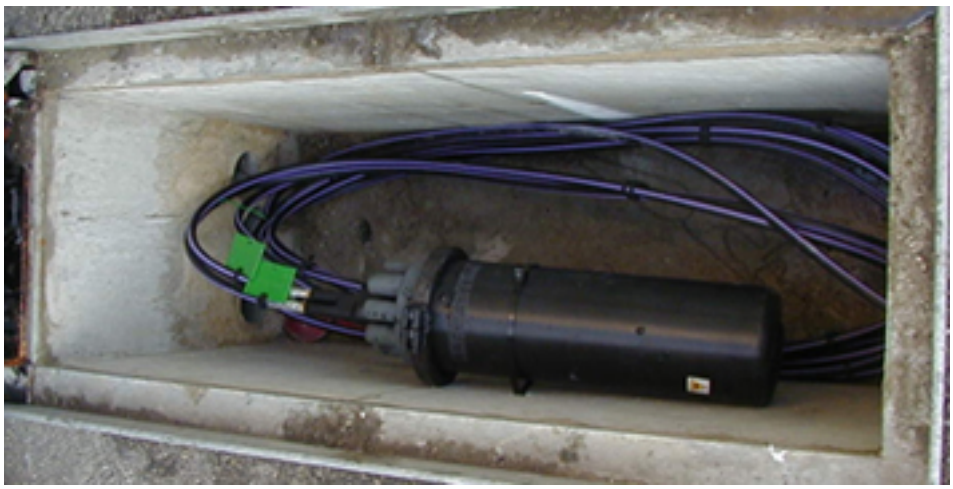


FIG. 3.3 – *Boîte de jonction placée dans une niche*

3.1.2.3 Boîtes de jonction

Les boîtes de jonction permettent le raccordement des câbles. Les fibres contenues dans les câbles qui sont reliés à la boîte sont séparées puis soudées entre elles, on parle d'*épissure*. La figure 3.3 permet de voir l'intérieur d'une chambre, dans laquelle on distingue les différents câbles pénétrant dans la boîte de jonction. Les entrées centrales sont réservées au câble principal, les emplacements périphériques aux câbles secondaires.

L'intérieur de la boîte de jonction est composé d'un certain nombre de *cassettes* (12 ou 24 suivant les modèles). La figure 3.4 permet de voir l'intérieur d'une boîte de jonction contenant 12 cassettes. On distingue les différents modules en entrée de boîte, raccordés aux différentes cassettes. Chacune des cassettes permet de faire 24 épissures. La figure 3.5 permet de voir l'intérieur d'une cassette. On y distingue les différentes fibres, enroulées dans la cassette, et les emplacements de raccordement (la moitié seulement étant utilisée sur l'image).

S'il est nécessaire de raccorder deux fibres reliées à deux cassettes différentes, on utilise alors une fibre de liaison. Le raccordement utilisera donc 2 épissures au lieu d'une.

Certaines boîtes de jonction peuvent être saturées par le câble principal. C'est notamment le cas pour les boîtes de jonction servant à relier entre eux les segments des câbles principaux du réseau de transport. Ces câbles ont une capacité de 288 fibres et les boîtes de jonction entre deux segments ont une capacité de 288 épissures. La continuité du câble nécessite donc la totalité de la capacité de la boîte de jonction.

FIG. 3.4 – *Boîte de jonction ouverte*

FIG. 3.5 – *Cassette de raccordement*

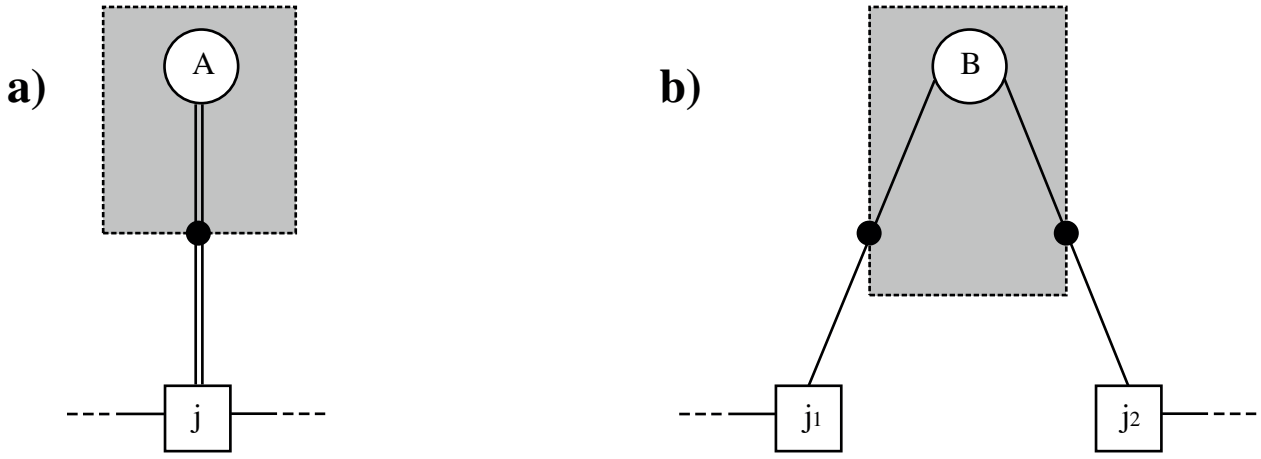


FIG. 3.6 – *Illustration a) de la simple adduction, b) de la double adduction.*

3.1.2.4 Sites clients

Comme nous l'avons vu dans la section précédente, la fiabilité des réseaux SDH tient principalement à sa topologie en anneau. Cependant, au niveau des sites clients, deux cas coexistent qui remettent en cause la fiabilité. Pour la plupart des clients, un seul câble pénètre dans le BFP par défaut, on parle de *simple adduction*. Une fibre est alors allouée à l'aller et une autre au retour. La connexion n'est donc pas sécurisée puisque la structure en anneau n'est pas vérifiée sur la totalité du parcours. Une coupure sur ce câble pourra entraîner la déconnexion du client. Pour pallier à ce risque, certains clients demandent une *double adduction*. Dans ce cas précis, deux câbles différents pénètrent l'immeuble par des BFP différents : ces câbles sont enfouis à partir de façades de l'immeuble donnant sur deux rues différentes, ainsi, la structure en anneau est vérifiée y compris au niveau du site client. La figure 3.6 illustre les deux cas de figure. Les bâtiments client sont symbolisés par des rectangles gris, BFP sont représentés par des points noirs, les boîtes de jonction par des carrés, et les équipement clients par des cercles.

3.2 Modélisation

Dans ce document, nous nous intéressons uniquement à la couche physique du réseau. Nous ne prenons en compte que les besoins en service des clients, ce qui peut être vu comme une demande en *bande passante*. Nous cherchons à déterminer quels anneaux de fibres optiques permettent de desservir tous les clients à moindre coût tout en respectant

les différentes capacités : celles des anneaux et celles des câbles. Nous montrons ici comment ce problème se réduit au k-γ-RSP en indiquant comment à partir d'un réseau SDH nous pouvons construire une instance du k-γ-RSP.

Le graphe support $G = (V,E,A)$ peut être construit de la manière suivante :

- Un sommet correspond à chaque dépôt, client, BFP ou boîte de jonction.
- Pour chaque câble entre deux connecteurs ou entre un dépôt et un connecteur il existe une arête e dont le coût de routage l_e est égale à la longueur du câble.
- Si un client i a demandé une double adduction, un sommet connecteur j est ajouté pour chaque BFP relié à ce client. Nous ajoutons également une arête $e = \{i,j\}$ dont le coût de routage l_e est égale à la longueur du câble, ainsi qu'un arc (i,i) de coût nul.
- À l'inverse, si un client i a requis une simple adduction, un arc (i,j) est ajouté pour tout chemin possible entre le BFP du client et un sommet connecteur j. Nous fixons alors le coût d'affection c_{ij} à deux fois la longueur du chemin correspondant.

De plus une capacité γ_e est considérée pour chaque arête $e \in E$ comme étant le nombre de fibres optiques disponibles dans le câble correspondant. La capacité totale maximale Q d'un anneau-étoile est égale à celle en Mbits/s de la trame considérée. Enfin, la demande d_i de chaque client $i \in U$ correspond à sa demande en bande passante.

Par exemple, la figure 3.7-a représente un réseau SDH simple mais réaliste avec 1 dépôt indiqué par la lettre d et 6 clients représentés par des numéros. Les clients sont regroupés dans des bâtiments symbolisés par des rectangles gris et les 7 points noirs correspondent aux BFP qui servent d'interface entre l'intérieur et l'extérieur du bâtiment. Le reste du réseau est constitué de câbles reliés entre eux par 7 niches (sommet Steiner) représentées par des triangles.

Par exemple, le client 1 doit être relié en double adduction en choisissant 2 niches parmi les 3 disponibles et en passant par les 2 BFP du bâtiment, tandis que le client 2 sera relié en simple adduction grâce au seul BFP disponible en choisissant l'une des niches accessibles.

La figure 3.7-b montre une solution réalisable pour ce réseau avec trois anneaux différenciés par l'utilisation de tracés différents : le premier dessert les clients 4 et 5 ; le second dessert les clients 1 et 3 ; le dernier desservant les clients 2 et 6. La figure 3.8-a (*resp.* 3.8-b) montre le graphe (*resp.* la solution) construit à partir du réseau de la figure 3.7. Les clients sont représentés par des cercles, les triangles blancs correspondent aux boîtes des jonction. Enfin, les triangles noires correspondent aux sommets connecteurs

représentant les BFP reliés aux clients en double adduction. On peut remarquer que le client 2 est relié par deux arcs à deux sommets Steiner dans l'instance 3.8-a.

Par construction, chaque anneau-étoile R sur le graphe G ainsi construit correspond à un anneau de fibre optique réalisable sur le réseau optique. La réciproque est également vraie. Comme le coût total de routage et d'affectation de l'anneau-étoile est exactement la longueur de fibre optique utilisée sur le réseau, nous avons donc le résultat suivant :

Théorème 3.2.1 *Le problème de conception de réseau SDH de seconde génération se réduit au k-γ-RSP.*

3.3 Conclusion

Dans ce chapitre, nous avons présenté les réseaux SDH et leur topologie en anneaux-étoiles. Nous avons également montré comment le problème de conception d'un tel réseau pouvait se ramener au problème k-γ-RSP. Si de nombreux cas particuliers ont déjà été étudiés, ce problème n'a, à notre connaissance, jamais été traité.

Afin de développer un outil de résolution exacte pour le problème de conception de réseaux en anneau-étoile, nous nous sommes intéressés au problème du k-γ-RSP. Nous présentons dans le chapitre 4 plusieurs formulations en nombres entiers pour le cas mono-dépôt de ce problème. Nous menons dans le chapitre 5 une étude polyédrale du dominant de l'une de ces formulations, ce qui nous permet de proposer un algorithme de Branch-and-Cut pour résoudre le 1-γ-RSP. Nous nous intéressons par la suite au k-γ-RSP que nous modélisons par une formulation à nombre exponentiel de variables dans le chapitre 6. Nous détaillons également plusieurs familles d'inégalités valides, en particulier issues du polytope du Stable, permettant de renforcer cette formulation. Cela nous permet dans le chapitre 7 de proposer un algorithme de Branch-and-Cut-and-Price dans lequel, pour la première fois, nous prenons en compte de manière exhaustive des contraintes issues du polytope du Stable. Pour cela nous utilisons l'algorithme du Branch-and-Cut du chapitre 5 pour résoudre le problème auxiliaire.

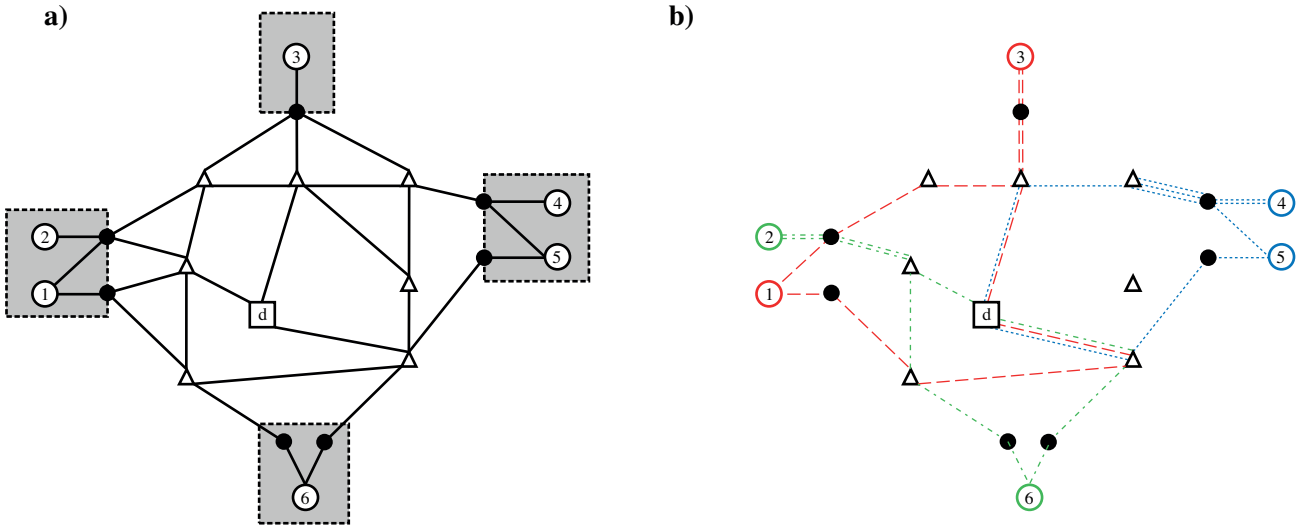


FIG. 3.7 – *a) Un exemple de réseau SDH - b) Le cheminement des fibres*

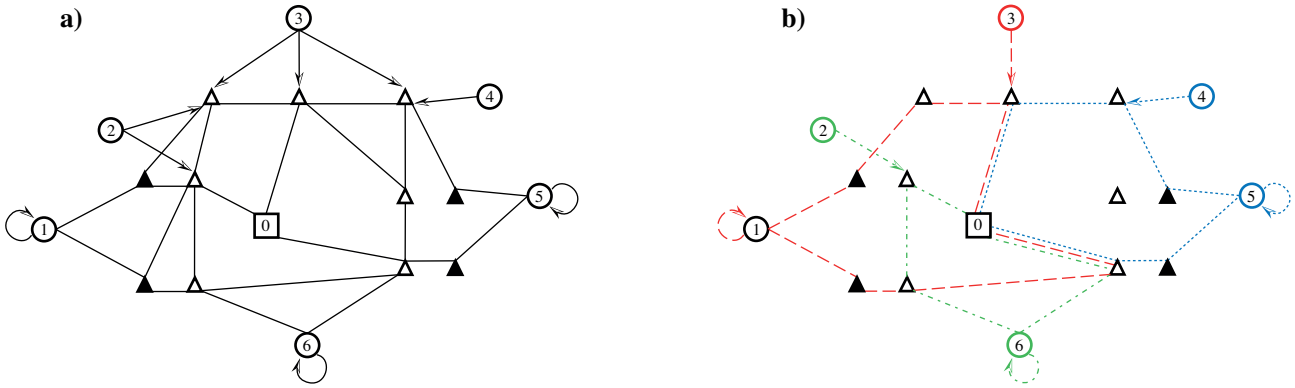


FIG. 3.8 – *a) Une instance du k-γ-RSP - b) Une solution*

Chapitre 4

Formulations linéaires mixtes pour le 1-γ-RSP

Dans ce chapitre, nous discutons de la caractérisation d'une solution au problème 1-γ-RSP. Nous proposons ensuite trois formulations linéaires en nombres entiers pour le 1-γ-RSP et comparons la valeur de leur relaxation linéaire. Pour la formulation naturelle, nous proposons également plusieurs familles d'inégalités permettant de renforcer ces formulations.

4.1 Caractérisation d'une solution

Pour nous permettre de proposer une formulation en nombres entiers pour le 1-γ-RSP, nous devons déterminer un ensemble de variables dont la valeur permet de retrouver une solution optimale. Dans les articles [8] et [75], les auteurs ont utilisé des formulations à 2 indices, c'est à dire des formulations basées sur des variables entières x_e, $e \in E$ représentant le nombre d'anneaux étoiles utilisant l'arête e. Dans ces travaux, toutes les arêtes ont une capacité unitaire et les variables associées sont donc binaires. Cela implique que l'ensemble $\{e \in E : x_e = 1\}$ est directement un ensemble d'anneaux-étoiles disjoints. Nous montrons dans cette section que pour le cas non-disjoint qui nous intéresse, une solution ne peut être décrite par un tel ensemble de variables.

Considérons pour cela le Problème de Décomposition en Anneaux-Étoiles, noté RSDP, basé sur une instance (G,d,γ,c,l,Q,m) du 1-γ-*RSP* et défini comme suit.

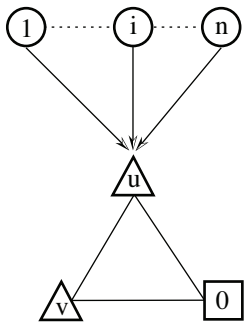
(RSDP): Soit un graphe G, une capacité Q et un vecteur $(x,y) \in \mathbb{N}^{|E|} \times \{0,1\}^{|A|}$, existe-t-il une solution $R = (R_1,\ldots,R_m)$ telle que :
- $R_i = (E(R_i),L(R_i))$, $i = 1,\ldots,m$,
- $|\{R_i,\ i = 1,\ldots,m : e \in E(R_i)\}| = x_e$, $\forall e \in E$,
- $\cup_{i=1,\ldots,m} L(R_i) = \{(i,j) \in A : y_{ij} = 1\}$?

Une telle instance sera notée (G,Q,x,y).

Théorème 4.1.1 *RSDP est NP-complet.*

Preuve. Nous allons montrer que le problème de Bin Packing (BPP) est réductible au RSDP. Une instance du BPP est donnée par un ensemble fini d'éléments $U = \{1,\ldots,n\}$, une taille $a_i \in \mathbb{N}$ pour chaque élément $i \in U$, une capacité entière Q et un entier positif m. Le BPP consiste à répondre à la question suivante : existe-t-il une partition de U en m ensembles $U_1,\ldots,U_m$ telle que la somme des tailles des éléments dans chaque ensemble U_i est inférieure ou égale à Q : $\sum_{k \in U_i} \leq Q$, $\forall i \in 1,\ldots,m$?

Étant donnée une instance $I = \{U,a,m,Q\}$ du *BPP*, il est possible de construire une instance $I' = (G,Q,x,y)$ du RSDP de la manière suivante : considérons U comme un ensemble de sommets clients et u, v, 0, trois sommets supplémentaires. On construit alors le graphe $G = (V,E,A)$ avec $V = \{0,u,v\} \cup U$ et $E = \{\{0,u\},\{u,v\},\{v,0\}\}$. Soient $d_i = a_i$ la demande du client $i \in U$ et u le seul connecteur de i : $L_i = \{(i,u)\}$, pour chaque client $i \in U$. Enfin, nous posons $x_e = m$, $\forall e \in E$ et $y_{iu} = 1$, $\forall i \in U$. La figure 4.1 illustre l'instance I' du *RSDP* obtenue à partir de l'instance I du BPP.



Par construction, l'instance I de BPP aura une solution si et seulement s'il existe une

solution pour l'instance I' de RSDP. Puisque BPP est NP-complet [50], nous pouvons en déduire que RSDP est également NP-complet. □

Le résultat précédent montre que, à moins que P = NP, il sera impossible de déduire en temps polynomial une solution du 1-γ-RSP d'après un vecteur $(x,y) \in \mathbb{N}^{|E|} \times \{0,1\}^{|A|}$. Nous pouvons en déduire le corolaire suivant :

Corollaire 4.1.2 *Il n'existe pas de formulation à 2 indices pour le* 1-γ-RSP.

Nous proposons ainsi dans les prochaines sections plusieurs formulations à 3 indices pour le 1-γ-RSP.

4.2 Formulations compactes

Pour ces deux premières formulations, considérons un graphe orienté $G' = (V,B,A)$ où B contient deux arcs (i,j) et (j,i) pour chaque arête $\{i,j\} \in E$. De plus, posons $l_{ij} = l_{ji} = l(e)$, $\forall e = \{i,j\} \in E$. À chaque arc $(i,j) \in B$ nous associons m variables binaires x_{ij}^t égale à 1 si l'arc appartient à l'anneau-étoile t, 0 sinon. À chaque affectation $(i,j) \in A$, associons m variables binaires y_{ij}^t égale à 1 si le client i est affecté au noeud j appartenant à l'anneau-étoile t, 0 sinon.

4.2.1 Formulation MTZ

Cette première formulation, notée (MTZ) est basée sur les contraintes MTZ, introduites par Miller, Tucker and Zemlin dans [84].

$$Min \sum_{t=1}^{m} \sum_{(i,j)\in B} l_{ij}x_{ij}^t + \sum_{t=1}^{m} \sum_{(i,j)\in A} c_{ij}y_{ij}^t$$

$$\sum_{t=1}^{m} \sum_{(i,j)\in A} y_{ij}^t = 1 \qquad \forall i \in U \tag{4.1}$$

$$x^t(\delta^+(i)) \geq y_{ki}^t \qquad \forall (k,i) \in A,\ \forall t = 1,\ldots,m \tag{4.2}$$

$$x^t(\delta^+(i)) = x^t(\delta^-(i)) \qquad \forall i \in V,\ \forall t = 1,\ldots,m \tag{4.3}$$

$$x^t(\delta^+(i)) \leq 1 \qquad \forall i \in V,\ \forall t = 1,\ldots,m \tag{4.4}$$

$$x_{ij}^t + x_{ji}^t \leq 1 \qquad \forall \{i,j\} \in E,\ \forall t = 1,\ldots,m \tag{4.5}$$

$$\sum_{t=1}^{m} (x_{ij}^t + x_{ji}^t) \leq \gamma_e \qquad \forall e = \{i,j\} \in E \tag{4.6}$$

$$\sum_{(i,j)\in A} d_i y_{ij}^t \leq Q \qquad \forall t = 1,\ldots,m \tag{4.7}$$

$$u_i^t - u_j^t + 1 \leq n(1 - x_{ij}^t) \qquad \forall (i,j) \in B,\ j \neq 0,\ \forall t = 1,\ldots,m \tag{4.8}$$

$$0 \leq u_i^t \leq n - 1 \qquad \forall i \in V,\ \forall t = 1,\ldots,m \tag{4.9}$$

$$x_{ij}^t \in \{0,1\} \qquad \forall (i,j) \in B,\ \forall t = 1,\ldots,m$$

$$y_{ij}^t \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall t = 1,\ldots,m$$

Les contraintes (4.1) imposent que chaque client soit desservi. Les contraintes (4.2), (4.3) et (4.4) induisent la recherche d'un flot unitaire passant par tous les sommets connecteurs actifs de l'anneau-étoile t. Associées aux contraintes MTZ (4.8), elles impliquent une topologie en anneau. En effet, les variables u_i^t induisent une numérotation sur les sommets de l'anneau, ce qui assure que chaque cycle passe par le dépôt 0 tout en interdisant les sous-tours. Les contraintes (4.6) assurent que la capacité de chaque arête est bien respectée pendant que les contraintes (4.7) limitent la capacité de chaque anneau-étoile à Q.

Comme présenté dans [34], les contraintes (4.8) peuvent être renforcées et remplacées par les contraintes suivantes :

$$u_i^t - u_j^t + 1 \leq n(1 - x_{ij}^t) + (2 - n)x_{ji}^t \tag{4.10}$$

Nous noterons ($MTZL^*$) la relaxation linéaire de la formulation définie par les contraintes (4.1)-(4.9) avec (4.8) remplacée par (4.10).

4.2.2 Formulation flot à une commodité

Pour le TSP, une autre formulation flot, connue sous le nom de formulation flot à une commodité, a été présentée par Gavish and Graves dans [52] et par la suite explorée par Gouveia dans [55]. Cette formulation peut facilement être adaptée à notre problème de la manière suivante :

on ajoute de nouvelles variables f correspondant à un flot tel que $f_{ij}^t = 0$ si $x_{ij} = 0$, et tel que f_{ij} représente la quantité de demande desservie par l'anneau-étoile t après la visite du noeud i. La formulation, notée (F1) est obtenue en remplaçant les contraintes (4.8) et (4.9) par les contraintes suivantes dans la formulation (MTZ).

$$f^t(\delta^+(i)) = f^t(\delta^-(i)) + \sum_{k \in U} d_k y_{ki}^t \quad \forall i \in V \setminus \{0\}, \forall t = 1, \ldots, m \tag{4.11}$$

$$0 \leq f_{ij}^t \leq Q x_{ij}^t \qquad \forall (i,j) \in B, \forall t = 1, \ldots, m \tag{4.12}$$

Nous noterons $(F1*)$ la relaxation linéaire de cette formulation.

4.3 Formulation naturelle

Le 1-γ-RSP peut aussi être formulé comme un programme linéaire en nombres entiers où les variables, dites naturelles, sont directement en correspondance avec les arêtes du graphe G : à chaque arête $e \in E$ nous associons m variables binaires x_e^t égale à 1 si l'arête e appartient à l'anneau-étoile t, 0 sinon. À chaque affectation $(i,j) \in A$ nous associons m variables binaires y_{ij}^t égale à 1 si le client i est affecté au noeud j appartenant à l'anneau-étoile t, 0 sinon.

Nous considérons la formulation (NF) suivante :

$$Min \sum_{t=1}^{m} \sum_{e \in E} l_e x_e^t + \sum_{t=1}^{m} \sum_{(i,j) \in A} c_{ij} y_{ij}^t$$

$$\sum_{t=1}^{m} \sum_{(i,j) \in A} y_{ij}^t = 1 \qquad \forall i \in U \tag{4.13}$$

$$\sum_{(i,j) \in A} d_i y_{ij}^t \leq Q \qquad \forall t = 1, \ldots, m \tag{4.14}$$

$$\sum_{e \in \delta(i)} x_e^t \leq 2 \qquad \forall i \in V, \forall t = 1, \ldots, m \tag{4.15}$$

$$\sum_{t=1}^{m} x_e^t \leq \gamma_e \qquad \forall e \in E \tag{4.16}$$

$$\sum_{e \in \delta(S)} x_e^t \geq 2 \sum_{(i,j) \in A, j \in S} y_{ij}^t \qquad \forall i \in U, \forall t = 1, \ldots, m, \quad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{4.17}$$

$$x_e^t \in \{0,1\} \qquad \forall e \in E, \forall t = 1, \ldots, m$$

$$y_{ij}^t \in \{0,1\} \qquad \forall (i,j) \in A, \forall t = 1, \ldots, m$$

Les contraintes (4.13) imposent que chaque client soit affecté. Les contraintes (4.14) imposent le respect de la capacité de chaque anneau-étoile, pendant que les contraintes (4.16) assurent le respect des capacités des arêtes. Les contraintes (4.17) sont les contraintes de 2-connexité, elles imposent que pour un anneau-étoile donné, il existe 2 chemins arête-disjoints entre le dépôt et tout noeud connecteur actif. Associées aux contraintes de degré (4.15), les contraintes (4.17) garantissent la structure en anneau de la solution.

La formulation (NF) est inspirée de la formulation proposée par Labbe et al. pour le problème de l'anneau-étoile [75]. Un algorithme polynomial de séparation pour les contraintes (4.17) sera donné en section 5.2.

4.4 Comparaison des formulations

La faible qualité de la relaxation linéaire des formulations basées sur les contraintes MTZ est avérée dans de nombreux cas (TSP et CVRP par exemple). Nous avons ici un résultat similaire :

Proposition 4.4.1 Si $n > 1$, la borne inférieure fournie par $(MTZL*)$ est égale à la borne inférieure $(R1*)$ obtenue grâce aux contraintes (4.13), (4.14), (4.15), aux contraintes triviales et renforcées par les contraintes suivantes :

$$\sum_{e \in \delta(i)} x_e^t \geq 2y_{ki}^t \quad \forall (k,i) \in A, \forall t = 1, \ldots, m \tag{4.18}$$

Preuve. Tout d'abord, étant donnée (x,y,u) une solution de $(MTZL*)$, nous pouvons construire une solution $(\bar{x},\bar{y})$ de $(R1*)$ ayant la même valeur en posant $\bar{x}_e^t = x_{ij}^t + x_{ji}^t$ pour toute arête $e = \{i,j\} \in E$ et $\bar{y}_{ij}^t = y_{ij}^t$ pour tout arc $(i,j) \in A$. Ensuite, étant donnée $(\bar{x},\bar{y})$ une solution de $(R1*)$, nous pouvons construire une solution (x,y,u) de $(MTZL*)$ ayant la même valeur en posant $x_{ij}^t = x_{ji}^t = \frac{\bar{x}_e^t}{2}$ pour toute arête $e = \{i,j\} \in E$, et $y_{ij}^t = \bar{y}_{ij}^t$ pour tout arc $(i,j) \in A$ et en fixant u comme le vecteur nul. □

La proposition précédente implique que la relaxation linéaire de la formulation (NF) est toujours plus intéressante que la relaxation linéaire de la formulation $(MTZL)$. Cependant les valeurs de relaxation des formulations $(F1)$ et (NF) sont incomparables. Une comparaison expérimentale, qui sera détaillée en section 5.3, a montré que si la relaxation linéaire de $(F1)$ est souvent meilleure que celle de (NF) ce n'est pas toujours le cas. Pour mener une étude polyédrale, la formulation (NF) semble être un meilleur cadre puisque cette formulation peut être vue comme une extension de plusieurs travaux polyédraux similaires [5, 28, 38, 69, 75]. De plus, nous pouvons faire la remarque suivante :

Remarque 4.4.2 *Soit $ax + by \leq c$ une inégalité valide pour (NF), nous pouvons construire une inégalité valide $ax + by \leq c$ pour $(F1)$ en posant $x_{ij}^t + x_{ji}^t = x_e^t$.*

Cette remarque nous permettra donc d'adapter tous les résultats théoriques obtenus dans les sections suivantes pour renforcer la formulation $(F1)$.

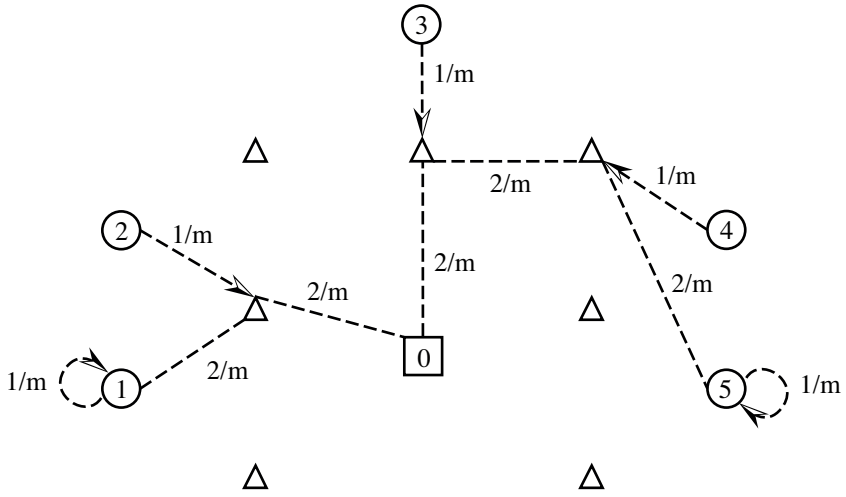


FIG. 4.1 – *Un point fractionnaire typique*

4.5 Inégalités valides

4.5.1 Contraintes de connexité simple

La figure 4.1 décrit un point fractionnaire typique obtenu à partir de la relaxation linéaire de (4.13)-(4.17). Cette solution est composée de m sous-graphes identiques G^t où $G^t = (V^t,E^t,A^t)$, $t = 1,\ldots,m$, est le graphe support des vecteurs x^t et y^t. On remarque que les sous-graphes $(V(G^t),E(G^t))$, que l'on désire être des anneaux-étoiles, ne sont ici que des "chemins-étoiles". Ces points fractionnaires sont obtenus lorsque les inégalités (4.17) sont satisfaites à l'égalité.

La structure de cette solution fractionnaire donne l'intuition de la remarque suivante : étant donné un sous-ensemble non vide $S \in V \setminus \{0\}$, si une arête $e' \in \delta(S)$ est retirée du graphe, il doit toujours exister au moins un chemin entre le dépôt et tout noeud connecteur. Cette remarque montre que les inégalités (4.19) sont valides pour le 1-γ-*RSP* :

$$\sum_{e\in\delta(S)\setminus\{e'\}} x_e^t \geq \sum_{(i,j)\in A, j\in S} y_{ij}^t \qquad \forall i \in U,\ \forall t = 1,\ldots,m, \quad \forall S \subseteq V \setminus \{0\},\ S \neq \emptyset,\ \forall e' \in \delta(S) \tag{4.19}$$

L'ajout de ces inégalités à la formulation permet de couper les points fractionnaires mis en évidence précédemment. Pour l'exemple de la figure 4.1, en prenant $S = \{1\}$ et e' étant une arête de $\delta(1)$ avec $x_{e'}^t = 2/m$ pour un indice t arbitrairement choisi, nous

pouvons produire une inégalité (4.19) violée par ce point fractionnaire.

4.5.2 Contraintes de capacité fractionnaire

Dans la formulation classique du CVRP donnée dans [28], les inégalités principales, connues sous le nom de contraintes de capacité, imposent les besoins en nombre de tournées pour les sous-ensembles de clients. Parmi ces inégalités, les seules qui soient séparables en temps polynomial sont les contraintes de capacité fractionnaire. Dans notre cas, la demande desservie par un anneau-étoile dépend de la valeur des variables y. Les contraintes de capacité doivent donc être adaptées de la manière suivante :

$$\sum_{e\in\delta(S)} x_e^t \geq \frac{2}{Q} \sum_{(i,j)\in A, j\in S} d_i y_{ij}^t, \qquad \forall t = 1,\ldots,m, \quad \forall S \subseteq V\setminus\{0\},\ S\neq\emptyset \tag{4.20}$$

Ces contraintes imposent qu'un nombre suffisant d'anneaux-étoiles entrent dans chaque sous-ensemble de noeud, et sont donc valides pour le 1-γ-RSP.

Remarquons que les contraintes de capacité du CVRP peuvent être obtenues en sommant certaines des inégalités précédentes de la manière suivante : Soit un sous-ensemble de clients $\mathcal{U} \subset U$, nous choisissons $S \subsetneq V$ tel que $\{j \in V : \exists (i,j) \in L_i,\ i \in \mathcal{U}\} \subset S$. En sommant les inégalités (4.20) définies sur S pour les indices $t = 1,\ldots,m$ nous obtenons

$$\sum_{t=1}^{m} \sum_{e\in\delta(S)} x_e \geq \frac{2d(\mathcal{U})}{Q}.$$

En posant $x_e = \sum_{t=1,\ldots,m} x_e^t$, $\forall e \in E$, cette ingalité est exactement la contrainte de capacité fractionnaire du CVRP pour le sous-ensemble S.

4.5.3 Contraintes de capacité arrondie

Dans la formulation du CVRP [28], le membre droit de la contrainte de capacité peut être arrondi à l'entier supérieur, donnant ainsi une contrainte plus forte. Dans le cas du 1-γ-RSP, une telle opération d'arrondi ne peut être faite directement. En

revanche, nous pouvons utiliser l'idée d'arrondi dans une contrainte valide. Notons que pour un sous-ensemble de clients $\mathcal{U} \subset U$, $\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil$ donne une borne inférieure du nombre d'anneaux-étoiles requis pour satisfaire la demande de ces clients. De plus, si l'on suppose qu'une quantité Q de la demande $d(\mathcal{U})$ est satisfaite grâce à l'anneau-étoile t', alors le nombre minimal d'anneaux-étoiles nécessaires pour desservir la demande résiduelle sera $\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil - 1$. Cette idée peut être généralisée de la façon suivante : en omettant un sous-ensemble d'anneaux-étoiles $\mathcal{T} \subsetneq \{1,\ldots,m\}$, le nombre minimal d'anneaux-étoiles nécessaires pour satisfaire la demande restante est alors $\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil - |\mathcal{T}|$. Conséquemment, les contraintes de capacité arrondie suivantes sont valides pour le 1-γ-RSP.

$$\sum_{t=1,\ t\notin\mathcal{T}}^{m} \sum_{e\in\delta(S)} x_e^t \geq 2\left(\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil - |\mathcal{T}|\right) \qquad \forall \mathcal{U} \subsetneq U, \mathcal{U} \neq \emptyset, \quad \forall \mathcal{T} \subsetneq \{1,\ldots,m\}, \quad \{j \in V : \exists (i,j) \in L_i,\ i \in \mathcal{U}\} \subset S \tag{4.21}$$

Notons qu'ici aussi, en choisissant S comme précédemment et en posant $\mathcal{T} = \emptyset$, nous obtenons exactement les contraintes de capacité arrondie du CVRP.

4.6 Conclusion

Dans ce chapitre, nous avons présenté plusieurs formulations linéaires en nombres entiers pour le 1-γ-RSP. Nous avons comparé théoriquement et expérimentalement la qualité des bornes inférieures données par la valeur de leur relaxation linéaire. Pour la formulation dite naturelle, nous avons également introduit plusieurs familles d'inégalités, inspirée de la littérature du TSP et du CVRP.

Dans le chapitre suivant, nous étudions ces inégalités en menant une étude polyédrale sur le dominant de la formulation naturelle défini comme la relaxation de l'obligation de desservir tous les clients.

Chapitre 5

Étude polyédrale et algorithme de Branch-and-Cut pour le 1-γ-RSP

5.1 Étude polyédrale d'un dominant de la formulation naturelle

Dans cette section, nous étudions le polytope du 1-γ-RSP que l'on définit comme l'enveloppe convexe des solutions de la formulation (NF). Malheureusement, le problème d'existence d'une solution réalisable à une instance du 1-γ-RSP est NP-complet, même si le graphe support G est complet. Conséquemment, déterminer la dimension du polytope associé est difficile. Cependant, nous pouvons étudier le dominant $\mathcal{P}$ défini comme la relaxation de l'obligation de desservir tous les clients. Puisque le polytope du 1-γ-RSP est une face de $\mathcal{P}$, étudier $\mathcal{P}$ nous donnera un bon aperçu des propriétés faciales de nos inégalités.

Nous définissons donc $\mathcal{P}$ comme l'enveloppe convexe des solutions vérifiant la formulation (NF) dans laquelle les contraintes (4.13) sont remplacées par les inégalités suivantes :

$$\sum_{t=1}^{m} \sum_{(i,j)\in A} y_{ij}^t \leq 1, \quad \forall i \in U. \tag{5.1}$$

Dans ce qui suit, nous supposerons que le graphe G est complet, qu'il contient au moins 5 sommets, dont au moins un client.

Par souci de concision, nous introduisons de nouvelles notations pour décrire succinctement une solution entière de $\mathcal{P}$: une solution du 1-γ-RSP σ sera donnée par ses composantes non nulles selon la notation $\sigma = \left(\{x_e^t : x_e^t = 1\}; \{y_{ij}^t : y_{ij}^t = 1\}\right)$.

Étant donné un arc (i,j), un entier $t \in \{1,\ldots,m\}$ et un noeud arbitraire k, nous considérerons fréquemment une solution particulière χ_{ij}^t

$$\chi_{ij}^t = (x_{0j}^t, x_{jk}^t, x_{k0}^t; y_{ij}^t).$$

χ_{ij}^t est donc le vecteur d'incidence d'un anneau-étoile passant par le dépôt, le noeud k et un connecteur actif j sur lequel est affecté le client i. Une telle solution existera toujours sous nos hypothèses.

5.1.1 Propriétés basiques

Dans cette section, nous présentons les propriétés basiques pour la description faciale de notre polytope.

Théorème 5.1.1 *$\mathcal{P}$ est de pleine dimension.*

Preuve. Pour montrer que $dim(\mathcal{P}) = m|E|+m|A|$ nous devons exhiber $m|E|+m|A|+1$ points entiers de $\mathcal{P}$ dont les vecteurs d'incidence sont affinement indépendants.

Tout d'abord, nous pouvons remarquer que les points $(x_e^t; \emptyset)$, avec $t = 1,\ldots,m$ et $e \in E$, sont contenus dans $\mathcal{P}$. De plus, étant donnés un arc (i,j) et un indice $t \in \{1,\ldots,m\}$, χ_{ij}^t est aussi clairement un point de $\mathcal{P}$. Nous obtenons donc $m|E|+m|A|$ points entiers de $\mathcal{P}$ dont les vecteurs d'incidence forment une matrice M. Si nous plaçons les colonnes correspondant aux variables x sur la gauche et celles correspondant aux variables y sur la droite, la matrice M peut donc être écrite de la manière suivante :

$$M = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ & & & 1 & \cdots & 0 \\ & M' & & 0 & \ddots & 0 \\ & & & 0 & \cdots & 1 \end{bmatrix}$$

Puisque M est une matrice non singulière et que le vecteur nul $(\emptyset;\emptyset)$ est aussi un point de $\mathcal{P}$, on peut en déduire que $\mathcal{P}$ est de pleine dimension. □

Grâce à des démonstrations similaires, nous pouvons obtenir le même résultat pour les inégalités triviales de positivité :

Corollaire 5.1.2 *Soit $t' \in \{1,\ldots,m\}$. Étant donnée une arête $e' \in E$, l'inégalité triviale $x^{t'}_{e'} \geq 0$ définit une facette de $\mathcal{P}$.*

Preuve. Notons que les points $(\emptyset;\emptyset)$ et $(x^t_e;\emptyset)$ pour tout couple $(e,t) \in E \times \{1,\ldots,m\}$, $(e,t) \neq (e',t')$, sont contenus dans $\mathcal{P}$. De plus, étant donné un arc (i,j) et un indice $t \in \{1,\ldots,m\}$, il est possible, sous nos hypothèses, de construire une solution χ^t_{ij} qui n'utilise pas l'arête e' et qui est aussi clairement un point de $\mathcal{P}$. Nous obtenons donc $m|E|+m|A|$ points entiers de $\mathcal{P}$ dont les vecteurs d'incidence sont affinement indépendants. De plus, ces points satisfont l'inégalités $x^t_{e'} \geq 0$ à l'égalité ce qui nous permet de conclure. □

Corollaire 5.1.3 *Soit $t \in \{1,\ldots,m\}$. Étant donné un arc $(i',j') \in A$, l'inégalité triviale $y^t_{i'j'} \geq 0$ définit une facette de $\mathcal{P}$.*

Preuve. En considérant toutes les solutions de la preuve du théorème 5.1.1 à l'exception de la solution $\chi^t_{i'j'}$, nous obtenons $m|E| + m|A|$ points entiers de $\mathcal{P}$ dont les vecteurs d'incidence sont affinement indépendants. De plus, ces points satisfont l'inégalités $y^t_{i'j'} \geq 0$ à l'égalité ce qui nous permet de conclure. □

Nous pouvons maintenant montrer que les contraintes de degré définissent des facettes de $\mathcal{P}$.

Théorème 5.1.4 *Les contraintes de degré (4.15) définissent des facettes de $\mathcal{P}$.*

Preuve. Soient $i_0 \in V$ et $t_0 \in \{1,\ldots,m\}$. Notons $ax + by \leq \alpha$ la contrainte de degré $\sum_{e \in \delta(i_0)} x^{t_0}_e \leq 2$ et supposons qu'il existe une inégalité $a'x+b'y \leq \alpha'$ définissant une facette de $\mathcal{P}$ telle que $\mathcal{F} = \{(x,y) \in \mathcal{P} : ax+by = \alpha\} \subset \mathcal{F}' = \{(x,y) \in \mathcal{P} : a'x+b'y = \alpha'\}$. Nous allons prouver grâce à l'assertion 5.1.1 suivante que la contrainte de degré $ax + by \leq \alpha$ définit une facette $\mathcal{P}$.

Assertion 5.1.1 *Soit $ax + by \leq \alpha$ (resp. $ax + by \geq \alpha$) une inégalité valide de $\mathcal{P}$. Supposons qu'il existe une inégalité $a'x + b'y \leq \alpha'$ (resp. $a'x + b'y \geq \alpha'$) définissant une facette de $\mathcal{P}$ telle que $\mathcal{F} = \{(x,y) \in \mathcal{P} : ax + by = \alpha\} \subset \mathcal{F}' = \{(x,y) \in \mathcal{P} : a'x + b'y = \alpha'\}$. Puisque, d'après le théorème 5.1.1, $\mathcal{P}$ est de pleine dimension, s'il existe $\lambda > 0$ tel que $(a',b') = \lambda(a,b)$ et $\alpha' = \lambda\alpha$, alors $ax + by \leq \alpha$ (resp. $ax + by \geq \alpha$) définit une facette de $\mathcal{P}$.* $\Diamond$

Soient $\{e_1,e_2\} \subset \delta(i_0)$ et $e_3 \in E \setminus \delta(i_0)$. En considérant les solutions $\sigma_1 = (x^{t_0}_{e_1},x^{t_0}_{e_2};\emptyset)$ et $\sigma'_1 = (x^{t_0}_{e_1},x^{t_0}_{e_2},x^{t_0}_{e_3};\emptyset)$, nous utilisons l'assertion 5.1.2 :

Assertion 5.1.2 *Considérons deux solutions σ et σ' telles que $ax^\sigma + by^\sigma = \alpha$ et $ax^{\sigma'} + by^{\sigma'} = \alpha$. Alors, par définition, les deux vecteurs vérifient également $a'x + b'y \leq \alpha'$ à l'égalité, et conséquemment, $a'(x^\sigma - x^{\sigma'}) + b'(y^\sigma - y^{\sigma'}) = 0$.* $\Diamond$

Nous obtenons donc que $a^{t_0}_{e_1} + a^{t_0}_{e_2} = a^{t_0}_{e_1} + a^{t_0}_{e_2} + a^{t_0}_{e_3}$, puis que $a^{t_0}_e = 0$, $\forall e \in E \setminus \delta(i_0)$.

Soient $e_4 \in E$ et $t' \neq t_0$. En considérant les solutions σ_1 et $\sigma''_1 = (x^{t_0}_{e_1},x^{t_0}_{e_2},x^{t'}_{e_4};\emptyset)$, grâce à l'assertion 5.1.2 nous obtenons que $a^{t'}_e = 0$, $\forall e \in E$, $t' \neq t_0$.

Soient $(i,j) \in A$ et $t' \neq t_0$. En considérant les deux solutions σ_1 et $\sigma_2 = \sigma_1 + \chi^{t'}_{ij}$, l'assertion 5.1.2 nous permet de conclure que $b^{t'}_{ij} = 0$, $\forall (i,j) \in A$, $t' \neq t_0$.

Soit $(i',j') \in A$. Remarquons que G étant un graphe complet, soit j' est le sommet i_0, soit $\{i_0,j'\} \in \delta(i)$. Si $j' = i_0$, nous posons $k \in V \setminus \{0,i_0\}$ et considérons les solutions $\sigma_2 = (x^{t_0}_{0i_0},x^{t_0}_{i_0k},x^{t_0}_{k0};y^{t_0}_{i'i_0})$ et $\sigma'_2 = (x^{t_0}_{0i_0},x^{t_0}_{i_0k},x^{t_0}_{k0};\emptyset)$. Si $j' \neq i_0$, nous considérons les solutions $\sigma_2 = (x^{t_0}_{0i_0},x^{t_0}_{i_0j'},x^{t_0}_{j'0};y^{t_0}_{i'j'})$ et $\sigma_2 = (x^{t_0}_{0i_0},x^{t_0}_{i_0j'},x^{t_0}_{j'0};\emptyset)$. Dans les deux cas, l'assertion 5.1.2 permet de conclure que $b^{t_0}_{ij} = 0$, $\forall (i,j) \in A$.

Soit $e_4 \in \delta(i)$. En considérant les solutions σ_1 et $\sigma_3 = (x^{t_0}_{e_1},x^{t_0}_{e_4};\emptyset)$, l'assertion 5.1.2, permet d'obtenir que $a^{t_0}_{e_1} = a^{t_0}_{e_2}$. En posant $\lambda = a^{t_0}_{e_1}$, nous obtenons que $a^{t_0}_e = \lambda$, $\forall e \in \delta(i_0)$.

Enfin, en considérant la solution σ_1, nous obtenons $\alpha = 2\lambda$.

Nous avons donc montré que $(a',b') = \lambda(a,b)$ et $\alpha' = \lambda\alpha$. De plus, la solution $(\emptyset,\emptyset)$ étant valide pour notre polytope, $\lambda \geq 0$. Enfin, comme $a'x + b'y \leq \alpha'$ définit une facette de $\mathcal{P}$, $\lambda > 0$. Par conséquent, grâce à l'assertion 5.1.1, nous pouvons conclure que l'inégalité $ax + by \leq \alpha$ définit une facette de $\mathcal{P}$. $\square$

5.1.2 Contraintes de connexité

Contrairement aux inégalités précédentes, les contraintes de connexité (4.17) et (4.19) ne définissent pas toujours de facettes. Pour nous permettre de donner les conditions nécessaires et suffisantes pour obtenir des facette, nous donnons tout d'abord une définition technique:

- *2-couverture*

Étant donné $S \subset V$, nous notons $U(S)$ l'ensemble de clients ayant un connecteur dans S, c'est à dire :

$$U(S) = \{i \in U : \exists (i,j) \in L_i,\ j \in S\}.$$

Un sous-ensemble $C \subset U(S)$ sera une *2-couverture* si et seulement si $d_i + d_j > Q$, $\forall i,j \in C$, $i \neq j$.
Une 2-couverture sera dite *maximale* pour $U(S)$ si pour chaque client $j \in U(S) \setminus C$, il existe un client $i \in C$ tel que $d_i + d_j \leq Q$. Nous noterons $A(C)$ l'ensemble des arcs allant de C dans S : $A(C) = \{(i,j) \in A : i \in C,\ j \in S\}$. Enfin, nous pouvons faire la remarque suivante :

Remarque 5.1.5 Puisque, par définition, la demande cumulée de tout couple de clients dans une 2-couverture dépasse la capacité d'un anneau-étoile, un seul client peut être affecté à un anneau-étoile t donné, autrement dit

$$\sum_{(i,j) \in A(C)} y_{ij}^t \leq 1.$$

- *Contraintes de 2-connexité généralisées*

Étant donnés un sous-ensemble $S \subset V \setminus \{0\}$ et un indice $t \in \{1, \ldots, m\}$, l'inégalité de 2-connexité (4.17) correspondante peut être renforcée en considérant une 2-couverture $C \subset U(S)$ plutôt qu'un seul client. D'après la remarque 5.1.5 la contrainte de 2-connexité généralisée suivante est valide pour $\mathcal{P}$:

$$\sum_{e \in \delta(S)} x_e^t \geq 2 \sum_{(i,j) \in A(C)} y_{ij}^t \tag{5.2}$$

De plus, nous avons le résultat suivant :

Théorème 5.1.6 *Les contraintes de 2-connexité généralisées (5.2) définissent des facettes de $\mathcal{P}$ si et seulement si i) et ii) sont vérifiées :*

i) C est maximale pour $U(S)$,
ii) si $|\bar{S}| < 3$, alors pour tout arc $(i,j) \in A$, $j \in S$ ou $i \in U \setminus C$.

Preuve. Soient $t_0 \in \{1,\ldots,m\}$, $S \subsetneq V$, $S \neq \emptyset$, et $C \subset U(S)$ une 2-couverture. Si C n'est pas maximale pour $U(S)$, il existe une 2-couverture $C' \subset U(S)$ avec $C \subset C'$, il s'ensuit que la contrainte de 2-connexité généralisée définie sur C' domine l'inégalité définie sur C. Dans ce qui suit, nous supposons donc que i) est vérifiée.

Supposons maintenant que ii) ne soit pas vérifiée, autrement dit que $|\bar{S}| < 3$ et qu'il existe un arc $(i',j') \in A$ tel que $j \in \bar{S}$, $i \in C$. Dans ce cas, toute solution valide telle que $y^{t_0}_{i'j'} = 1$ doit utiliser au moins deux arêtes $e,e' \in \delta(S)$. Il s'ensuit que l'inégalité (5.2) est dominée par l'inégalité suivante :

$$\sum_{e \in \delta(S_0)} x^{t_0}_e \geq 2 \sum_{(i,j) \in A(C)} y^{t_0}_{ij} + 2y^{t_0}_{i'j'}$$

À l'inverse, supposons que i) et ii) sont vérifiées. Notons $ax + by \geq 0$ la contrainte de 2-connexité généralisée $\sum_{e \in \delta(S)} x^{t_0}_e \geq 2 \sum_{(i,j) \in A(C)} y^{t_0}_{ij}$. Supposons que les solutions (x,y) de $\{(x,y) \in P : ax + by = 0\}$ satisfassent aussi $a'x + b'y = \alpha'$, où $a'x + b'y \geq \alpha'$ est une inégalité définissant une facette de $\mathcal{P}$. D'après à l'assertion 5.1.1, nous voulons prouver que $a'x + b'y \geq \alpha'$ est un multiple de $ax + by \geq 0$.

Puisque la solution $\sigma_1 = (\emptyset;\emptyset)$ appartient à $\mathcal{F}$, nous pouvons déduire que $\alpha' = 0$.

Soit $e_1 \in E \setminus \delta(S)$, en considérant les solutions σ_1 et $\sigma'_1 = (x^{t_0}_{e_1};\emptyset)$, grâce à l'assertion 5.1.2, nous obtenons que $a^{t_0}_e = 0$, $\forall e \in E \setminus \delta(S)$. Soient $e_2 \in E$ et $t' \neq t_0$, en considérant les solutions σ_1 et $\sigma''_1 = (x^{t'}_{e_2};\emptyset)$, l'assertion 5.1.2 nous permet de déterminer que $a^{t'}_e = 0$, $\forall e \in E$, $t' \neq t_0$. Soient $(i,j) \in A$ et $t' \neq t_0$, en considérant les solutions σ_1 et $\chi^{t'}_{ij}$, grâce à l'assertion 5.1.2, nous pouvons déduire que $b^{t'}_{ij} = 0$, $\forall (i,j) \in A$, $t' \neq t_0$.

Soient $(i,j) \in A(C)$, $e_3 = \{u,v\} \in \delta(S) \setminus \{0,j\}$, $u \in \bar{S}$, $v \in S$ et $k \in \bar{S} \setminus \{0,u\}$. Un tel sommet k existera toujours puisque $|\bar{S}| \geq 3$.
Si $u \neq 0$ et $v \neq j$, considérons les solutions $\sigma_2 = (x^{t_0}_{0k},x^{t_0}_{kj},x^{t_0}_{j0};y^{t_0}_{ij})$ et $\sigma'_2 = (x^{t_0}_{0u},x^{t_0}_{uv},x^{t_0}_{vj},x^{t_0}_{jk},x^{t_0}_{k0};y^{t_0}_{ij})$.
Si $u \neq 0$ et $v = j$, considérons les solutions σ_2 et $\sigma''_2 = (x^{t_0}_{0u},x^{t_0}_{uj},x^{t_0}_{jk},x^{t_0}_{k0};y^{t_0}_{ij})$.
Si $u = 0$ et $v \neq j$, considérons les solutions σ_2 et $\sigma'''_2 = (x^{t_0}_{0v},x^{t_0}_{vj},x^{t_0}_{jk},x^{t_0}_{k0};y^{t_0}_{ij})$.
Dans tous les cas, par l'assertion 5.1.2, nous obtenons $a^{t_0}_{\{uv\}} = a^{t_0}_{\{j0\}}$. En posant $\lambda = a^{t_0}_{\{j0\}}$, nous obtenons ensuite que $a^{t_0}_e = \lambda$, $\forall e \in \delta(S)$.

Soit $(i,j) \in A$ tel que $j \notin S$. Puisque $|\bar{S}| \geq 3$, il existe un noeud $k \in \bar{S}$, $k \neq j$, et nous pouvons considérer les solutions σ_1 et $\sigma_3 = (x^{t_0}_{0j},x^{t_0}_{jk},x^{t_0}_{k0};y^{t}_{ij})$. L'assertion 5.1.2 nous permet d'en déduire que $b^{t_0}_{ij} = 0$, $\forall (i,j) \in A$ tel que $i \in U$ et $j \notin S$.

Soit $(i,j) \in A$ avec $j \in S$ et $i \notin C$. Puisque C est maximale pour $U(S)$, il existe un arc $(i',j') \in A(C)$ tel que $d_i + d_{i'} \leq Q$. Soit $k \in V \setminus \{0,j,j'\}$.
Si $j' \neq j$, nous considérons les solutions $\sigma_4 = (x^{t_0}_{0j'},x^{t_0}_{j'j},x^{t_0}_{j'k},x^{t_0}_{k0}; y^{t_0}_{i'j'})$
et $\sigma'_4 = (x^{t_0}_{0j'},x^{t_0}_{j'j},x^{t_0}_{j'k},x^{t_0}_{k0}; y^{t}_{i'j'},y^{t}_{ij})$.
Si $j' = j$, nous considérons les solutions $\sigma_5 = (x^{t_0}_{0j},x^{t_0}_{jk},x^{t_0}_{k0}; y^{t_0}_{ij})$ et $\sigma'_5 = (x^{t_0}_{0j},x^{t_0}_{jk},x^{t_0}_{k0}; y^{t_0}_{i'j},y^{t_0}_{ij})$
Dans les deux cas, l'assertion 5.1.2 nous permet de conclure que $b^{t_0}_{ij} = 0$, $\forall i \in U \setminus C$, $j \in S$.

Enfin, soient $(i,j) \in A(C)$, $(i',j') \in A(C)$, $(i,j) \neq (i',j')$.
Si $j' \neq j$, nous considérons les solutions $\sigma_6 = (x^{t_0}_{0j'},x^{t_0}_{j'j},x^{t_0}_{j0}; y^{t_0}_{ij})$ et $\sigma'_6 = (x^{t_0}_{0j'},x^{t_0}_{j'j},x^{t_0}_{j0}; y^{t_0}_{i'j'})$.
Si $j' = j$, puisque $|\bar{S}| \geq 3$, il existe un noeud $k \in \bar{S}$, $k \neq 0$, et nous pouvons considérer les solutions $\sigma_7 = (x^{t_0}_{0j},x^{t_0}_{jk},x^{t_0}_{k0}; y^{t_0}_{ij})$ et $\sigma'_7 = (x^{t_0}_{0j},x^{t_0}_{jk},x^{t_0}_{k0}; y^{t_0}_{i'j'})$.
Dans les deux cas, grâce à l'assertion 5.1.2, nous obtenons que $b^{t_0}_{ij} = b^{t_0}_{i'j'}$, $\forall (i,j),(i',j') \in A(C)$.

En considérant la solution σ_7, nous pouvons aussi déduire que $b^t_{ij} = -2a^t_e$, $e \in \delta(S)$, $\forall (i,j) \in A(C)$, c'est à dire $b^t_{ij} = -2\lambda$, $\forall (i,j) \in A(C)$.

Nous avons donc montré que $(a',b') = \lambda(a,b)$. De plus, soit $e \in \delta(S)$, puisque la solution $(x^{t_0}_e,\emptyset)$ est valide pour notre polytope, nous pouvons déduire que $\lambda \geq 0$. Comme l'inégalité $a'x + b'y \geq \alpha'$ définit une facette de notre polytope, $\lambda > 0$. Donc, d'après l'assertion 5.1.2, l'inégalité $ax + by \geq \alpha$ définit bien une facette de $\mathcal{P}$. □

• *Contrainte de connexité généralisée*

De la même manière que précédemment, en utilisant la remarque 5.1.5, nous pouvons renforcer les contraintes de connexité de telle sorte que les inégalités suivantes soient valides :

$$\sum_{e \in \delta(S) \setminus \{e'\}} x^t_e \geq \sum_{i \in \mathcal{C}} \sum_{(i,j) \in A, j \in S} y^t_{ij} \quad \forall S \subseteq V \setminus \{0\},\ S \neq \emptyset, \qquad (5.3)$$
$$\forall C \subset U(S),\ \forall e' \in \delta(S),\ \forall t = 1,\ldots,m$$

Grâce à une preuve similaire à celle du précédent théorème, nous obtenons une condition nécessaire et suffisante pour que les contraintes (5.3) définissent une facette de notre polytope.

Théorème 5.1.7 *Les contraintes de connexité généralisées (5.3) définissent des facettes de $\mathcal{P}$ si et seulement si i) et ii) sont vérifiées :*
i) C est maximale pour $U(S)$,
ii) si $|\bar{S}| < 3$, alors pour chaque arc $(i,j) \in A$, $j \in S$ ou $i \in U \setminus C$.

Preuve. Soient $t_0 \in \{1,\ldots,m\}$, $S \subsetneq V$, $S \neq \emptyset$, $e_0 = \{u,v\} \in \delta(S)$ et $C \subset U(S)$ une 2-couverture. Si C n'est pas maximale pour $U(S)$, il existe une 2-couverture $C' \subset U(S)$ avec $C \subset C'$, il s'ensuit que la contrainte de connexité généralisée définie sur C' domine l'inégalité définie sur C. Dans ce qui suit, nous supposons donc que i) est vérifiée.

Supposons maintenant que ii) ne soit pas vérifiée, autrement dit que $|\bar{S}| < 3$ et qu'il existe un arc $(i',j') \in A$ tel que $j \in \bar{S}$, $i \in C$. Toute solution valide telle que $y^{t_0}_{i'j'} = 1$ doit utiliser au moins deux arêtes $e,e' \in \delta(S)$. Il s'ensuit que l'inégalité (5.3) est dominée par l'inégalité suivante :

$$\sum_{e \in \delta(S)\setminus\{e_0\}} x^t_e \geq \sum_{(i,j)\in A(C)} y^{t_0}_{ij} + y^{t_0}_{i'j'} \quad \forall S \subseteq V \setminus \{0\},\ S \neq \emptyset.$$

À l'inverse, supposons que i) et ii) sont vérifiées. Notons $ax + by \geq 0$ la contrainte de connexité généralisée $\sum_{e\in\delta(S)\setminus\{e_0\}} x^{t_0}_e \geq \sum_{(i,j)\in A(C)} y^{t_0}_{ij}$. Supposons que les solutions (x,y) de $\{(x,y) \in P : ax + by = 0\}$ satisfassent aussi $a'x + b'y = \alpha'$, où $a'x + b'y \geq \alpha'$ est une inégalité définissant une facette de $\mathcal{P}$. D'après à l'assertion 5.1.1, nous voulons prouver que $a'x + b'y \geq \alpha'$ est un multiple de $ax + by \geq 0$.

Puisque la solution $\sigma_1 = (\emptyset;\emptyset) \in \mathcal{F}$, nous obtenons $\alpha' = 0$.

Soit $e_1 \in E \setminus \delta(S)$, considérons les solutions σ_1 et $\sigma'_1 = (x^{t_0}_{e_1};\emptyset)$, grâce à l'assertion 5.1.2, nous obtenons que $a^{t_0}_e = 0$, $\forall e \in E \setminus \delta(S)$. De la même manière, en considérant les solutions σ_1 et $\sigma''_1 = (x^{t_0}_{e_0};\emptyset)$, nous obtenons $a^{t_0}_{e_0} = 0$.

Soient $e_2 \in E$ et $t' \neq t_0$, en considérant les solutions σ_1 et $\sigma'''_1 = (x^{t'}_{e_2};\emptyset)$, l'assertion 5.1.2 nous permet de déterminer que $a^{t'}_e = 0$, $\forall e \in E$, $t' \neq t_0$.

Soient $(i,j) \in A$ et $t' \neq t_0$, considérons les solutions σ_1 et $\chi^{t'}_{ij}$, grâce à l'assertion 5.1.2, nous pouvons déduire que $b^{t'}_{ij} = 0$, $\forall (i,j) \in A$, $t' \neq t_0$.

Soit $(i,j) \in A$ tel que $j \notin S$. Puisque $|\bar{S}| \geq 3$, il existe un noeud $k \in \bar{S}$, $k \neq j$, et nous pouvons considérer les solutions σ_1 et $\sigma_2 = (x^{t_0}_{0j},x^{t_0}_{jk},x^{t_0}_{k0};y^t_{ij})$. L'assertion 5.1.2 nous permet d'en déduire que $b^{t_0}_{ij} = 0$, $\forall (i,j) \in A$ tel que $i \in U$ et $j \notin S$.

Soient $(i,j) \in A(C)$ et $e_1,e_2 \in \delta(S) \setminus \{e_0\}$, avec $e_1 = \{u_1,v_1\}$, $e_2 = \{u_2,v_2\}$. Considérons les deux solutions σ_3 et σ'_3 qui sont données par la figure 5.1. Sur cette figure, un trait pointillé entre deux sommets u et v indique l'arête $\{u,v\}$ si $u \neq v$ ou le fait que $u = v$. Comme $e_1 \neq e_0$ et $e_2 \neq e_0$, nous pouvons construire ces deux solutions σ_3 et σ'_3 quelles que soient les positions des arêtes e_0, e_1 et e_2. En considérant ces deux solutions, nous pouvons conclure grâce à l'assertion 5.1.2 que $a^{t_0}_{e_1} = a^{t_0}_{e_2}$, $\forall (e_1,e_2) \in \delta(S) \setminus \{e_0\}$. En posant $\lambda = a^{t_0}_{e_1}$, nous obtenons ensuite que $a^{t_0}_e = \lambda$, $\forall e \in \delta(S) \setminus \{e_0\}$.

Soit $(i,j) \in A$ avec $j \in S$ et $i \notin C$. Puisque C est maximal pour $U(S)$, il existe un arc $(i',j') \in A(C)$ tel que $d_i + d_{i'} \leq Q$. Nous pouvons alors considérer les solutions σ_4 et σ'_4 données par la figure 5.1, l'assertion 5.1.2 nous permet alors de conclure que $b^{t_0}_{ij} = 0$, $\forall i \in U \setminus C$, $j \in S$.

Enfin, soient $(i,j) \in A(C)$, $(i',j') \in A(C)$, $(i,j) \neq (i',j')$. Nous pouvons alors considérer les solutions σ_5 et σ'_5 données par figure 5.1. Grâce à l'assertion 5.1.2, nous obtenons que $b^{t_0}_{ij} = b^{t_0}_{i'j'}$, $\forall (i,j),(i',j') \in A(C)$.

En considérant la solution σ_5, nous pouvons aussi déduire que $b^t_{ij} = -a^t_e$, $e \in \delta(S) \setminus \{e_0\}$, $\forall (i,j) \in A(C)$, c'est à dire $b^t_{ij} = -\lambda$, $\forall (i,j) \in A(C)$.

Nous avons donc montré que $(a',b') = \lambda(a,b)$. De plus, soit $e \in \delta(S) \setminus e_0$, puisque la solution $(x^{t_0}_e,\emptyset)$ est valide pour notre polytope, nous pouvons déduire que $\lambda \geq 0$. Comme l'inégalité $a'x + b'y \geq \alpha'$ définit une facette de notre polytope, $\lambda > 0$. Donc, d'après l'assertion 5.1.2, l'inégalité $ax + by \geq \alpha$ définit bien une facette de $\mathcal{P}$. □

5.2 Algorithme de Branch-and-Cut

Dans cette section nous présentons un algorithme de Branch-and-Cut pour le 1-γ-*RSP* basé sur les résultats théoriques précédents.

L'algorithme commence par résoudre la relaxation linéaire de la formulation limitée aux contraintes d'affectations (4.13), aux contraintes de degré (4.15) et aux contraintes triviales. Nous utilisons ensuite un algorithme de séparation exacte pour les contraintes de 2-connexité généralisées (5.2). Généralement, la solution obtenue à la fin de cette phase de coupes n'est pas entière et, par conséquent, il est nécessaire de générer des contraintes additionnelles : des inégalité de connexité généralisée (5.3) et des contraintes de capacité (4.20) ou (4.21). Si la solution est toujours fractionnaire, nous utilisons ensuite un arbre de branchement.

Dans ce qui suit, nous présentons les algorithmes de séparation utilisés pour les différentes familles d'inégalités, ainsi que la règle de branchement choisie et une heuristique primale dédiée à notre problème.

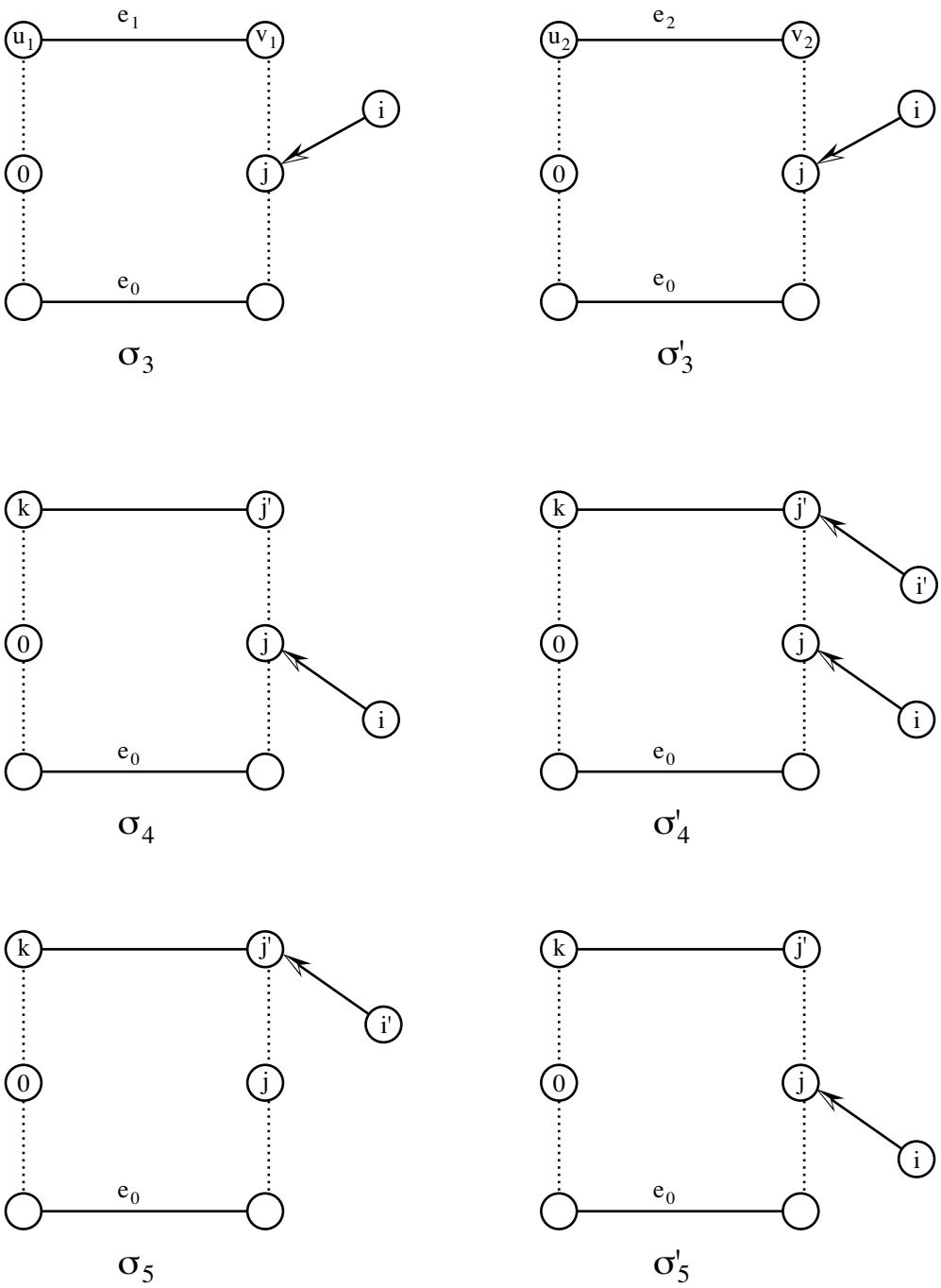


FIG. 5.1 – *Illustration des solutions de la preuve 5.1.2*

5.2.1 Algorithmes exactes et approchés de séparation

Nous présentons maintenant les algorithmes de séparation utilisés dans notre algorithme. Ces séparations sont exactes à l'exception de celle utilisée pour les contraintes de capacité arrondie (4.21). Nous noterons $(\tilde{x},\tilde{y})$ une solution optimale fractionnaire de la relaxation linéaire de (4.13)-(4.15).

5.2.1.1 Contraintes de capacité

Introduisons tout d'abord quelques notations. Étant donné un sous-ensemble de clients C, nous noterons $\tilde{G}_{(C)} = (\tilde{V},\tilde{E})$ le *graphe support* associé à l'ensemble de client C défini comme suit : nous ajoutons à l'ensemble des sommets V un sommet artificiel $n+1$: $\tilde{V} = V \cup \{n+1\}$; le noeud $n+1$ sera relié par une arête à tous les sommets connecteurs des clients de C, nous noterons cet ensemble $E(C)$:

$$E(C) = \{\{j,n+1\} : \exists (i,j) \in A(C)\}\,.$$

L'ensemble d'arêtes $\tilde{E} = E \cup E(C)$ sera constitué des arêtes du graphe d'origine G ainsi que des arêtes de $E(C)$.

- *Contraintes de capacité fractionnaire*

Considérons maintenant la séparation des contraintes de capacité fractionnaire (4.20). L'algorithme de séparation exacte suivant est inspiré de celui utilisé pour les contraintes de capacité fractionnaire du CVRP présenté dans [3]. Étant donnés un indice $t_0 \in \{1,\ldots,m\}$ et une solution partielle $(\tilde{x}^{t_0},\tilde{y}^{t_0})$, nous munissons le graphe support $\tilde{G}_{(U)}$ d'un vecteur de capacité w défini comme suit :

- $\forall e = \{j,n+1\} \in E(C)$, $w_e = \frac{2}{Q} \sum_{(i,j)\in A} \tilde{y}_{ij}^{t_0}$
- $\forall e \in \tilde{E} \setminus E(C)$, $w_e = \tilde{x}_e^{t_0}$

Recherchons ensuite dans ce graphe la coupe de capacité minimale séparant le dépôt du sommet $n+1$. Soit S l'ensemble de noeuds défini par cette coupe et contenant le sommet $n+1$. Soit $\bar{D}$ la somme suivante :

$$\bar{D} = \frac{2}{Q} \sum_{(i,j)\in A} d_i \tilde{y}_{ij}^{t_0}.$$

Notons que pour $(\tilde{x}^{t_0},\tilde{y}^{t_0})$ fixé, $\bar{D}$ est une constante. En soustrayant $\bar{D}$ à la capacité de la coupe nous obtenons :

$$f(S) = \sum_{e\in\delta(S)} \tilde{x}_e^{t_0} + \frac{2}{Q} \sum_{(i,j)\in A, j\in\bar{S}} \tilde{y}_{ij}^{t_0} - \bar{D} = \sum_{e\in\delta(S)} x_e^{t_0} - \frac{2}{Q} \sum_{(i,j)\in A, j\in S} \tilde{y}_{ij}^{t_0}$$

qui est donc la violation maximale des contraintes (4.20). Si $f'(S) < 0$, l'ensemble S définit une contrainte de capacité fractionnaire (4.20) violée par la solution partielle $(\tilde{x}^{t_0},\tilde{y}^{t_0})$; dans le cas contraire, toutes les contraintes de capacité (4.20) sont vérifiées par la solution partielle $(\tilde{x}^{t_0},\tilde{y}^{t_0})$. Notons cependant que tel quel, l'algorithme peut retourner un ensemble S vide. Si cela se produit, nous relançons au plus n fois la même procédure en forçant chaque connecteur actif j (c'est à dire tel que $\sum_{(i,j)\in A} \tilde{y}_{ij}^{t_0} > 0$) à appartenir à l'ensemble S en posant $w_{\{j,n+1\}} = \infty$. Ainsi, en résolvant au plus $n \times m$ problèmes de flot maximal, nous pouvons séparer de manière exacte les contraintes de capacité fractionnaire (4.20).

- *Contraintes de capacité arrondie*

Tournons maintenant notre attention sur les contraintes de capacité arrondie (4.21). Rappelons que ces contraintes ne peuvent être séparées de manière exacte en temps polynomial [28], cependant, étant donné un sous-ensemble d'indices $\mathcal{T} \subsetneq \{1,\ldots,m\}$, un sous-ensemble de clients $\mathcal{U}$ et une solution fractionnaire partielle $(\tilde{x},\tilde{y})$, nous pouvons rechercher le sous-ensemble de clients S tel que $\{j \in V : \exists (i,j) \in L_i,\ i \in \mathcal{U}\} \subset S$ et minimisant la quantité $f'(S)$ suivante :

$$f'(S) = \sum_{t=1,\ t\notin\mathcal{T}}^{m} \sum_{e\in\delta(S)} \tilde{x}_e^t - 2\left(\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil - |\mathcal{T}|\right).$$

En effet, $\mathcal{U}$ et $\mathcal{T}$ étant connus, la quantité $\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil - |\mathcal{T}|$ est une constante. En recherchant une coupe de capacité minimale séparant le dépôt du sommet $n+1$ sur le graphe support $\tilde{G}_{(\mathcal{U})}$ muni du vecteur de capacité w suivant :

- $\forall e = \{j,n+1\} \in E(\mathcal{U})$, $w_e = \infty$,
- $\forall e \in \tilde{E} \setminus E(C)$, $w_e = \sum_{t=1,\ t\notin\mathcal{T}}^{m} \tilde{x}_e^{t_0}$,

le sous-ensemble de sommets S défini par cette coupe contenant le sommet $n+1$ contiendra également tous les connecteurs des clients $i \in \mathcal{U}$. De plus, la valeur de la coupe sera $\sum_{t=1,\ t\notin\mathcal{T}}^{m} \sum_{e\in\delta(S)} \tilde{x}_e^{t_0}$. Si cette quantité est strictement inférieure à la quantité $2\left(\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil - |\mathcal{T}|\right)$, nous aurons déterminé une contrainte de capacité arrondie (4.20)

violée. Cette constatation est la base de la procédure de séparation heuristique des contraintes (4.20). Cette séparation est en fait basée sur deux heuristiques différentes pour construire des couples $(\mathcal{U},\mathcal{T})$ pour lesquels les contraintes de capacité arrondie associées sont testées par la suite. L'algorithme de séparation détaillée dans l'Algorithme 1 est donc appelé deux fois. Dans les deux cas la procédure *completer_ T(T,S)* donne l'indice $t' \in \{1,\ldots,m\} \setminus T$ maximisant le nombre d'affectation de S :

$$t' = \arg \max_{t \in \{1,\ldots,m\} \setminus T} \sum_{i \in S} \sum_{(i,j) \in A} y_{ij}^t.$$

Pour le premier lancement, la procédure *completer_ clients_ 1(S)* recherche le client $i' \in U \setminus S$ minimisant la capacité résiduelle des anneaux-étoiles nécessaires pour desservir la demande de $S \cup \{i'\}$:

$$i' = \arg \min_{i \in U \setminus S} \left\lceil \frac{d(S \cup \{i\})}{Q} \right\rceil - \frac{d(S \cup \{i\})}{Q}$$

Pour le second lancement, la procédure *completer_ clients_ 2(S)* recherche le client $i' \in U \setminus S$ ayant le plus grand nombre de connecteurs en commun avec les clients de S :

$$i' = \arg \max_{i \in U \setminus S} |L_i \cap \{\cup_{j \in S} L_j\}|$$

Entrées : La solution fractionnaire $s = (\tilde{x},\tilde{y})$.
Sorties : Une contrainte de capacité violée si possible.

pour chaque *client* $i \in U$ **faire**
  $S = \{i\}$;
  répéter
    si *l'inégalité définie par* S *et* $\mathcal{T} = \emptyset$ *est violée par* s **alors**
      retourner $(S,\emptyset)$;
    sinon
      pour t *de* 1 *à* m **faire**
        $\mathcal{T} \leftarrow \{t\}$;
        répéter
          si *l'inégalité définie par* S *et* $\mathcal{T}$ *est violée par* s **alors**
            retourner (S,T);
          fin
        tant que *completer_ T(*$\mathcal{T}$*)*;
      fin
    fin
  tant que *completer_ clients_ 1(S) ou completer_ clients_ 2(S)*;
fin

Algorithme 1 : Séparation heuristique des contraintes de capacité arrondie

5.2.1.2 Contraintes de connexité

Le lemme 5.2.1 suivant nous permet de limiter la séparation des inégalités de connexité généralisée (5.2) et (5.3) à celles définies sur une 2-couverture maximale pour U.

Lemme 5.2.1 *Si* $ax + by \leq 0$ *est une contrainte de connexité généralisée définie sur un anneau-étoile* $t_0 \in \{1,\ldots,m\}$, *un sous-ensemble* $S \subset V$ *et une 2-couverture* C *maximale pour* $U(S)$, *alors il existe une 2-couverture* C' *maximale pour* U *définissant la même contrainte de connexité pour l'anneau-étoile* t_0 *et le sous-ensemble* $S \subset V$.

Preuve. Soit C une 2-couverture maximale pour $U(S)$. Si C n'est pas maximale pour U, alors il existe une 2-couverture C' maximale pour U telle que $C \subset C'$. Montrons que C' définit la même contrainte de connexité (5.2) ou (5.3) définie sur un sous-ensemble S et un anneau-étoile t_0 que C. Soit $ax + by \leq 0$ la contrainte de connexité définie sur S, t_0 et C ; et soit $a'x + b'y \leq 0$ la contrainte de connexité définie sur S, t_0 et C'. Nous

avons, par définition des contraintes, $a' = a$. Soit $(i,j) \in A$, regardons les coefficients b_{ij} et b'_{ij} :

- si $\in C$ et $j \in S$, comme $C \subset C'$, $b'_{ij} = b_{ij}$.
- si $\in C$ et $j \notin S$, comme $C \subset C'$, $b'_{ij} = b_{ij}$.
- si $i \in C' \setminus C$ et $j \notin S$. Puisque $j \notin S$ nous avons $b'_{ij} = 0$, de plus $b_{ij} = 0$ car $i \notin C$, et donc $b'_{ij} = b_{ij}$.
- Enfin, il ne peut pas exister un arc $(i,j) \in A$ tel que $i \in C' \setminus C$ et $j \in S$. Dans le cas contraire C ne serait pas maximale pour $U(S)$ ce qui serait une contradiction.

Nous avons donc bien $b' = b$ puis, par extension, $a'x + b'y \leq 0$ est exactement la contrainte $ax + by \leq 0$. □

La séparation exacte des contraintes de connexité généralisées (5.2) peut se faire en temps polynomial grâce à une adaptation de l'algorithme proposé par Labbé et al. [75] pour les contraintes de connexité du problème de l'anneau-étoile. Soit une 2-couverture maximale pour U, et soit $\bar{Y}$ la somme suivante :

$$\bar{Y} = \sum_{t=1,t\neq t_0} \sum_{i\in C} \sum_{(i,j)\in A} y_{ij}^t.$$

Rechercher la contrainte $\sum_{e\in\delta(S_0)} x_e^{t_0} \geq 2\sum_{(i,j)\in A(C)} y_{ij}^{t_0}$ la plus violée revient à rechercher la plus grande violation de $\sum_{e\in\delta(S_0)} x_e^{t_0} + 2\bar{Y} \geq 2|C|$. Ceci est équivalent à la recherche d'un flot maximal sur le graphe $\tilde{G}^{t_0}$ sur lequel les capacités sont définies comme suit :

- $\forall e = \{j,n+1\} \in E(C)$, $w_e = 2\sum_{i\in C} y_{ij}^{t_0}$
- $\forall e \in \tilde{E} \cap E$, $w_e = x_e^{t_0}$

Soit $S' \subset \tilde{V} \setminus \{0\}$ avec $n+1 \in S'$ et tel que la capacité Δ de la coupe $\delta(S')$ soit minimale pour $\tilde{G}^{t_0}$. Si $\Delta \geq 2|C|$, il n'y a pas de contrainte de capacité définie sur C pour l'anneau-étoile t_0 qui soit violée par la solution courante. Sinon, $S = S' \setminus \{n+1\}$ et C induisent une contrainte de connexité (5.2) de violation maximale. Comme précédemment, l'algorithme peut retourner un ensemble S vide. Si cela se produit, nous relançons au plus n fois la même procédure en forçant chaque connecteur actif j à appartenir à l'ensemble S en posant $w_{\{j,n+1\}} = \infty$. Conséquemment, en recherchant un flot maximal pour chaque 2-couverture maximale pour U et pour chaque indice $t \in \{1,\ldots,m\}$, nous pouvons séparer de manière exacte les contraintes (5.2) en résolvant au plus $m \times n \times n_u$ problèmes de flot maximal.

L'algorithme de séparation des contraintes de connexité est basé sur la même idée que précédemment. Nous recherchons la contrainte $\sum_{e\in\delta(S_0)\setminus\{e'\}} x_e^{t_0} \geq \sum_{(i,j)\in A(C)} y_{ij}^{t_0}$ la plus

violée ce qui est équivalent à rechercher la plus grande violation de $\sum_{e\in\delta(S_0)\setminus\{e'\}} x_e^{t_0} + \bar{Y} \geq |C|$. Nous pouvons le faire en résolvant un problème de flot maximal sur le graphe $\tilde{G}^{t_0}$ munis des capacités suivantes :

- $w_{e'} = 0$
- $\forall e = \{j,n+1\} \in E(C)$, $w_e = \sum_{i\in C} y_{ij}^{t_0}$
- $\forall e \in \tilde{E} \cap E$, $e \neq e'$, $w_e = x_e^{t_0}$

Remarquons que ces contraintes sont séparées après les contraintes de 2-connexité, il s'ensuit que toute coupe définissant une contrainte violée contiendra nécessairement l'arête e'.

Une nouvelle fois si l'algorithme renvoie un ensemble S vide, nous forçons chaque connecteur actif à appartenir à l'ensemble S en posant $w_{\{j,n+1\}} = \infty$. Conséquemment, en recherchant un flot maximal pour chaque 2-couverture maximale pour U, pour chaque indice $t \in \{1,\ldots,m\}$ et pour chaque arête $e \in E$ telle que $\tilde{x}_e > 0$, nous pouvons séparer de manière exacte les contraintes (5.3) de manière exacte en résolvant au plus $m \times n \times n_u \times |E|$ problèmes de flot maximal.

5.2.2 Règle de branchement

Sur les instances auxquelles nous nous intéressons dans la Section 5.3, nous avons remarqué que quand les variables y sont entières, quasiment toutes les variables x sont également entières à la fin de la phase de coupes. Cela est lié au fait que si les affectations sont connues, notre problème revient alors à résoudre m Steiner TSP. Or, les inégalités que nous séparons pendant la base de séparation sont quasiment suffisantes pour décrire complètement le polytope associé au Steiner TSP sur ces instances [5], puisqu'elles sont planaires.

Pour tirer partie de cette constatation, il est utile de brancher en premier sur les variables d'affectation y. De plus, la règle de branchement peut être améliorée en décidant par quel anneau-étoile la demande d'un client sera desservie et non pas sur quel connecteur et quel anneau-étoile un client sera affecté. Autrement dit nous branchons sur les contraintes d'affectations :

$$\sum_{(i,j)\in L_i} y_{ij}^t = a,\ a \in \{0,1\}.$$

L'Algorithme 2 détaille la procédure complète.

Entrées : La solution fractionnaire $s = (\tilde{x},\tilde{y})$.
Sorties : Deux nouveaux noeuds de l'arbre de branchement.

Créer deux noeuds de l'arbre de branchement N_1 et N_2;
si $\exists(i,t) \in U \times \{1,\ldots,m\}$ *tel que* $\sum_{(i,j)\in A} y_{ij}^t \in]0,1[$ **alors**
  Soit (i,t) le couple tel que la somme $\sum_{(i,j)\in A} y_{ij}^t$ soit la plus fractionnaire;
  Ajouter la contrainte $\sum_{(i,j)\in A} y_{ij}^t = 1$ à N_1 ;
  Ajouter la contrainte $\sum_{(i,j)\in A} y_{ij}^t = 0$ à N_2 ;
sinon
  si $\exists(i,j,t) \in U \times V \times \{1,\ldots,m\}$ *tel que* $y_{ij}^t \in]0,1[$ **alors**
    Soit (i,j,t) le triplet tel que la variable y_{ij}^t soit la plus fractionnaire;
    Ajouter la contrainte $y_{ij}^t = 1$ à N_1 ;
    Ajouter la contrainte $y_{ij}^t = 0$ à N_2 ;
  sinon
    Soit (e,t) le couple tel que la variable x_e^t soit la plus fractionnaire;
    Ajouter la contrainte $x_e^t = 1$ à N_1 ;
    Ajouter la contrainte $x_e^t = 0$ à N_2 ;
  fin
fin

Algorithme 2 : Algorithme de Branchement

5.2.3 Heuristique primale

Nous avons également intégré une heuristique primale dédiée au 1-γ-RSP pour nous permettre d'obtenir rapidement une solution réalisable dont la valeur fournit une borne supérieure de la valeur optimale de l'instance traitée. Cette heuristique fonctionne de la manière suivante :

Pour chaque indice $t = 1,\ldots,m$ nous essayons de construire un cycle élémentaire $E(\mathcal{R}_t)$ grâce à un algorithme glouton qui démarre du dépôt et ajoute itérativement l'arête ayant la plus grande valeur $\tilde{x}_e^t$ parmi les arêtes incidentes au dernier noeud atteint. Si cette première phase a réussi à construire un cycle élémentaire pour chaque indice $t = 1,\ldots,m$, nous essayons ensuite d'affecter chaque client à l'un des cycles élémentaires en guidant notre recherche à l'aide des variables $\tilde{y}_{ij}^t$.

Soient RS un tableau d'anneaux-étoiles indicé sur $t = 1,\ldots,m$ et $s = (\tilde{x},\tilde{y})$ une solution fractionnaire de (NF), l'Algorithme 3 donne le détail de l'heuristique primale

dans laquelle nous utilisons les fonctions suivantes :

- *initialiser*($RS[t]$) : initialise la solution,
- *meilleur_ voisin*(n,G,s,$RS[t]$) : renvoie le sommet i voisin de n tel que i est accessible, c'est à dire qui n'appartient pas au cycle partiel de $RS[t]$, et tel que la variable $\tilde{x}^t_{\{n,i\}}$ soit de valeur maximale dans le voisinage de n ;
- *ajouter_ cycle*(n,$RS[t]$) : ajoute le sommet n au cycle partiel de $RS[t]$;
- *trier_ connecteurs*(i,G,s) : trie les arcs $(i,j) \in A$ par valeur $\tilde{y}^t_{ij}$ décroissante ;
- *appartient_ cycle*(j,$RS[t]$) : renvoie vrai si le connecteur j appartient au cycle de $RS[t]$, faux sinon ;
- *demande*(i) : renvoie la demande en bande passante du client i ;
- *demande*($RS[t]$) : renvoie la demande cumulée des clients affectés à l'anneau-étoile $RS[t]$;
- *ajouter_ client*(i,j,$RS[t]$) : ajoute l'arc (i,j) à l'anneau-étoile $RS[t]$.

Entrées : Le graphe $G = (V,E,A)$,
La capacité d'un anneau-étoile Q,
La solution fractionnaire $s = (\tilde{x},\tilde{y})$.
Sorties : Une solution sous la forme d'un tableau d'anneaux-étoiles RS.

```
//Construction des cycles par une procédure gloutonne;
pour t de 1 à m faire
    initialiser(RS[t]);
    Node n = depot(G);
    ajouter_cycle(n, RS[t]);
    répéter
        Node n ← meilleur_voisin(n,G,s,RS[t]);
        si n ≠ null alors
            ajouter_cycle(n, RS[t]);
        sinon
            retourner faux;
        fin
    tant que node ≠ depot(G);
fin

//Affectation des clients par une procédure gloutonne;
pour chaque client i ∈ U faire
    Liste_connecteur = trier_connecteurs(i,G,s);
    client_non_affecté = vrai;
    pour chaque (j,t) ∈ Liste_connecteur faire
        si appartient_cycle(j,RS[t]) et demande(i) ≤ Q - demande(RS[t]) alors
            ajouter_client(i,j,RS[t]);
            client_non_affecté = faux;
            sortir;
        fin
    fin
    si client_non_affecté alors
        retourner faux;
    fin
fin
retourner RS;
```

Algorithme 3 : Heuristique Primale

5.3 Résultats expérimentaux

L'algorithme de Branch-and-Cut présenté dans les sections précédentes a été implémenté en C++ grâce à l'environnement SCIP 3.0 (voir [1] pour plus de détail sur ce logiciel) couplé au solveur Cplex 12.4. Le graphe support de nos instances a été implémenté grâce à la bibliothèque Lemon 1.2.1. Les expérimentations ont été menées sur un ordinateur de bureau équipé d'un processeur Intel2 Duo cadencé à 3.33 GHz et d'une mémoire vive de 8 Go. Nous avons considéré deux classes d'instances, la première générée aléatoirement, la seconde obtenue à partir d'un réseau SDH réel déployé sur Paris et sa région.

La première classe d'instances a été créée de la manière suivante :
Étant donnés les paramètres du problème $|U|$, $|W|$ et m, nous générons aléatoirement un nuage de points dont les coordonnées sont comprises dans un plan défini par les coordonnées $[0{,}100 * |V|] \times [0{,}100 * |V|]$. Nous considérons ensuite un graphe complet induit par cet ensemble de points où chaque arête $\{i,j\}$ est pondérée par la distance euclidienne entre les points i et j. Nous trions les arêtes par longueur décroissante et supprimons chaque arête intersectant une arête plus petite. L'ensemble des connecteurs d'un client i est alors composé de tous ses voisins et lui-même : $L_i = \{(i,j) : j \in \delta(i)\} \cup \{(i,i)\}$ et le coût d'affectation c_{ij} est la distance euclidienne entre les noeuds i et j. Enfin, nous fixons la capacité Q et les valeurs d_i de manière à ce qu'il existe des solutions comportant exactement m anneaux-étoiles. Pour chaque triplet $(|U|,|W|,m)$ nous générons dix instances.

Les instances réalistes que nous avons traitées ont été obtenues à partir du réseau optique parisien de COLT, l'un des principaux fournisseurs de solutions télécoms et hébergement pour les entreprises en Europe. La taille de ce réseau étant trop importante (il s'étend de la Défense à Évry), nous l'avons morcelé de la manière suivante :

Soit $\alpha \in]0{,}1]$ et un dépôt i, nous avons considéré tous les points du réseau dont la distance euclidienne au dépôt était inférieure ou égale à αl_{max} où l_{max} est la plus grande distance entre deux points du réseau. Pour s'assurer que l'instance obtenue soit réalisable, nous avons dû ajouter des arêtes jusqu'à obtenir un graphe 2-connexe. Grâce à cette procédure, nous avons pu obtenir trois instances réalistes.

Dans toutes les tables qui suivent, la première colonne donne les caractéristiques de l'instances : le nombre de dépôts $|N|$, le nombre de clients $|U|$, le nombre de sommets Steiner $|W|$ et le nombre minimal d'anneaux-étoiles d'une solution réalisable m. Les valeurs qui sont reportées sont les moyennes obtenues sur 10 instances d'une taille donnée.

TAB. 5.1 – *Détail sur la densité des instances aléatoires pour* $|N| = 1$

$\|N\|$	$\|U\|$	$\|W\|$	m	$\|E\|$	$min(\|\delta(i)\|)$	$max(\|\delta(i)\|)$	$\|A\|$	$min(\|L_i\|)$	$max(\|L_i\|)$
1	10	10	3	49,3	4,8	8,7	55,2	4,1	7,1
1	10	10	4	50	5	9,2	55,3	3,8	7,5
1	10	10	5	49,3	5	8,8	53,9	4,1	6,9
1	10	20	3	76,6	5	9,4	58,2	3,9	7,6
1	10	20	4	77,3	5	9,4	59,8	4,3	7,8
1	10	20	5	78,8	5	9,4	61,5	4,7	8
1	10	40	3	132,1	4,9	10,1	62,8	4,6	8,3
1	10	40	4	131,5	5	9,7	62,9	4,6	8
1	10	40	5	133	4,9	9,8	64,6	5	8,4
1	15	10	3	63,1	5,1	9,1	83,7	3,7	7,7
1	15	10	4	63,8	4,8	9,4	84,2	3,6	7,6
1	15	10	5	64,5	5	9,5	85,8	3,9	7,9
1	15	20	3	89,3	4,9	9,5	89,4	3,8	8,5
1	15	20	4	89,2	4,7	9,4	85,5	3,8	7,8
1	15	20	5	90,6	5	9,7	91,8	4,2	8,3
1	15	40	3	146,6	5	10	95,5	4,7	8,7
1	15	40	4	145,4	4,9	10,1	92,6	4,2	8,6
1	15	40	5	144,5	4,8	10,1	93,3	4,7	8

La Table 5.1 donne plus d'informations sur la densité des instances aléatoires. Nous y reportons le nombre d'arêtes $|E|$ et le nombre d'arcs $|A|$. Nous y indiquons également le nombre minimal (*resp.* maximal) d'arêtes incidentes à un sommet $min(|\delta(i)|)$ (*resp.* $max(|\delta(i)|)$). Enfin, la taille minimale (*resp.* maximale) d'un ensemble d'affectation est donné par $min(|L_i|)$ (*resp.* $max(|L_i|)$), ce chiffre correspond au nombre minimal (*resp.* maximal) d'arcs sortant d'un client $i \in U$. Les Tables 5.3, 5.5 et 5.6 résument les performances de notre algorithme sur ces deux classes d'instances.

La Table 5.2 compare la qualité des relaxations linéaires des différentes formulations. Elle indique le gap relatif entre la meilleure solution connue et la valeur de la relaxation linéaire de la formulation flot (colonne $F1$), de la formulation naturelle (colonne NF) et de la formulation naturelle avec nos contraintes additionnelles (colonne NFC). On remarque que les valeurs de relaxation des formulations naturelles et flot sont très proches lorsque le nombre d'anneaux-étoiles m est faible. Lorsque celui-ci augmente, la formulation flot donne de meilleurs résultats. Dans tous les cas cependant, la formulation naturelle renforcée par nos contraintes additionnelles donne une bien meilleure borne que la formulation flot, ce qui donne une première indication sur l'utilité de ces contraintes. Les deux dernières colonnes donnent la proportion d'instances pour les-

TAB. 5.2 – *Comparaison des relaxations linéaires des formulations flot et naturelle*

Instances				Écarts relatifs			Meilleure valeur	
$\|N\|$	$\|U\|$	$\|W\|$	m	$F1$	NF	NFC	$\#FB$	$\#NFB$
1	10	10	3	19,11%	23,99%	8,13%	80,00%	20,00%
1	10	10	4	16,27%	27,81%	6,98%	90,00%	10,00%
1	10	10	5	10,96%	43,02%	5,61%	100,00%	0,00%
1	10	20	3	20,29%	18,53%	5,40%	50,00%	50,00%
1	10	20	4	16,75%	34,78%	8,77%	100,00%	0,00%
1	10	20	5	12,58%	40,11%	6,06%	100,00%	0,00%
1	10	40	3	22,58%	22,87%	6,12%	70,00%	30,00%
1	10	40	4	17,07%	31,97%	7,61%	100,00%	0,00%
1	10	40	5	13,42%	39,79%	6,12%	100,00%	0,00%
1	15	10	3	20,58%	21,93%	9,13%	60,00%	40,00%
1	15	10	4	17,69%	31,35%	9,59%	100,00%	0,00%
1	15	10	5	20,14%	43,91%	14,40%	100,00%	0,00%
1	15	20	3	20,15%	23,21%	8,02%	50,00%	50,00%
1	15	20	4	17,67%	38,25%	10,39%	100,00%	0,00%
1	15	20	5	20,17%	41,93%	13,23%	100,00%	0,00%
1	15	40	3	20,54%	26,16%	8,54%	60,00%	40,00%
1	15	40	4	20,21%	31,15%	8,98%	70,00%	30,00%
1	15	40	5	22,62%	40,35%	14,26%	100,00%	0,00%

quelles la formulation flot est meilleure (colonne $\#FB$) que la formulation naturelle sans contrainte additionnelle (colonne $\#NFB$). Si l'avantage va à la formulation flot, nous avons là la preuve que ces deux formulations sont incomparables.

La Table 5.3 illustre l'efficacité de notre heuristique primale. Elle donne l'écart relatif (G1st) entre la meilleure borne supérieure connue et la première solution primale obtenue lors du parcours de l'arbre de branchement. Les autres colonnes donnent plus de détails sur ces deux solutions :

T1st (*resp.* TB^{st})	:	le temps de calcul (en secondes) nécessaire pour obtenir la première (*resp.* meilleure) solution.
$\#$N1st (*resp.* $\#$NB^{st})	:	le nombre de noeuds de l'arbre de branchement traités avant d'obtenir la première (*resp.* meilleure) solution.
H1st (*resp.* HB^{st})	:	la proportion d'instances pour lesquelle notre heuristique a fourni la première (*resp.* meilleure) solution.

La première solution est toujours obtenue en moins de trente minutes. Remarquons cependant que dans la plupart des cas, la première solution est trouvée en moins de

TAB. 5.3 – *Efficacité de l'heuristique primale*

Instances				1^{ere} solution				Meilleure solution		
$\|N\|$	$\|U\|$	$\|W\|$	m	G1^{st}	T1^{st}	#N1^{st}	H1^{st}	TB^{st}	#NB^{st}	HB^{st}
1	10	10	3	12,12%	5,471	5,3	30,00%	7,657	53,9	0,00%
1	10	10	4	10,46%	8,23	15,5	20,00%	22,147	292,1	0,00%
1	10	10	5	8,70%	9,993	16,8	30,00%	34,858	358,6	0,00%
1	10	20	3	8,56%	16,183	9,4	10,00%	17,503	21,7	0,00%
1	10	20	4	8,53%	35,0903	19,94	10,00%	74,4013	247,37	0,00%
1	10	20	5	7,57%	46,769	20,2	20,00%	208,45	1063	0,00%
1	10	40	3	11,94%	178,469	4,9	40,00%	186,653	25,6	0,00%
1	10	40	4	12,37%	174,977	18,8	20,00%	216,94	116,5	0,00%
1	10	40	5	6,27%	326,747	34,1	10,00%	654,719	640,2	0,00%
1	15	10	3	7,23%	25,241	24,9	10,00%	35,751	109,9	0,00%
1	15	10	4	11,43%	49,015	42,6	20,00%	263,788	1071,5	0,00%
1	15	10	5	4,19%	96,795	141,8	0,00%	1534,713	5759,3	0,00%
1	15	20	3	15,24%	81,107	20,1	30,00%	110,277	121,3	0,00%
1	15	20	4	13,01%	233,69	51,6	30,00%	1209,482	2714,3	0,00%
1	15	20	5	4,93%	223,406	124,4	10,00%	2151,11	4112	0,00%
1	15	40	3	12,26%	421,567	35,9	20,00%	531,727	176,3	0,00%
1	15	40	4	3,06%	723,913	58,7	0,00%	2301,629	1548,1	0,00%
1	15	40	5	6,13%	1238,787	190,9	10,00%	1366,935	324,2	0,00%

dix minutes. Cela tend à montrer que notre algorithme peut fournir rapidement une bonne solution.

Les solutions réalisables ne sont que très rarement trouvées par notre heuristique primale : elle ne trouve la première solution que dans 17,78% des cas, et jamais la meilleure. Ces résultats sont assez étonnants puisque lors d'une précédente campagne d'expérimentation [48] menée sous SCIP 2.0 pour des instances similaires à celles-ci, notre heuristique primale montrait de très bon résultats. Elle fournissait la meilleure (*resp.* première) solution dans 77% (*resp.* 85%) des cas. L'origine de ce comportement pourrait être les améliorations apportées aux traitement automatiques de SCIP entre la version 2 et la version 3.

Pour les autre tables, les colonnes donnent les résultats suivants :

#Opt	:	le nombre d'instances résolues à l'optimalité.
FG	:	l'écart relatif entre la meilleure solution trouvée et la meilleure borne inférieure fournie.
RG	:	l'écart relatif entre la meilleure solution trouvée et la borne inférieure obtenue à la racine de l'arbre de branchement.
#N	:	le nombre de noeuds dans l'arbre de branchement.
TT	:	le temps total de calcul en secondes (limité à deux heures).
#con	:	le nombre de contraintes de 2-connexité séparées au cours de la résolution.
#ineq	:	le nombre de contraintes séparées à la racine de l'arbre de branchement pour chaque famille d'inégalités.

La Table 5.4 compare l'efficacité de notre algorithme avec ou sans nos contraintes additionnelles de connexité (5.3) et de capacité (4.20), (4.21). Cette table ne présente que les instances qui ont pu être résolues dans les deux cas. Nous pouvons remarquer que l'ajout de nos inégalités a permis d'améliorer nettement à la fois la borne inférieure fournie par la relaxation linéaire, la taille de l'arbre de branchement et le nombre d'instances résolues optimalement. De plus, le temps total de calcul est généralement meilleur ou équivalent. Cela témoigne de l'efficacité des algorithmes de séparation développés pour ces inégalités.

La Table 5.5 résume les résultats obtenus sur les instances aléatoires. Chaque ligne donne les valeurs moyennes obtenues sur les dix instances de chaque classe. Pour chaque

TAB. 5.4 – *Impact des contraintes additionnelles sur la formulation naturelle*

Instances				sans contraintes					avec contraintes				
$\|N\|$	$\|U\|$	$\|W\|$	m	FG	#Opt	RG	#N	TT	FG	#Opt	RG	#N	TT
1	10	10	3	0,00%	10	23,96%	240,6	7,125	0,00%	10	8,13%	90,6	8,462
1	10	10	4	0,00%	8	27,81%	1834,3	49,587	0,00%	10	6,98%	456,1	24,807
1	10	10	5	0,00%	6	43,02%	9034,8	308,041	0,00%	10	5,61%	1289,8	62,753
1	10	20	3	0,00%	8	18,49%	94,2	6,192	0,00%	10	5,40%	40,5	18,216
1	10	20	4	0,00%	9	34,51%	2400,4	197,247	0,00%	10	8,77%	672,2	105,383
1	10	20	5	0,00%	9	40,11%	8527,8	604,531	0,00%	10	6,06%	2463,3	275,216
1	15	10	3	0,00%	10	20,73%	275,3	20,71	0,00%	10	9,13%	183,2	38,113
1	15	10	4	0,00%	9	31,19%	16474,2	1411,306	0,00%	10	9,59%	4097,7	520,201
1	15	10	5	14,83%	2	44,18%	33381,3	6117,107	6,60%	4	14,40%	21971,2	5358,002
1	10	20	3	0,00%	8	23,24%	428,4	81,938	0,00%	10	8,02%	188,1	116,023
1	10	20	4	2,01%	6	38,23%	18961,3	4381,734	0,00%	10	10,39%	7104,5	2029,519
1	10	20	5	15,70%	1	43,06%	20260,7	6679,817	7,43%	3	13,23%	12510,4	5884,216

TAB. 5.5 – *Résultats expérimentaux sur les instances aléatoires*

Instances				Écarts relatifs et temps de calcul				Contraintes additionnelles			
$\|N\|$	$\|U\|$	$\|W\|$	m	FG	*#Opt*	#N	TT	# 1-con	# capa	# Rcapa	# 2-con
1	10	10	3	0,00%	10	90,6	8,462	700,3	784,8	140,3	4728,1
1	10	10	4	0,00%	10	456,1	24,807	838,5	979,3	121,4	16377
1	10	10	5	0,00%	10	1289,8	62,753	1323,8	1504,1	135,8	32576,9
1	10	20	3	0,00%	10	40,5	18,216	968	1198,8	206,9	4754,4
1	10	20	4	0,00%	10	672,2	105,383	1545,5	2121,6	229	56223,7
1	10	20	5	0,00%	10	2463,3	275,216	1938,6	2657,9	219,8	95947,9
1	10	40	3	0,00%	10	42,6	188,35	2063,1	3147,6	458,3	11369,7
1	10	40	4	0,00%	10	359,8	269,057	2622	4028	403,4	50489,1
1	10	40	5	0,00%	10	2305,9	1032,583	3238,2	5149,8	414,3	225563,5
1	15	10	3	0,00%	10	183,2	38,113	991,4	717,1	196,7	13266,2
1	15	10	4	0,00%	10	4097,7	520,201	1428,4	1267,4	220,2	254672
1	15	10	5	6,60%	4	21971,2	5358,002	2116,2	1747,9	238,8	2327174,7
1	15	20	3	0,00%	10	188,1	116,023	1616,3	1291,5	320,8	27513,2
1	15	20	4	0,00%	10	7104,5	2029,519	2558,8	2482,3	362,5	707042,2
1	15	20	5	7,43%	3	12510,4	5884,216	2870,2	2860,6	335,6	2253833,8
1	15	40	3	0,00%	10	397,7	591,847	2708,3	2407,7	579,2	87331,7
1	15	40	4	0,77%	8	3780,7	3431,673	3975,6	4697	604,5	786752,1
1	15	40	5	12,88%	1	4663,5	6766,792	5463	6371,7	597,1	1781853,9

taille d'instance, au moins l'une des instances a été résolue optimalement, et pour chacune d'elle une solution réalisable a pu être fournie. Nous avons pu résoudre des instances allant jusqu'à 15 clients, 40 sommets Steiner et 5 anneaux-étoiles. Nous remarquons également que la taille de l'arbre de branchement dépend fortement du nombre d'anneaux-étoiles. Cela est dû au fait que la symétrie de notre formulation augmente en fonction de m. Remarquons également que le nombre de contraintes de 2-connexité générées au cours de la résolution dépasse les deux millions.

Enfin, la Table 5.6 donne les résultats obtenus sur les instances réalistes. Nous voyons immédiatement que ces instances sont plus faciles à résoudre que nos instances aléatoires. Notre algorithme parvient à résoudre des instances de la taille d'un quartier parisien (5% du réseau global déployé sur Paris et sa banlieue).

TAB. 5.6 – *Résultats expérimentaux sur les instances réalistes*

Instances					Écarts relatifs et temps de calcul				Contraintes additionnelles			
α	$\|N\|$	$\|U\|$	$\|W\|$	m	FG	RG	#N	TT	# 1-con	# capa	# Rcapa	# 2-con
5%	1	23	104	3	0%	12,01%	1903	10,60'	2789	3118	1616	226836
5.3%	1	35	127	3	0%	7,61%	5442	64,73'	9503	9096	3782	1018048
5.4%	1	42	136	3	18,33%	21,98%	7300	120'	7668	7527	4914	2473000

5.4 Conclusion

Dans ce chapitre, nous avons réalisé une étude faciale partielle du polytope $\mathcal{P}$, dominant du polytope associé à la formulation naturelle du 1-γ-*RSP*. Nous avons donné des conditions nécessaires et suffisantes pour que les contraintes triviales, les contraintes de degré et les contraintes de connexité définissent des facettes de $\mathcal{P}$. Nous avons également donné des algorithmes de séparation exacts et heuristiques pour ces contraintes.

Ces résultats théoriques nous ont permis de construire un algorithme de Branch-and-Cut pour le 1-γ-*RSP*. Nous avons appliqué cet algorithme à des instances du problème de conception de réseaux en anneaux-étoiles générées aléatoirement ou obtenues à partir d'un réseau réel déployé en région parisienne.

Les résultats expérimentaux ont montré que l'ajout de nouvelles inégalités valides a permis de significativement améliorer la qualité de la borne inférieure et a permis à notre algorithme de résoudre optimalement des instances de taille importante : en effet, il est envisageable de l'utiliser pour résoudre des instances à l'échelle d'un quartier, ou pour déterminer un petit ensemble d'anneaux à l'échelle d'une ville.

Chapitre 6

Génération de colonnes et inégalités valides pour le k-γ-RSP

Dans ce chapitre nous abordons la résolution de la version multi-dépôts du k-γ-RSP. Comme l'ont montré les résultats expérimentaux de l'algorithme de Branch-and-Cut du chapitre 5, la formulation utilisée pour le 1-γ-RSP souffre déjà de nombreuses symétries et la gestion de plusieurs dépôts ne ferait qu'accroître les difficultés. Pour ces deux raisons, à savoir limiter l'impact de la symétrie et pouvoir prendre en compte facilement la présence de plusieurs dépôts, nous proposons une formulation proche de celle utilisée pour le CVRP, appelée fréquemment formulation *set partitionning* (*cf.* chapitre 2). D'autre part, nous proposons plusieurs inégalités valides pour renforcer cette formulation.

6.1 Problème maître

Dans les sections qui suivent, nous noterons Ω l'ensemble des anneaux-étoiles valides sur le graphe G. Étant donné un anneau-étoile $r \in \Omega$, nous considérons les vecteurs a^r et b^r définissant l'anneau-étoile r de la façon suivante : $a^r_{ij} = 1$ si le client i est affecté au connecteur j sur l'anneau-étoile r, 0 sinon ; et $b^r_e = 1$ si l'anneau-étoile r utilise l'arête e, 0 sinon. Enfin, c_r sera le coût de l'anneau-étoile r : $c_r = \sum_{(i,j)\in A} a^r_{ij} c_{ij} + \sum_{e\in E} b^r_e l_e$.

Nous associons une variable binaire θ_r à chaque anneau-étoile $r \in \Omega$. Cette variable indiquera si l'anneau-étoile r est sélectionné ($\theta_r = 1$) ou non ($\theta_r = 0$) dans la solution. Le k-γ-RSP se formule alors naturellement par le programme linéaire en nombres

entiers suivant :

$$\begin{array}{lll} (MP(\Omega)) & \min \sum_{r\in\Omega} c_r\theta_r & \\ s.t. & \sum_{r\in\Omega}\sum_{(i,j)\in A} a_{ij}^r\theta_r = 1 & \forall i \in U, \quad (6.1)\\ & \sum_{r\in\Omega} b_e^r\theta_r \leq \gamma_e & \forall e \in E, \quad (6.2)\\ & \theta_r \in \mathbb{N} & \forall r \in \Omega. \quad (6.3) \end{array}$$

Les contraintes (6.1) assurent que chaque client soit connecté, les contraintes (6.2) limitent l'utilisation d'une arête à sa capacité.

Un tel modèle ne peut être résolu par une approche standard de type Branch-and-Bound. Cela est dû à la taille de l'ensemble Ω qui augmente de manière exponentielle avec le nombre de sommets du graphe. Le calcul de la relaxation linéaire de $(MP(\Omega))$ peut néanmoins se faire grâce à un algorithme de génération de colonnes. Nous considérons donc un ensemble d'anneaux-étoiles de taille raisonnable $\Omega' \subset \Omega$ et considérons le problème maître $(MP(\Omega'))$ limité aux variables θ_r, $r \in \Omega'$. Nous appellerons $(MP(\Omega'))$ le problème restreint.

6.2 Problème auxiliaire

6.2.1 Définition

Soient λ_i la variable duale associée à la contrainte d'affectation (6.1) du client i et ω_e la variable duale associée à la contrainte de capacité (6.2) de l'arête e. Le programme dual associé au problème maître restreint $MP(\Omega')$ est alors

$$\begin{array}{lll} & \max \sum_{i\in U} \lambda_i + \sum_{e\in E} \gamma_e\omega_e & \\ s.t. & \sum_{i\in U}\sum_{(i,j)\in A} a_{ij}^r\lambda_i + \sum_{e\in E} b_e^r\omega_e \leq c_r & \forall r \in \Omega' \quad (6.4)\\ & \omega_e \leq 0 & \forall e \in E \quad (6.5) \end{array}$$

Soit θ^* une solution de $(MP(\Omega'))$ et (λ^*,ω^*) la solution duale associée. Le coût réduit $\hat{c_r}$ d'un anneau-étoile dans le problème maître $MP(\Omega')$ est :

$$\hat{c_r} = c_r - \sum_{i\in U}\sum_{(i,j)\in A} a_{ij}^r\lambda_i^* - \sum_{e\in E} b_e^r\omega_e^*.$$

L'objectif du problème auxiliaire est de déterminer s'il existe un anneau-étoile $r \in \Omega \setminus \Omega'$ tel que $\hat{c_r}$ soit strictement négatif, et, si oui, de le produire. Ici, le problème

auxiliaire est équivalent au problème de recherche d'un anneau-étoile de coût minimal sur le graphe $G = (V,E,A)$ où le poids d'une arête $e \in E$ devient $\hat{l_e} = l_e - \omega_e^*$ et le poids d'un arc $(i,j) \in A$ devient $\hat{c_{ij}} = c_{ij} - \lambda_i^*$. La recherche d'un tel anneau-étoile est NP-difficile puisque le TSP en est un cas particulier.

Dans les sous-sections suivantes nous étudions différentes approches permettant de résoudre ce problème malgré tout.

6.2.2 Résolution algorithmique

L'approche la plus plébiscitée dans la littérature du CVRP est l'utilisation d'un algorithme de programmation dynamique. C'est également celle qui a été utilisée par Hoshino et al. dans [64] pour le 1-1-RSP. Pour chaque dépôt d, nous voulons rechercher un anneau-étoile de coût minimal passant par d. Nous montrons ici comment cela peut se réduire à la recherche de $|N|$ plus courts chemins élémentaires avec contraintes de ressources (ESPPRC).

Tout d'abord, notons U_j l'ensemble des clients ayant j comme connecteur : $U_j = \{i \in U : (i,j) \in A\}$, $\forall j \in W$. Si nous recherchons un anneau-étoile de coût minimal passant par le dépôt $d \in N$, nous construisons un graphe orienté $G' = (V',A')$ où V' sera composé de deux sommets d et d', de deux ensembles L et L' de sommets et d'un ensemble U' de sommets correspondant à des sous-ensembles de clients. G' est alors construit de la manière suivante :

- nous créons un noeud origine d et un noeud destination d'.
- Pour chaque dépôt $i \in N \setminus \{d\}$ nous créons un noeud i dans L.
- Pour chaque sommet Steiner $j \in W$:
  - Si $U_j = \emptyset$
    - nous créons un noeud j dans L.
  - Si $U_j \neq \emptyset$
    - nous créons un sommet j dans L et un sommet j' dans L'
    - nous ajoutons un arc de coût nul (j,j') dans A',
    - pour tout ensemble de client $\mathcal{F} \subset U_j$, nous ajoutons dans U' un noeud u associé à l'ensemble $\mathcal{F}$,
    - nous ajoutons également deux arcs (j,u) et (u,j') à A', de coûts $c_{ju} = 0$ et $c_{uj'} = \sum_{i \in \mathcal{F}} c_{ij} - \lambda_i^*$

- enfin pour chaque arête $e = \{i,j\} \in E$ nous ajoutons 2 arcs (i,j') et (j,i') à A' de coût $c_{ij'} = c_{ji'} = l_e - \omega^*$ où j' (*resp.* i') est la copie du sommet j (*resp.* i) dans L' si elle existe, la copie du sommet j (*resp.* i) dans L sinon.

Pour un sommet $j \in V'$ nous noterons $o(j)$ le sommet de V à l'origine de la création de j. Le vecteur ressource sera composé d'une ressource unitaire s_i pour chaque sommet $i \in V \setminus \{d\}$ et d'une ressource en bande-passante b limitée à Q. La consommation des ressources sera la suivante :

- les sommets $i \in L' \cup \{d,d'\}$ ne consommeront aucune ressource,
- les sommets $j \in L$ consommeront la ressource unitaire $s_{o(j)}$,
- soit $u \in U$ un sommet associé au sous-ensemble de clients $\mathcal{F}$ et relié à un sommet $j \in L'$, u consommera la ressource unitaire s_i de chaque client $i \in \mathcal{F}$ si $i \neq o(j)$ ainsi qu'une quantité $d_u = \sum_{i \in \mathcal{F}} d_i$ de la ressource b.

Par construction nous avons le résultat suivant :

Proposition 6.2.1 *Rechercher un anneau-étoile de coût minimal sur G est équivalent à la recherche d'un plus court chemin élémentaire avec contrainte de ressource sur le graphe G'.*

Rappelons que si l'approche par programmation dynamique a montré de bons résultats dans la littérature du CVRP, c'est principalement parce que la contrainte d'élémentarité peut être relâchée sous certaines hypothèses, en particulier lorsque le graphe support vérifie les inégalités triangulaires (*cf.* chapitre 2). Dans le cas du k-γ-RSP, la structure en anneaux élémentaires doit être vérifiée dans le graphe d'origine puisqu'elle est essentielle à la fiabilité d'un réseau SDH. L'astuce de réduction du CVRP à la recherche de tournées disjointes dans un graphe complet des plus courts chemins ne peut être utilisée ici. Conséquemment la relaxation de l'élémentarité des tournées ne peut être appliquée dans notre cas. Nous devons donc résoudre exactement l'ESPPRC et non sa relaxation. Une solution possible est alors d'utiliser l'algorithme de programmation dynamique proposé par Feillet et al. pour l'ESPPRC [39]. Malheureusement, les premières expérimentations ont montré que, même sur les plus petites de nos instances, l'algorithme de programmation dynamique consommait beaucoup de mémoire et de temps (plus d'une heure de calcul) menant ainsi à une procédure inefficace de génération de colonnes. Cela est principalement dû à la taille importante du graphe G'. À titre d'exemple, la figure 6.1 montre le graphe obtenu à partir du graphe de la figure 3.8-a qui ne contenait que 6 clients et 14 sommets Steiners. Sur la figure 6.1, les sommets de U' sont représentés par des cercles et les sommets de L et L' sont représentés

par des triangles, de couleur identique lorsque les deux sommets ont le même sommet d'origine.

6.2.3 Résolution par un algorithme de Branch-and-Cut

Puisque l'approche par programmation dynamique ne permet pas une génération efficace de colonnes, nous proposons dans cette section de résoudre le problème auxiliaire grâce à un algorithme de Branch-and-Cut inspiré de nos travaux pour le 1-γ-RSP.

À chaque arête $e \in E$ nous associons une variable binaire x_e. À chaque arc $(i,j) \in A$ nous associons une variable binaire y_{ij}. La recherche d'un anneau-étoile de coût minimal sur G est équivalent à la résolution du programme linéaire en nombres entiers (AP) suivant :

$$
\begin{array}{lll}
(AP) & \min \sum_{e\in E}(l_e - \omega_e^*)x_e + \sum_{(i,j)\in A}(c_{ij} - \lambda_i^*)y_{ij} & \\
st. & \sum_{(i,j)\in A} y_{ij} \geq 1 & (6.6)\\
 & \sum_{(i,j)\in A} y_{ij} \leq 1 \quad \forall i \in U & (6.7)\\
 & \sum_{(i,j)\in A} d_i y_{ij} \leq Q & (6.8)\\
 & \sum_{e\in\delta(i)} x_e \leq 2 \quad \forall i \in V & (6.9)\\
 & \sum_{e\in\delta(S)\setminus\{e'\}} x_e \geq 2\sum_{(i,j)\in A(\mathcal{C})} y_{ij} \quad \forall S, \emptyset \neq S \subseteq V\setminus\{0\}, & (6.10)\\
 & \qquad\qquad \forall C \subset U(S) & \\
 & \sum_{e\in\delta(S)} x_e \geq \sum_{e\in\delta(i)} x_e, \quad \forall S \subseteq V\setminus\{0\}, \forall i \in S & (6.11)
\end{array}
$$

On constate que cette formulation est très proche de celle proposée en section 4.3. Nous retrouvons des contraintes d'affectation (6.6) et (6.7). La première assure que l'anneau-étoile recherché dessert au moins un client, les secondes limitent le nombre d'affectation de chaque client à 1. La contrainte de capacité (6.8) impose le respect de la bande passante. Les contraintes (6.9) et (6.10) assurent la structure cyclique de la solution. Nos coûts étant cette fois-ci quelconques, nous devons ajouter les contraintes (6.11) pour interdire la sélection d'arêtes isolées. La résolution de ce programme linéaire est possible grâce à l'algorithme de Branch-and-Cut présenté en sous-section 7.1.5.

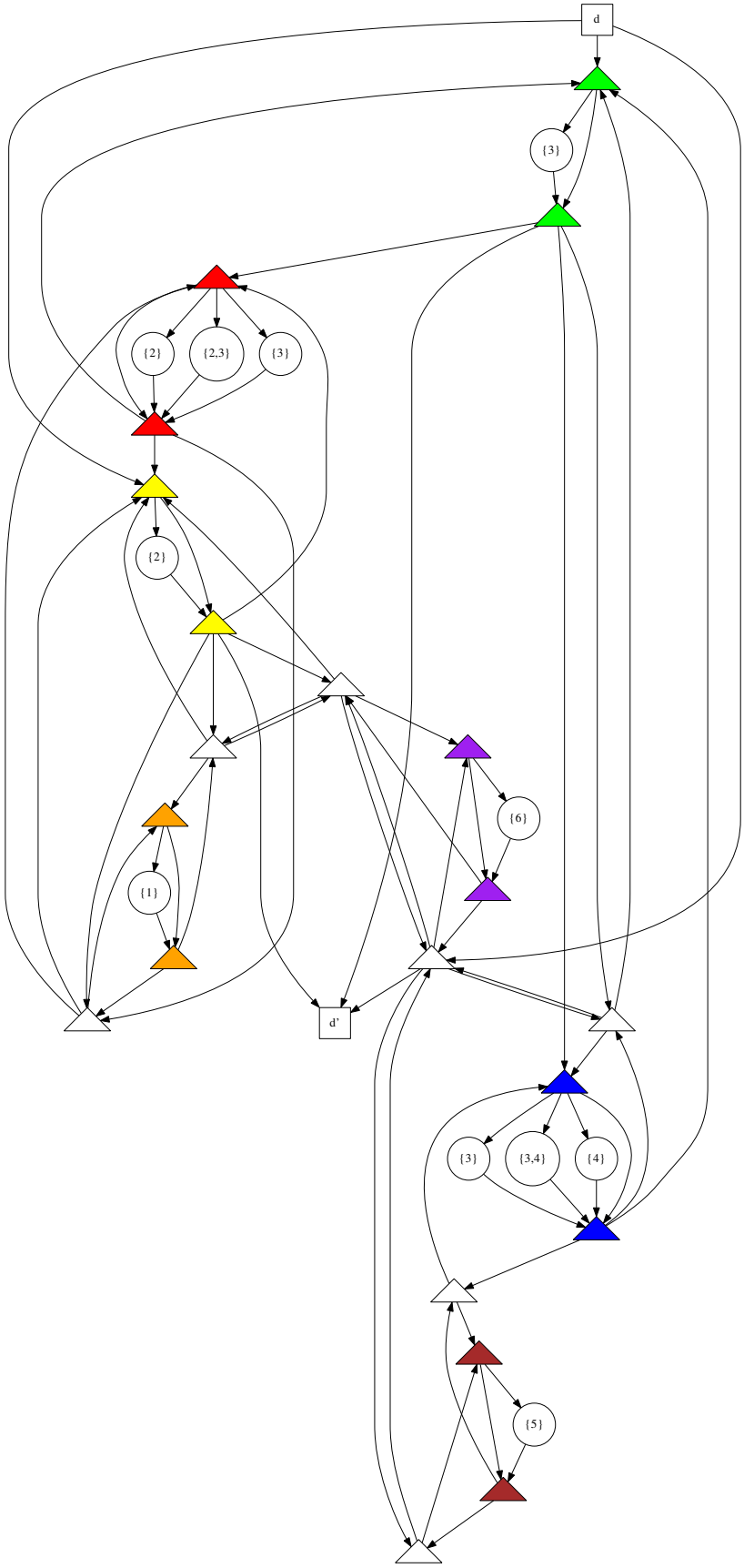


FIG. 6.1 – *Le graphe construit à partir de la figure 3.8-a pour le problème auxiliaire*

6.3 Inégalités valides

Afin d'améliorer la qualité de la relaxation linéaire de la formulation set partitionning précédente, et ainsi limiter la taille de l'arbre de branchement, il est important d'ajouter plusieurs familles d'inégalités valides basées sur la structure de la formulation. Nous présentons ici trois familles d'inégalités, l'une utilisée classiquement pour le polytope du CVRP, les autres issues du polytope du Stable. Nous les renforçons également en étudiant des propriétés de dominance.

6.3.1 Contraintes de capacité

Soit un sous-ensemble de clients $\mathcal{U} \subset U$. Les contraintes de capacité utilisées dans notre algorithme de Branch-and-Cut pour le 1-γ-RSP peuvent être ici adaptées. Nous nous limiterons aux contraintes de capacité arrondie qui dominent les contraintes de capacité fractionnaire.

Soit $S \subsetneq V$ avec $\{j \in V : \exists (i,j) \in L_i,\ i \in \mathcal{U}\} \subset S$, au moins $\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil$ anneaux-étoiles doivent entrer dans le sous-ensemble S. Nous obtenons donc une adaptation des contraintes de capacité du CVRP :

$$\sum_{r\in\Omega} \sum_{e\in\delta(S)} b_e^r \theta_r \geq 2 \left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil . \tag{6.12}$$

Les inégalités précédentes peuvent cependant être renforcées [9] de la manière suivante. Soit $\Omega_{(\mathcal{U})}$ l'ensemble des anneaux-étoiles desservant au moins un client de $\mathcal{U}$: $\Omega(\mathcal{U}) = \{r \in \Omega : \sum_{i\in\mathcal{U}} \sum_{(i,j)\in A} a_{ij}^r \geq 1\}$. $\left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil$ étant une borne inférieure sur le nombre d'anneaux-étoiles nécessaires pour desservir les clients de $\mathcal{U}$, nous pouvons exprimer ces contraintes de capacité d'une manière plus directe :

$$\sum_{r\in\Omega(\mathcal{U})} \theta_r \geq \left\lceil \frac{d(\mathcal{U})}{Q} \right\rceil . \tag{6.13}$$

Nous pouvons voir facilement que si les inégalités (6.13) sont satisfaites, alors toutes les contraintes (6.12) le sont aussi. Conséquemment nous avons le lemme suivant :

Lemme 6.3.1 *Les contraintes de capacité (6.13) dominent les contraintes (6.12).*

Pour illustrer ce lemme, considérons la figure 6.2 sur laquelle on peut voir une solution fractionnaire avec 4 anneaux-étoiles r_1, r_2, r_3 et r_4. En posant $d_1 = d_4 = 3$,

$d_2 = d_3 = 2$, $Q = 5$, $\theta_1 = \theta_2 = \theta_3 = 0.5$ et $\theta_4 = 1$, nous remarquons que toutes les contraintes (6.12) sont satisfaites alors que la contrainte (6.13) définie sur les clients 1, 2 et 3 est violée : $\theta_1 + \theta_2 + \theta_3 \geq 2$. De plus, la remarque reste valable même si l'on ne considère que les tournées de $\Omega(\{1,2,3\})$. Par exemple, la contrainte (6.12) définie sur la coupe tracée en pointillée noire sera vérifiée alors que la contrainte (6.13) définie sur les clients 1, 2 et 3 est violée.

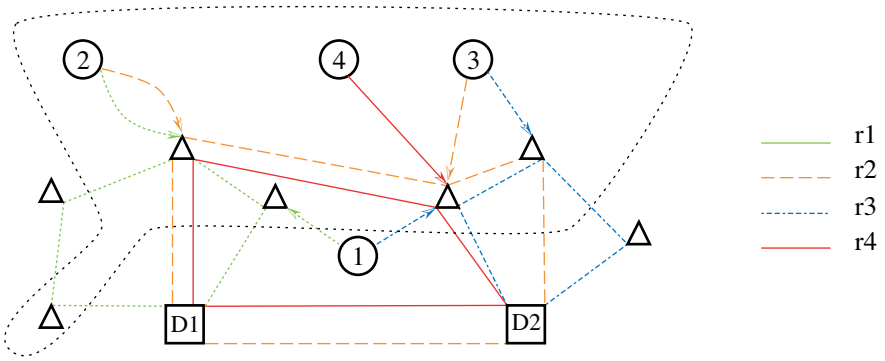


FIG. 6.2 – *Exemple de dominance des contraintes (6.12) sur les contraintes (6.13)*

6.3.2 Contraintes issues du polytope du Stable

6.3.2.1 Inégalités classiques

Remarquons tout d'abord que dans toute solution valide, les anneaux-étoiles sélectionnés desservent des sous-ensembles disjoints de clients. Partant de cette observation nous considérons le graphe $\mathcal{H} = (\Omega, T)$, appelé *graphe de conflit*, où Ω est l'ensemble de sommets en correspondance avec l'ensemble d'anneaux-étoiles et où il existe une arête entre les sommets r et r' si et seulement si $U(r) \cap U(r') \neq \emptyset$. Toute solution valide de $(MP(\Omega))$ correspondra alors à un stable dans $\mathcal{H}$. Il s'ensuit que toutes les inégalités valides du polytope du Stable seront valides pour notre problème. Nous nous limiterons cependant aux contraintes de clique et aux contraintes de cycle impair qui ont déjà montré leur efficacité.

Soit $\mathcal{K}(\mathcal{H})$ l'ensemble des cliques de $\mathcal{H}$, toute solution de $(MP(\Omega))$ vérifiera les *inégalités de clique* suivantes :

$$\sum_{r \in K} \theta_r \leq 1, \quad \forall K \in \mathcal{K}(\mathcal{H}). \tag{6.14}$$

Soit $\mathcal{C}(\mathcal{H})$ l'ensemble de tous les sous-ensembles de sommets induisant un cycle-impair dans $\mathcal{H}$, toute solution de $(MP(\Omega))$ vérifiera les *inégalités de cycle impair* suivantes :

$$\sum_{r\in C}\theta_r \leq \frac{|C|-1}{2}, \quad \forall C \in \mathcal{C}(\mathcal{H}). \tag{6.15}$$

6.3.2.2 Opération de réduction et renforcement des contraintes

Une remarque importante dans le cas du k-γ-RSP est que l'ensemble de clients desservis ne décrit pas totalement un anneau-étoile. Puisque de très nombreux anneaux-étoiles peuvent correspondre à un sous-ensemble de clients donné, $\mathcal{H}$ contiendra beaucoup de cliques particulières, chacune d'entre-elles étant induite par un sous-ensemble de clients.

Pour tirer partie de cette remarque, nous considérons une opération de réduction $\mathcal{T}$ sur le graphe $\mathcal{H}$ qui peut se définir de la façon suivante : étant donné un sous-ensemble de clients $\mathcal{U}$, l'opération $\mathcal{T}$ consiste à identifier tous les sommets anneaux-étoiles de $\mathcal{H}$ desservant exactement $\mathcal{U}$. On appelle $r_{\mathcal{U}}$ le sommet résultant de cette identification. Réciproquement, nous noterons $\mathcal{T}^{-1}(r_{\mathcal{U}})$ l'ensemble des sommets de $\mathcal{H}$ correspondant à des anneaux-étoiles desservant exactement l'ensemble de clients $\mathcal{U}$ et, par extension, pour un sous-ensemble de sommets $\mathcal{R} \in \Omega'$, $\mathcal{T}^{-1}(\mathcal{R})$ sera l'ensemble des sommets de $\mathcal{H}$ correspondant à des anneaux-étoiles desservant exactement l'un des ensembles de clients $\mathcal{U}$ tel que $r_{\mathcal{U}}$ appartienne à $\mathcal{R}$ *i.e.*

$$\mathcal{T}^{-1}(\mathcal{R}) = \bigcup_{r_{\mathcal{U}}\in\mathcal{R}} \mathcal{T}^{-1}(r_{\mathcal{U}}).$$

Soit $\mathcal{H}' = (\Omega', T')$ le graphe de conflit obtenu en utilisant l'opération de réduction $\mathcal{T}$ sur $\mathcal{H}$ pour tout sous-ensemble de clients $\mathcal{U} \subset U$. Considérons le polytope du Stable $P_S(\mathcal{H})$ (*resp.* $P_S(\mathcal{H}')$) associé au graphe $\mathcal{H}$ (*resp.* $\mathcal{H}'$). Remarquons que les variables θ correspondant à une solution de $MP(\Omega)$ sont des solutions de $P_S(\mathcal{H})$. On a alors le lemme suivant :

Lemme 6.3.2 *Soit θ une solution de $MP(\Omega)$, posons le vecteur θ' tel que $\theta'_{r_{\mathcal{U}}} = \sum_{r\in\mathcal{T}(r_{\mathcal{U}})}\theta_r$ pour tout $r_{\mathcal{U}} \in \mathcal{H}'$. Alors, comme $\mathcal{T}^{-1}(r_{\mathcal{U}})$ est une clique, θ' est une solution de $P_S(\mathcal{H}')$.*

On considérera par la suite une solution θ de $MP(\Omega)$ et le vecteur θ' donné par le lemme précédent. Soient $\mathcal{C}(\mathcal{H}')$ l'ensemble des sous-ensembles de sommets induisant

un cycle impair de $\mathcal{H}'$ et $\mathcal{K}(\mathcal{H}')$ l'ensemble des cliques de $\mathcal{H}'$.

Théorème 6.3.3 *Il existe une clique K dans $\mathcal{H}$ définissant une inégalité de clique violée par θ si et seulement s'il existe une clique K' dans $\mathcal{H}'$ définissant une inégalité de clique violée par θ'.*

Preuve. Soit K' une clique de $\mathcal{H}'$ telle que la contrainte de clique associée soit violée par θ', alors par construction la contrainte suivante sera violée par θ

$$\sum_{r_{\mathcal{U}} \in K'} \sum_{r \in \mathcal{T}^{-1}(r_{\mathcal{U}})} \theta_r \leq 1.$$

On remarque que cette inégalité correspond à une inégalité de clique dans $\mathcal{H}$.

Inversement, soit une clique K de $\mathcal{H}$, posons alors $\mathcal{W}$ comme la famille des sous-ensembles de clients desservis par l'un des sommets de K *i.e.*

$$\mathcal{W} = \{U(r) : \exists r \in K\}.$$

En appliquant l'opération $\mathcal{T}$ à tous les ensembles de sommets $\mathcal{U} \in \mathcal{W}$, nous obtenons un ensemble K' qui est une clique de $\mathcal{H}'$ correspondant par construction à une contrainte de clique violée par θ' dans $\mathcal{H}'$. □

On peut déduire de la preuve précédente le corollaire suivant.

Corollaire 6.3.4 *Soit K' une clique de $\mathcal{H}'$, alors toute clique K de $\mathcal{H}$ telle que $K \subseteq \mathcal{T}^{-1}(K')$ induit une contrainte de clique dominée par l'inégalité de clique*

$$\sum_{r_{\mathcal{U}} \in K'} \sum_{r \in \mathcal{T}^{-1}(r_{\mathcal{U}})} \theta_r \leq 1.$$

Tournons notre attention sur les inégalités de cycle impair définies sur $\mathcal{H}'$. De la même manière que précédemment, nous avons par construction le théorème suivant.

Théorème 6.3.5 *Soit $C' \in \mathcal{C}(\mathcal{H}')$, l'inégalité de cycle impair associée à C' sera violée par θ' dans $\mathcal{H}'$ si et seulement si l'inégalité*

$$\sum_{r_{\mathcal{U}} \in C'} \sum_{r \in \mathcal{T}^{-1}(r_{\mathcal{U}})} \theta_r \leq \frac{|C'|-1}{2} \tag{6.16}$$

est violée par θ dans $\mathcal{H}$. Nous appellerons les contraintes (6.16) des inégalités de clique-cycle.

On peut déduire de la preuve précédente le corollaire suivant.

Corollaire 6.3.6 *Les inégalités de clique-cycles (6.16) sont valides pour* $(MP(\Omega))$ *et dominent les inégalités de cycle impair.*

Conséquemment, pour séparer efficacement les contraintes de clique et les contraintes de clique-cycle dans $P_S(\mathcal{H})$ on peut se limiter à la séparation des contraintes de clique et de cycle impair dans $P_S(\mathcal{H}')$. Nous détaillerons les algorithmes de séparation dans le chapitre suivant.

6.4 Conclusion

Dans ce chapitre, nous avons étudié la version multi-dépôts du k-γ-RSP. Nous avons proposé pour ce problème une formulation à nombre exponentiel de variables. Nous avons étudié le problème de génération de colonnes associé à cette formulation et avons présenté pour le résoudre une formulation linéaire en nombres entiers inspirée de celle proposée pour le cas mono-dépôt dans le chapitre 5.

Nous avons également introduit plusieurs familles d'inégalités valides, issues de la littérature du CVRP et du problème du stable. Une étude de ces contraintes nous a permis d'obtenir plusieurs nouvelles familles de contraintes valides dominant les précédentes.

Dans le chapitre suivant, nous construisons un algorithme de Branch-and-Cut-and-Price sur la base de ces résultats.

Chapitre 7

Algorithme de Branch-and-Cut-and-Price pour le k-γ-RSP

7.1 Algorithme Branch-and-Cut-and-Price

Nous détaillons maintenant l'algorithme de Branch-and-Cut-and-Price basé sur la formulation et les contraintes valides du chapitre 6. Ces contraintes additionnelles sont difficiles à utiliser à cause de l'obligation de prise en compte des coûts duaux supplémentaires induits dans le problème auxiliaire. Nous présentons ici une première tentative d'utilisation exhaustive de ces contraintes.

Nous présentons également les procédures de séparation utilisées lors de la résolution du problème maître ou du problème auxiliaire ainsi que la règle de branchement utilisée.

Dans ce qui suit nous noterons θ^* une solution optimale fractionnaire du problème maître.

7.1.1 Initialisation

Un algorithme de Branch-and-Price doit disposer d'une première solution réalisable *a priori*. Dans notre cas, le problème d'existence d'une telle solution est un problème NP-complet. Ne pouvant fournir une solution réalisable au problème, nous utilisons des

variables artificielles : pour chaque client $i \in U$, nous ajoutons une variable binaire s_i de coût $\sum_{e \in E} l_e + \sum_{(i,j) \in A} c_{ij}$ dans la contrainte d'affectation (6.1) du client i. Ainsi, nous avons l'assurance qu'une première solution réalisable ($s_i = 1, \forall i \in U$; $\theta_r = 0, \forall r \in \Omega$) sera disponible pour lancer la procédure de génération de colonnes dans un second temps. De plus, le coût très important de ces variables artificielles nous assure qu'elles seront écartées dès qu'une solution réalisable sera rencontrée.

7.1.2 Contraintes de branchement

La solution obtenue en résolvant la relaxation linéaire du programme ($MP(\Omega)$) n'est pas toujours entière. Il est donc nécessaire d'utiliser ensuite une phase de branchement. Cependant, une règle de branchement classique ne serait pas efficace dans notre cas. En effet, si le fait de fixer une variable à 1 réduit grandement l'espace de recherche puisque un certain nombre de clients s'en trouvent traités, le fait de fixer une variable à 0 interdit simplement l'utilisation d'un seul élément dans l'ensemble de taille très importante Ω.

Nous avons donc choisi d'utiliser la stratégie de Ryan et Foster [94]. S'il existe un couple de clients i et j tel que la somme des affectations communes soit fractionnaire, nous créons deux noeuds. C'est-à-dire si $\sum_{\substack{r \in \Omega \\ i \in U(r), j \in U(r)}} \theta_r^* \in]0,1[$.

Dans le premier noeud nous imposons que les deux clients soient desservis par deux anneaux-étoiles différents grâce à la contrainte $\sum_{\substack{r \in \Omega \\ i \in U(r), j \in U(r)}} \theta_r^* = 0$ que nous dénoterons *diff*(i,j). Dans le second noeud nous imposons que les deux clients soient desservis par le même anneau-étoile grâce à la contrainte $\sum_{\substack{r \in \Omega \\ i \in U(r), j \in U(r)}} \theta_r^* = 1$ que nous dénoterons *same*(i,j). Dans chacun des nouveaux noeuds, les colonnes qui ne respectent pas la nouvelle contrainte sont supprimées.

Il faut également s'assurer lors de la résolution du problème auxiliaire que les nouvelles colonnes générées respectent bien les décisions prises pour arriver jusqu'au noeud courant de l'arbre de branchement. Cela peut se faire très facilement grâce à l'introduction dans (AP) d'un nombre réduit de contraintes additionnelles. Étant donnée une décision de branchement *diff*(i,j), l'inégalité suivante invalidera les anneaux-étoiles contenant les deux clients :

$$\sum_{(i,k) \in A} y_{ik} + \sum_{(j,k) \in A} y_{jk} \leq 1 \tag{7.1}$$

De la même manière, pour une contrainte *same*(i,j) donnée, la contrainte suivante

invalidera les anneaux-étoiles contenant uniquement l'un ou l'autre des clients.

$$\sum_{(i,k)\in A} y_{ik} = \sum_{(j,k)\in A} y_{jk} \tag{7.2}$$

Notons que dans les deux cas, tous les anneaux-étoiles ne contenant ni le client i ni le client j restent parfaitement valides et qu'une seule contrainte par décision de branchement doit être ajoutée.

7.1.3 Séparation des contraintes additionnelles du Problème Maître

Nous présentons maintenant les algorithmes de séparation utilisés dans notre algorithme lors de la résolution du problème maître.

- *Contraintes de capacité arrondie*

Nous utilisons ici une séparation heuristique très proche de celle utilisée dans notre algorithme de Branch-and-Cut pour les contraintes de capacité (4.21). Nous générons de manière heuristique un sous-ensemble de clients $\mathcal{U}$ puis vérifions si la contraintes de capacité (6.13) associée est bien satisfaite, si ce n'est pas le cas une inégalité violée a été trouvée et est ajoutée au problème maître. L'algorithme de séparation est détaillé dans l'algorithme 4. Cette séparation est comme précédemment appelée deux fois, la différence entre les deux appels étant la procédure utilisée pour construire l'ensemble $\mathcal{U}$. Lors du premier appel, la procédure *completer_ 1($\mathcal{U}$)* recherche le client $i' \in U \setminus \mathcal{U}$ minimisant la capacité résiduelle des anneaux-étoiles nécessaires pour desservir la demande de $\mathcal{U} \cup \{i'\}$:

$$i' = \arg\min_{i\in U\setminus\mathcal{U}} \left\lceil \frac{d(\mathcal{U}\cup\{i\})}{Q} \right\rceil - \frac{d(\mathcal{U}\cup\{i\})}{Q}$$

Pour le second appel, la procédure *completer_ 2($\mathcal{U}$)* recherche le client $i' \in U \setminus \mathcal{U}$ ayant le plus grand nombre de connecteurs en commun avec les clients de $\mathcal{U}$:

$$i' = \arg\max_{i\in U\setminus\mathcal{U}} |L_i \cap \{\cup_{j\in\mathcal{U}} L_j\}|$$

Entrées : La solution fractionnaire θ^*.
Sorties : Une contrainte de capacité violée si possible.

pour chaque *client* $i \in U$ **faire**
  $\mathcal{U} = \{i\}$;
  répéter
    si $\sum_{r \in \Omega_{(\mathcal{U})}} \theta^*_r < \left\lceil \frac{d(\mathcal{U}}{Q} \right\rceil$ **alors**
      retourner $\mathcal{U}$;
    fin
  tant que *completer_ 1(* $\mathcal{U}$*) ou completer_ 2(* $\mathcal{U}$*)*;
fin

Algorithme 4 : Séparation heuristique des contraintes de capacité arrondie

- *Contraintes de clique*

La procédure de séparation utilisée ici est celle proposée par Nemhauser et Sigismondi [87] pour le problème du stable. Bien qu'heuristique, cette séparation ne génère que des contraintes non dominées par d'autres contraintes de clique.

Considérons le graphe de conflit réduit $\mathcal{H}'$. Nous munissons chaque sommet $r_\mathcal{U} \in \Omega'$ d'un poids $w_{r_\mathcal{U}} = \theta'^*_{r_\mathcal{U}} = \sum_{r \in \mathcal{T}(r_\mathcal{U})} \theta^*_r$. Soit $\Omega_0 \subset \Omega'$ l'ensemble des sommets de $\mathcal{H}'$ ayant un poids strictement inférieur à 1. Nous trions par poids décroissant les sommets de Ω_0 puis ajoutons itérativement les sommets de Ω_0 intersectant tous les sommets de la clique en cours de construction. Soit $\mathcal{C}$ la clique ainsi obtenue, si $\sum_{r_\mathcal{U} \in \mathcal{C}} w_{r_\mathcal{U}} > 1$ alors une nouvelle contrainte de clique a été trouvée.

- *Contraintes de clique-cycle*

La séparation des inégalités de cycle impair dans un graphe $\mathcal{H}' = (\Omega', T')$ peut être effectuée en $O(|\Omega'|^3)$ [56]. Les inégalités les plus intéressantes sont celles définies sur des trous impairs [14]. En modifiant légèrement l'algorithme cité dans [56], la séparation des inégalités de trou impair peut être également réalisée en $O(|\Omega'|^3)$.

L'idée principale de l'algorithme développé dans [56] consiste à considérer un nouveau graphe $\mathcal{H}'' = (\Omega'', T'')$ avec deux sommets $r_\mathcal{U}$ et $r'_\mathcal{U}$ pour tout sommet $r_\mathcal{U} \in \Omega'$. Pour toute arête $\{r_{\mathcal{U}_1}, r_{\mathcal{U}_2}\}$ de T', on définit dans T'' les arêtes $\{r_{\mathcal{U}_1}, r'_{\mathcal{U}_2}\}$ et $\{r'_{\mathcal{U}_1}, r_{\mathcal{U}_2}\}$ munies du poids $w_{r_{\mathcal{U}_1} r_{\mathcal{U}_2}} = 1 - \theta'^*_{r_{\mathcal{U}_1}} - \theta'^*_{r_{\mathcal{U}_2}}$. Il est clair que le graphe $\mathcal{H}''$ est biparti.

Maintenant, remarquons que la chaîne élémentaire $P''_{r_{\mathcal{U}_1} r'_{\mathcal{U}_2}}$ de $\mathcal{H}''$ allant d'un sommet $r_{\mathcal{U}_1}$ à un sommet $r'_{\mathcal{U}_2}$ correspond à une chaîne impaire $P'_{r_{\mathcal{U}_1} r_{\mathcal{U}_2}}$ dans H' entre $r_{\mathcal{U}_1}$ et $r_{\mathcal{U}_2}$. Pour tout sommet $r_{\mathcal{U}_1}$ de $\mathcal{H}'$, on calcule une plus courte chaîne $P''_{r_{\mathcal{U}_1} r'_{\mathcal{U}_1}}$ dans H''

entre $r_{\mathcal{U}_1}$ et $r'_{\mathcal{U}_1}$. La plus courte chaîne $P'_{r_{\mathcal{U}_1} r_{\mathcal{U}_1}}$ correspond à un cycle impair $C_{r_{\mathcal{U}_1}}$ de H' contenant $r_{\mathcal{U}_1}$ et sa longueur est exactement $|C_{r_{\mathcal{U}_1}}| - 2\sum_{r_{\mathcal{U}} \in C_{r_{\mathcal{U}_1}}} \theta'^*_{r_{\mathcal{U}}}$

Soit $C_{r_{\mathcal{U}_0}}$ le cycle de longueur minimale, si sa longueur est supérieure ou égale à 1, alors $\sum_{r_{\mathcal{U}} \in C} \theta'^*_{r_{\mathcal{U}}} \leq \frac{|C|-1}{2}$ pour tout cycle impair et ainsi les inégalités de cycle impair sur $\mathcal{H}'$ sont satisfaites par θ'^*, puis par extension les inégalités de clique-cycle (6.16) sur $\mathcal{H}$ sont toutes satisfaites par θ^*. Dans le cas contraire, la contrainte de clique-cycle associée à $C_{r_{\mathcal{U}_0}}$ est violée par θ^*.

De plus, si la chaîne $P'_{u'_0 u''_0}$ correspondant à C_{u_0} est choisie comme étant une chaîne de plus petite cardinalité parmi les plus courtes chaînes de G' entre u'_0 et u''_0, alors le cycle C_{u_0} est un trou impair.

7.1.4 Gestion des coûts duaux dans le problème auxiliaire

L'introduction d'inégalités supplémentaires dans le problème maître perturbe la procédure de génération de colonnes. En effet, les nouveaux coûts duaux induits par ces contraintes doivent être pris en compte lors de l'évaluation du coût réduit de chaque anneau-étoile. Les coûts duaux associés aux contraintes que nous avons introduites ne peuvent être directement reportés sur les coûts des arcs et arêtes du graphe support. C'est d'ailleurs la principale raison de leur très faible utilisation dans la littérature tournées de véhicules. Nous détaillons ici comment les prendre en compte.

7.1.4.1 Contraintes de capacité

Soit $\sigma_{\mathcal{U}}$ la variable duale associée à la contrainte de capacité (6.12) définie sur le sous-ensemble de client $\mathcal{U} \subset U$. Nous ajoutons au problème auxiliaire (AP) une variable continue $z_{\sigma_{\mathcal{U}}} \in [0,1]$ et fixons son coût dans la fonction objectif à $-\sigma_{\mathcal{U}}$. Nous ajoutons également les contraintes suivantes :

$$z_{\sigma_{\mathcal{U}}} \geq \sum_{(i,j) \in A} y_{ij}, \qquad \forall i \in \mathcal{U} \tag{7.3}$$

$$z_{\sigma_{\mathcal{U}}} \leq \sum_{i \in \mathcal{U}} \sum_{(i,j) \in A} y_{ij}. \tag{7.4}$$

Les inégalités (7.3) forceront la variable $z_{\sigma_{\mathcal{U}}}$ à prendre la valeur 1 si au moins l'un des clients du sous-ensemble $\mathcal{U}$ est desservi par l'anneau-étoile construit. À l'inverse, l'inégalité (7.4) impose que la variable $z_{\sigma_{\mathcal{U}}}$ soit égale à 0 si aucun client du sous-ensemble $\mathcal{U}$ n'est desservi.

7.1.4.2 Contraintes de clique

Soit μ_K la variable duale associée à la contrainte de clique (6.14) définie sur la clique K. Nous ajoutons au problème auxiliaire (AP) une variable continue $z_K \in$ [0,1]et fixons son coût dans la fonction objectif à $-\mu_K$.

Soit un sous-ensemble de clients $\mathcal{U} \subset U$ intersectant tous les sommets de la clique : $\mathcal{U} \cap U(r) \neq \emptyset$, $\forall r \in K$. Tout anneau-étoile desservant les clients de $\mathcal{U}$ fera partie de la clique. L'inégalité suivante imposera alors que la variable z_K soit égale à 1 si tous les clients de $\mathcal{U}$ sont desservis :

$$\sum_{i\in\mathcal{U}} \sum_{(i,j)\in A} y_{ij} - z_K \leq |\mathcal{U}| - 1. \tag{7.5}$$

Inversement, soit un sous-ensemble de clients $\mathcal{U} \subset U$ n'ayant aucun client commun avec l'un des sommets de la clique : $\exists r \in K : \mathcal{U} \cap U(r) = \emptyset$. Tout anneau-étoile desservant uniquement les clients de $\mathcal{U}$ n'appartiendra pas à la clique K. La contrainte suivante imposera que la variable z_K soit égale à 0 si aucun autre client n'est desservi par l'anneau-étoile construit :

$$z_K + \sum_{i\in\mathcal{U}} \sum_{(i,j)\in A} y_{ij} - \sum_{i\notin\mathcal{U}} \sum_{(i,j)\in A} y_{ij} \leq |\mathcal{U}|. \tag{7.6}$$

7.1.4.3 Inégalité de clique-cycle

Soit μ_C la variable duale associée à la contrainte de clique-cycle (6.16) définie sur le cycle C. Nous ajoutons une variable continue z_C au problème auxiliaire (AP) et fixons son coût dans la fonction objectif à $-\mu_C$.

Soit un sous-ensemble de clients $\mathcal{U} \subset U$ appartenant au clique-cycle : $\exists r \in C : \mathcal{U} = U(r)$. Un anneau-étoile ne desservant que les clients de $\mathcal{U}$ fera partie du clique-cycle. La contrainte suivante imposera alors que la variable z_C prenne la valeur 1 si aucun autre client n'est desservi par l'anneau-étoile construit :

$$\sum_{i\in\mathcal{U}} \sum_{(i,j)\in A} y_{ij} - z_K - \sum_{i\in\mathcal{U}} \sum_{(i,j)\notin A} y_{ij} \leq |\mathcal{U}| - 1. \tag{7.7}$$

À l'inverse, soit un sous-ensemble de clients $\mathcal{U} \subset U$ n'appartenant pas au clique-cycle : $\mathcal{U} \neq U(r)$, $\forall r \in C$. Un anneau-étoile ne desservant que les clients de $\mathcal{U}$ ne fera pas partie du clique-cycle. La contrainte suivante imposera alors que la variable z_C prenne la valeur 0 si aucun autre client n'est desservi :

$$z_C + \sum_{i\in\mathcal{U}} \sum_{(i,j)\in A} y_{ij} - \sum_{i\notin\mathcal{U}} \sum_{(i,j)\in A} y_{ij} \leq |\mathcal{U}|. \tag{7.8}$$

7.1.4.4 Gestion des contraintes additionnelles dans le problème auxiliaire

Nous avons déterminé un ensemble de contraintes additionnelles (7.3)-(7.8) pour le problème auxiliaire qui assurent la prise en compte des coût duaux additionnels. Cependant, si les inégalités (7.3), (7.4) et (7.7) sont en nombre polynomial, ce n'est pas le cas des inégalités (7.5), (7.6) et (7.8). De plus, la complexité du problème de séparation associé à ces dernières contraintes est une question ouverte à ce jour. Il est néanmoins possible d'utiliser ces contraintes de manière exacte : remarquons tout d'abord que lorsque l'ensemble des clients $\mathcal{U}$ desservi par l'anneau-étoile est connu, leur séparation devient polynomiale puisqu'il suffit de vérifier la satisfaction des inégalités définies sur $\mathcal{U}$. Or, il est possible d'obtenir la satisfaction de cette condition très tôt dans l'arbre de branchement si une règle de branchement particulière est utilisée. Nous utilisons ici une règle de branchement très proche de celle utilisée pour notre algorithme de Branch-and-Cut. Nous branchons en priorité sur les contraintes d'affectations :

$$\sum_{(i,j)\in L_i} y_{ij} = a,\ a \in \{0,1\}.$$

Si toutes les contraintes d'affectations sont entières, nous branchons ensuite sur les variables y_{ij} puis enfin sur les variables x_e.

En pratique, nous introduisons directement dans la formulation les inégalités en nombre polynomial : (7.3), (7.4) et (7.7). Nous utilisons ensuite une procédure de séparation pour les inégalités (7.5), (7.6) et (7.8) qui commence par vérifier si les affectations sont connues, c'est à dire si $\sum_{(i,j)\in A} y_{ij} \in \{0,1\}$, $\forall i \in U$.

Si c'est le cas, soit $\mathcal{U} = \{i \in U : \sum_{(i,j)\in A} y_{ij} = 1\}$ et $\bar{\mathcal{U}} = U \setminus \mathcal{U}$. Nous vérifions si les contraintes définies sur les ensembles de client $\mathcal{U}$ et $\bar{\mathcal{U}}$ sont vérifiées. Si une contrainte violée est trouvée, nous l'ajoutons au programme linéaire (AP), sinon nous pouvons affirmer qu'elles sont toutes vérifiées dans le noeud courant et tous ses descendants.

7.1.5 Algorithme de Branch-and-Cut pour le problème auxiliaire

L'algorithme utilisé est très proche de celui présenté en section 5.2 pour le cas mono-dépôt :

- L'algorithme commence par résoudre la relaxation linéaire de la formulation limitée aux contraintes (6.6)-(6.9), contraintes triviales et contraintes additionnelles (7.3), (7.4) et (7.7).

TAB. 7.1 – *Comparaison entre le Branch-and-Cut et le Branch-and-Cut-and-Price*

Instances				Branch-and-Cut					Branch-and-Cut-and-Price					
$\|N\|$	$\|U\|$	$\|W\|$	m	$\#Opt$	FG	$RGAC$	$\#N$	TT	$\#Opt$	FG	$RGAC$	RG	$\#N$	TT
1	10	10	3	10	0,00%	8,10%	90,6	8,46	10	0,00%	0,00%	4,05%	1	3,47
1	10	10	4	10	0,00%	6,99%	456,1	24,807	10	0,00%	0,06%	5,62%	2,56	2,49
1	10	10	5	10	0,00%	5,60%	1289,8	62,753	10	0,00%	0,00%	0,04%	1	1,10
1	10	20	3	10	0,00%	5,34%	40,5	18,216	10	0,00%	0,34%	2,86%	1,8	9,36
1	10	20	4	10	0,00%	8,41%	672,2	105,383	10	0,00%	0,00%	4,40%	1	9,13
1	10	20	5	10	0,00%	6,06%	2463,3	275,216	10	0,00%	0,00%	0,43%	1	6,24
1	10	40	3	10	0,00%	5,65%	42,6	188,35	10	0,00%	0,00%	3,60%	1,2	99,06
1	10	40	4	10	0,00%	7,61%	359,8	269,057	10	0,00%	0,26%	3,35%	1,7	347,10
1	10	40	5	10	0,00%	6,12%	2305,9	1032,583	10	0,00%	0,00%	0,45%	1	104,93
1	15	10	3	10	0,00%	7,73%	183,2	38,113	10	0,00%	0,09%	2,01%	1,7	12,73
1	15	10	4	10	0,00%	9,40%	4097,7	520,201	10	0,00%	0,77%	1,62%	21,5	26,38
1	15	10	5	4	6,60%	15,97%	21971,2	5358,002	10	0,00%	0,28%	1,82%	5,3	20,64
1	15	20	3	10	0,00%	8,02%	188,1	116,023	10	0,00%	0,12%	0,27%	2,8	41,87
1	15	20	4	10	0,00%	10,36%	7104,5	2029,519	10	0,00%	0,03%	1,10%	2,2	68,49
1	15	20	5	3	7,43%	15,72%	12510,4	5884,216	10	0,00%	0,45%	0,91%	4,9	32,40
1	15	40	3	10	0,00%	8,47%	397,7	591,847	10	0,00%	0,04%	0,63%	1,3	255,90
1	15	40	4	8	0,77%	9,01%	3780,7	3431,673	10	0,00%	0,77%	1,97%	35,6	2176,82
1	15	40	5	1	12,88%	18,15%	4663,5	6766,792	10	0,00%	1,06%	2,07%	9	1278,62

- Nous utilisons ensuite un algorithme de séparation exacte pour les contraintes de 2-connexité généralisées (6.10).
- Si la solution obtenue à la fin de cette phase de coupes n'est pas entière nous séparons les inégalités de connexité généralisée (5.3). Notons que les contraintes de capacité n'ont pas d'intérêt ici puisque nous ne recherchons qu'un seul anneau-étoile.
- Si la solution est toujours fractionnaire, nous utilisons ensuite un arbre de branchement dans lequel les inégalités additionnelles ne sont plus séparées mais dans lequel nous séparons dès que possible les contraintes (7.5), (7.6) et (7.8).

7.2 Résultats expérimentaux

L'algorithme de Branch-and-Cut-and-Price présenté dans les sections précédentes a été implémenté en C++ grâce à l'environnement SCIP 3.0 couplé au solveur Cplex

TAB. 7.2 – *Détail sur la densité des instances aléatoires pour* $|N| = 2$

$\|N\|$	$\|U\|$	$\|W\|$	m	$\|E\|$	$min(\delta(i))$	$max(\delta(i))$	$\|A\|$	$min(\|L_i\|)$	$max(\|L_i\|)$
2	10	10	3	53,2	4,9	9	52,7	3,5	7,2
2	10	10	4	52,2	5,1	9,3	52	3,8	7,1
2	10	10	5	52,8	4,9	9,1	52,8	3,7	6,8
2	10	20	3	80,7	5	9,6	58,1	4,4	7,1
2	10	20	4	80,7	4,9	9,4	59,5	4,5	7,3
2	10	20	5	80,7	4,9	9,1	58,7	4,3	7,5
2	10	40	3	135,1	4,9	9,8	61,9	4,6	7,9
2	10	40	4	136,3	5	9,6	62,6	4,9	7,9
2	10	40	5	136,6	5	10	60,7	4,7	7,5
2	15	10	3	65,6	5	9,3	81,5	3,7	7,6
2	15	10	4	66,9	4,8	9,4	83,9	3,8	7,3
2	15	10	5	65,9	4,9	9,3	82,9	4	7,5
2	15	20	3	93,3	4,9	9,9	85,8	3,7	8
2	15	20	4	93,9	4,9	9,8	88,5	3,9	7,9
2	15	20	5	92,1	5	9,5	85,6	4,1	7,9
2	15	40	3	146,7	5	10,1	89,9	3,9	7,9
2	15	40	4	150,1	5,1	10,2	94,1	4,1	8,3
2	15	40	5	148,8	4,8	10,4	89,7	4,1	8,3

12.4. Les expérimentations ont été menées sur un ordinateur de bureau équipé d'un processeur Intel2 Duo cadencé à 3.33 GHz et d'une mémoire vive de 8 Go.

Nous comparons dans un premier temps l'efficacité de notre algorithme de Branch-and-Cut-and-Price (noté *BCP* dans ce qui suit) avec l'algorithme de Branch-and-Cut (*BC*) présenté au chapitre 5 sur la classe d'instances aléatoires du chapitre 5. Si nous n'utilisons pas de contrainte additionnelle dans la formulation set partitionning, nous parlerons d'algorithme de Branch-and-Price (*BP*). Ensuite, nous étudions l'efficacité de notre algorithme BCP sur une classe d'instances multi-dépôts générées aléatoirement de la même manière que précédemment pour le cas mono-dépot.

Dans toutes les tables qui suivent, la première colonne donne les caractéristiques de l'instances : le nombre de dépôts $|N|$, le nombre de clients $|U|$, le nombre de sommets Steiner $|W|$ et le nombre minimal m d'anneaux-étoiles d'une solution réalisable. Les valeurs qui sont reportées sont les moyennes obtenues sur 10 instances générées aléatoirement.

TAB. 7.3 – *Détail sur la densité des instances aléatoires pour* $|N| = 3$

$\|N\|$	$\|U\|$	$\|W\|$	m	$\|E\|$	$min(\delta(i))$	$max(\delta(i))$	$\|A\|$	$min(\|L_i\|)$	$max(\|L_i\|)$
3	10	10	3	56,1	5,1	8,9	52,2	3,8	7
3	10	10	4	55,2	4,9	9,1	51,8	3,5	7
3	10	10	5	55,7	4,9	9	50,2	3,2	6,5
3	10	20	3	83,6	5,2	9,4	56,2	4	7,5
3	10	20	4	83,7	4,8	9,5	54,8	3,8	7,2
3	10	20	5	84,4	4,8	9,5	57,5	4	7,4
3	10	40	3	138,5	4,9	10,2	60,5	4,5	7,7
3	10	40	4	138,8	4,9	9,8	62,5	4,6	7,9
3	10	40	5	138,2	4,9	10	60,1	4,6	7,7
3	15	10	3	68,7	4,8	9,3	79,4	3,6	7,3
3	15	10	4	68,6	4,9	9,4	80,2	3,6	7,3
3	15	10	5	69,5	4,9	9	80,6	3,5	7,5
3	15	20	3	94,1	5	9,5	83,6	3,9	7,8
3	15	20	4	96,5	4,9	9,6	86,1	3,9	7,5
3	15	20	5	96,5	5,1	9,5	85,9	4	7,5
3	15	40	3	150	5	9,7	88,6	4,1	7,9
3	15	40	4	151,1	4,8	10	93,9	4,3	8,1
3	15	40	5	152,2	4,9	10	90,6	4	8,3

7.2.1 Comparaison des approches pour le 1-γ-RSP

La table 7.1 compare de manière synthétique le comportement de l'algorithme de Branch-and-Cut-and-Price avec l'algorithme du chapitre 5 pour les instances à un seul dépôt en indiquant les résultats suivants :

- #Opt : le nombre d'instances résolues à l'optimalité.
- FG : l'écart relatif entre la meilleure solution trouvée et la meilleure borne inférieure fournie.
- GRAC : l'écart relatif entre la meilleure solution trouvée et la meilleure borne inférieure fournie par la relaxation linéaire de la formulation avec les coupes additionnelles.
- GR : l'écart relatif entre la meilleure solution trouvée et la meilleure borne inférieure fournie par la relaxation linéaire de la formulation sans coupe additionnelle.
- #N : le nombre de variables générées.
- TT : le temps total de calcul en secondes (limité à deux heures).

Il apparaît nettement que BCP bat largement BC à la fois sur la taille des instances résolues et le temps de calcul. Même en comparant la qualité de relaxation de BC avec celle de BP obtenue en ne considérant que la formulation sans contrainte addi-

TAB. 7.4 – *Détail sur la densité des instances aléatoires pour* $|N| = 4$

$\|N\|$	$\|U\|$	$\|W\|$	m	$\|E\|$	$min(\delta(i))$	$max(\delta(i))$	$\|A\|$	$min(\|L_i\|)$	$max(\|L_i\|)$
4	10	10	3	57,9	5,1	9,4	48,1	3,3	6,6
4	10	10	4	57	4,8	8,9	49,8	3,3	6,6
4	10	10	5	58,1	4,9	9,1	49,4	3,1	6,5
4	10	20	3	85,7	5,1	9,5	55,7	4,1	7,5
4	10	20	4	86,7	5	9,6	55,4	3,9	7,3
4	10	20	5	84,2	5,1	9,2	53,7	3,4	6,8
4	10	40	3	140,9	4,9	9,9	60,7	4,3	7,7
4	10	40	4	141,1	4,9	9,9	59,6	4,3	7,4
4	10	40	5	141,7	5	9,6	61,2	4,6	7,8
4	15	10	3	71,7	5	9,7	77	3,2	7,3
4	15	10	4	70,9	4,9	9,3	76,8	2,9	7,3
4	15	10	5	71,1	4,8	9,3	77,1	3,3	7,4
4	15	20	3	99	4,8	9,3	83,1	3,6	7,4
4	15	20	4	99,7	5	9,5	83,2	3,5	7,7
4	15	20	5	98,6	5	9,6	82,5	3,7	7,5
4	15	40	3	154,5	5,1	10,2	93,1	4,3	8
4	15	40	4	154	5	10,3	90,8	4,2	7,9
4	15	40	5	153,7	4,8	10,1	88,5	4	8,1
4	20	40	5	166,7	4,7	9,7	117,9	3,8	8,3
4	20	40	7	166,2	4,9	10	119,6	3,7	8,1
4	25	40	5	182,7	4,9	10,4	150,5	3,6	8,3
4	25	40	7	179,9	4,9	10,4	147,6	3,6	8,3

tionnelle, BP est largement meilleur. Cela semble confirmer que les limitations de BC proviennent de la symétrie induite par l'obligation d'utiliser une formulation à 3 indices et donc qu'une approche par génération de colonnes est un moyen efficace d'éviter cette symétrie. Une piste d'amélioration de BC serait d'utiliser le concept de branchement orbitopal récemment mis en évidence dans la littérature [88, 68].

On peut cependant constater que BC fonctionne très bien pour la recherche d'un anneau-étoile unique puisqu'il se retrouve au coeur de l'algorithme de BCP qui fonctionne efficacement. En effet, dans ce cas, il n'y plus de symétrie dans la formulation.

TAB. 7.5 – *Impact des contraintes additionnelles pour* $|N| = 1$

Instances				Problème Maître							Problème Auxiliaire				
$\|N\|$	$\|U\|$	$\|W\|$	m	GR	Int	$GRAC$	$IntAC$	$\#RC$	$\#CC$	$\#OC$	$\#Calls$	$\#AR$	$\#AC$	$\#AO$	$\#AB$
1	10	10	3	4,05%	1	0,00%	10	2,1	1,7	1,7	29,3	92,4	44,2	5,6	0
1	10	10	4	5,62%	0	0,06%	9	1,3	0,9	1	25,8	103,5	16,9	7,7	6,2
1	10	10	5	0,04%	9	0,00%	10	0,1	0,1	0,2	12,3	0,8	0,1	0,1	0
1	10	20	3	2,86%	3	0,34%	9	0,9	0,9	0,7	27,4	89	38,3	21,5	7,3
1	10	20	4	4,40%	0	0,00%	10	1,7	1,3	1,5	22,6	70,1	11,1	12,5	0
1	10	20	5	0,43%	7	0,00%	10	0,3	0,4	0,8	14	2,4	1	1,4	0
1	10	40	3	3,60%	1	0,00%	9	1,6	1,6	1,8	32,8	99,2	85	13,4	0,2
1	10	40	4	3,35%	1	0,26%	8	1,4	1	1,2	23,7	79	33,8	12,3	3%,5
1	10	40	5	0,45%	6	0,00%	10	0,5	0,5	0,8	15,5	8,2	3,9	3,9	0
1	15	10	3	2,01%	7	0,09%	9	0,2	1,7	1,5	42	82,8	433,1	9,7	12,2
1	15	10	4	1,62%	2	0,77%	6	1,1	0,8	1,6	77,9	650,4	112,8	61,3	234,9
1	15	10	5	1,82%	3	0,28%	3	0,7	1,6	3,4	39,5	91,7	39,7	18,7	24,6
1	15	20	3	0,27%	8	0,12%	9	0,1	0,4	0,8	45,3	1,4	241,1	4,4	20,5
1	15	20	4	1,10%	3	0,03%	8	1,7	1,9	3,1	46,2	89,2	133,6	15,4	12,7
1	15	20	5	0,91%	5	0,45%	5	0,1	0,9	1	35,7	15,6	109,9	100,5	25,2
1	15	40	3	0,63%	8	0,04%	9	0,2	0,4	0,5	40,8	4,8	193,3	30,2	4,1
1	15	40	4	1,97%	0	0,77%	5	1,3	2,5	2,7	119,1	300	776,2	112,5	482,2
1	15	40	5	2,07%	1	1,06%	4	0,2	2,9	3,7	52,9	11,8	205,2	88,2	85,6

7.2.2 Résultats expérimentaux pour le k-γ-RSP

- *Description des instances*

Les tables 7.2 à 7.4 donnent plus d'informations sur la densité des instances aléatoires multi-dépôts. Nous y reportons le nombre d'arêtes $|E|$ et le nombre d'arcs $|A|$. Nous y indiquons également le nombre minimal (*resp.* maximal) d'arêtes incidentes à un sommet $min(|\delta(i)|)$ (*resp.* $max(|\delta(i)|)$). Enfin, la taille minimale (*resp.* maximale) d'un ensemble d'affectation est donné par $min(|L_i|)$ (*resp.* $max(|L_i|)$), ce chiffre correspond au nombre minimal (*resp.* maximal) d'arcs sortant d'un client $i \in U$.

- *Impact des contraintes additionnelles*

Les tables 7.5 à 7.8 mettent en évidence l'impact des contraintes additionnelles que nous séparons lors de la résolution du problème maître en donnant les résultats suivants :

TAB. 7.6 – *Impact des contraintes additionnelles pour* $|N| = 2$

Instances				Problème Maître							Problème Auxiliaire				
$\|N\|$	$\|U\|$	$\|W\|$	m	*GR*	*Int*	*GRAC*	*IntAC*	*#RC*	*#CC*	*#OC*	*#Calls*	*#AR*	*#AC*	*#AO*	*#AB*
2	10	10	3	2,92%	2	0,17%	6	1	1,1	1,2	39,2	127,6	49,1	22	9,4
2	10	10	4	4,18%	1	0,07%	9	1,8	1,4	1,5	36,2	113,8	19,2	3,4	2,8
2	10	10	5	1,18%	7	0,22%	9	0,5	0,3	0,7	23,6	36,2	5,7	6,7	8,6
2	10	20	3	2,32%	2	0,38%	6	1,2	1	1,3	46,6	156,8	47,6	13	14,2
2	10	20	4	3,09%	0	0,00%	9	1,3	0,9	0,9	39	139,6	20,8	2,7	3,4
2	10	20	5	1,60%	4	0,01%	9	0,7	0,7	1,5	26,4	78	16	17,3	10
2	10	40	3	2,49%	1	0,37%	7	1,8	1,7	1,5	48,6	163,2	112,7	20,8	6,2
2	10	40	4	3,31%	2	0,63%	7	1	0,8	0,7	38	134,4	34,5	38,2	11,2
2	10	40	5	0,59%	6	0,00%	10	0,7	0,6	0,8	22,8	32	8,2	7	0
2	15	10	3	0,53%	7	0,21%	9	0,1	0,4	0,1	71,4	4,8	110,6	4,1	36,6
2	15	10	4	1,01%	5	0,07%	8	1,2	1,5	1,7	59,8	221,2	118,1	46	21,8
2	15	10	5	1,58%	4	0,91%	5	0,2	1,6	2,4	80	61,8	126,3	38,2	133,4
2	15	20	3	1,65%	3	0,11%	8	0,4	2,1	1,3	67,8	79,2	368,7	9	2,6
2	15	20	4	1,77%	2	0,75%	5	1,2	1	2,9	105,8	717,6	141,8	59,4	212
2	15	20	5	1,56%	2	0,01%	9	0,5	2,3	2,2	51,8	31,2	93,4	25,9	0,8
2	15	40	3	0,87%	5	0,34%	7	0,3	1,1	0,8	70,4	88,2	233	10,1	35,2
2	15	40	4	1,71%	1	0,04%	9	1,1	2,1	2,6	68,6	253,2	151,8	43,1	15,6
2	15	40	5	2,19%	1	0,94%	5	0,7	2,9	3,9	91,2	129,6	290	35	156,6

GR : l'écart relatif entre la meilleure solution trouvée et la meilleure borne inférieure fournie par la relaxation linéaire de la formulation sans coupe additionnelle.

Int : le nombre d'instances pour lesquelles la relaxation linéaire sans coupe additionnelle fournit une solution entière.

GRAC : l'écart relatif entre la meilleure solution trouvée et la meilleure borne inférieure fournie par la relaxation linéaire de la formulation avec les coupes additionnelles.

IntAC : le nombre d'instances pour lesquelles la relaxation linéaire avec les coupes additionnelles fournit une solution entière.

#Calls : le nombre d'appel au problème auxiliaire.

#AR : le nombre de contraintes additionnelles nécessaires à la prise en compte des contraintes de capacité du problème maître.

#AC : le nombre de contraintes additionnelles nécessaires à la prise en compte des contraintes de clique du problème maître.

#AO : le nombre de contraintes additionnelles nécessaires à la prise en compte des contraintes de clique-cycle du problème maître.

#AB : le nombre de contraintes additionnelles nécessaires à la prise en compte des contraintes de branchement.

TAB. 7.7 – *Impact des contraintes additionnelles pour* $|N| = 3$

Instances				Problème Maître							Problème Auxiliaire				
$\|N\|$	$\|U\|$	$\|W\|$	m	GR	Int	$GRAC$	$IntAC$	$\#RC$	$\#CC$	$\#OC$	$\#Calls$	$\#AR$	$\#AC$	$\#AO$	$\#AB$
3	10	10	3	1,31%	5	0,00%	10	0,7	0,5	0,5	51,9	82,2	42,4	3,8	0
3	10	10	4	1,59%	4	0,00%	10	0,6	0,6	0,7	35,7	70,8	8,8	2	0
3	10	10	5	0,25%	9	0,00%	10	0,1	0,1	0,2	24	1,8	0,1	0,4	0
3	10	20	3	1,00%	6	0,04%	9	0,4	0,2	0,3	48	107,4	16,8	2,3	6
3	10	20	4	7,19%	3	0,03%	9	0,8	0,6	0,8	39,6	75,3	12,3	19	0,9
3	10	20	5	1,52%	5	0,00%	10	0,5	0,6	1,1	29,1	17,4	4	2,7	0
3	10	40	3	1,44%	3	0,41%	9	0,8	0,8	0,8	54,3	118,2	54	57,8	4,2
3	10	40	4	1,89%	3	0,00%	10	0,8	0,7	0,5	46,2	96,3	14,4	2,5	0
3	10	40	5	1,43%	6	0,00%	10	0,7	0,5	1	28,8	33	3,9	4,3	0
3	15	10	3	0,86%	5	0,17%	9	0,1	1,4	0,4	80,7	28,8	259,4	5,6	3,6
3	15	10	4	0,91%	6	0,64%	7	0,1	1,2	1,1	98,7	27	115,2	20,2	123
3	15	10	5	2,09%	1	1,10%	2	0,4	2,1	3,5	125,7	73,2	195	130,4	242,4
3	15	20	3	1,27%	3	0,28%	7	0,3	1,4	1,5	90,6	45,6	390,3	109,6	12,6
3	15	20	4	1,54%	5	0,74%	6	0,6	1,4	1,2	76,2	259,2	153,6	16,1	27
3	15	20	5	2,66%	2	1,85%	3	0,3	2	3,1	151,8	225,9	211,5	162,7	509,7
3	15	40	3	0,54%	8	0,00%	10	0	1,2	0,5	77,7	0	98,4	5,2	0
3	15	40	4	1,47%	2	0,53%	7	0,9	2,3	2,3	109,8	605,4	506,2	960	122,4
3	15	40	5	1,00%	3	0,16%	7	0,4	1,5	1,7	60,6	30,3	102,5	15,7	7,8

Nous pouvons remarquer que la relaxation de la formulation est généralement très bonne, le gap relatif entre la valeur de la relaxation linéaire et la meilleure solution connue est de 1,82% en moyenne (il y a cependant un pire cas à 16,87%). De plus, la relaxation linéaire fournie une solution entière optimale dans 38,68% des cas.

Il y a peu de contraintes séparées lors de la résolution du problème maître :
- au plus 7 contraintes de capacité et 2,1 en moyenne ;
- au plus 12 contraintes de clique et 2,9 en moyenne ;
- au plus 10 contraintes de clique-cycle et 3,9 en moyenne.

Bien que peu nombreuses, ces contraintes sont cependant très efficaces : le gap relatif entre la valeur de la relaxation linéaire et la meilleure solution connue passe de 1,82% à 0,40%. De plus nous obtenons maintenant une solution entière dès la racine de l'arbre de branchement dans 78,95% des cas. Autrement dit, les inégalités valides que nous ajoutons nous permettent de combler le gap d'intégrité pour 65,67% des instances pour lesquelles la relaxation linéaire ne donnait qu'une solution fractionnaire.

TAB. 7.8 – *Impact des contraintes additionnelles pour* $|N| = 4$

Instances				Problème Maître							Problème Auxiliaire				
$\|N\|$	$\|U\|$	$\|W\|$	m	*GR*	*Int*	*GRAC*	*IntAC*	*#RC*	*#CC*	*#OC*	*#Calls*	*#AR*	*#AC*	*#AO*	*#AB*
4	10	10	3	2,97%	3	0,36%	5	0,4	1,4	1,1	64	40,4	87,6	89,7	24,4
4	10	10	4	2,23%	1	0,00%	10	1	1	0,8	53,2	161,2	25,1	5,9	0
4	10	10	5	1,47%	6	0,00%	10	0,3	0,4	0,8	33,2	14,4	5,1	2,5	0
4	10	20	3	0,83%	6	0,20%	8	0,4	0,4	0,8	50,4	42,8	33,3	13,2	4
4	10	20	4	1,60%	3	0,05%	9	1,2	1,2	2	55,6	176	41,7	10,8	0,8
4	10	20	5	1,26%	5	0,00%	10	0,4	0,6	0,9	37,2	22	5,1	5,2	0
4	10	40	3	1,02%	7	0,12%	9	0,3	0,3	0,4	46	29,6	4,4	4,4	3,6
4	10	40	4	1,65%	4	0,50%	8	0,5	0,4	0,7	53,2	79,6	33,5	16,7	14,8
4	10	40	5	0,41%	6	0,00%	10	0,6	0,4	0,6	36	41,6	8	7,2	0
4	15	10	3	1,16%	3	0,66%	6	0,4	1	1,1	109,2	163,6	88	39,8	30,8
4	15	10	4	0,97%	4	0,57%	7	0,3	0,3	0,5	129,2	247,2	48,7	4,6	159,6
4	15	10	5	2,40%	2	0,61%	6	0,5	2	2,4	91,2	270,4	75,9	66,8	72,8
4	15	20	3	0,17%	8	0,07%	8	0	0,1	0,3	88,4	0	53,4	11,9	4
4	15	20	4	1,05%	4	0,10%	8	0,6	0,8	1,7	85,6	119,2	64,1	11,7	23,6
4	15	20	5	1,77%	3	0,55%	7	0,3	1,6	1,5	120	139,6	98,6	152	174,8
4	15	40	3	0,37%	7	0,00%	9	0,2	0,9	0,9	94,4	14,4	142,6	7,8	1,2
4	15	40	4	0,18%	7	0,00%	9	0,1	0,5	0,9	76,4	22,4	40	59,4	2,8
4	15	40	5	1,68%	6	0,46%	8	0	1,5	1,5	73,2	0	152	163	8,8
4	20	40	5	0,72%	4	0,00%	10	0,2	0,9	1	89,2	50,4	51	35,2	0
4	20	40	7	1,40%	1	0,35%	7	1,7	1,9	3,4	105,6	562,8	108,9	100,4	27,6
4	25	40	5	0,18%	4	0,16%	6	0,3	1,8	1	161,2	449,2	740,4	36,6	26,8
4	25	40	7	11,02%	1	10,34%	5	1	1,5	2,3	240,4	2700,4	340	100,2	616,4

Le nombre de contraintes ajoutées dans le problème auxiliaire est également faible. Ramené au nombre moyen de contraintes ajoutées par résolution du problème auxiliaire nous obtenons :
- au plus 19,9 contraintes dues au contraintes de capacité et 2,41 en moyenne ;
- au plus 41,1 contraintes dues au contraintes de clique et 1,84 en moyenne ;
- au plus 107,09 contraintes dues au contraintes de clique-cycle et 0,70 en moyenne ;
- au plus 7 contraintes dues au contraintes de branchement et 0,80 en moyenne.

• *Résultats expérimentaux*

Les tables 7.9 à 7.12 résument les résultats obtenus sur les instances aléatoires. Elles donnent les résultats suivants :

TAB. 7.9 – *Résolution exacte pour* $|N| = 1$

Instances				Problème Maître				
$\|N\|$	$\|U\|$	$\|W\|$	m	FG	$\#Opt$	$\#N$	$\#Var$	TT
1	10	10	3	0,00%	10	1	3,564	42,8
1	10	10	4	0,00%	10	2,4	2,513	33,7
1	10	10	5	0,00%	10	1	1,153	19,5
1	10	20	3	0,00%	10	1,8	9,358	44,8
1	10	20	4	0,00%	10	1	9,126	33,1
1	10	20	5	0,00%	10	1	6,238	22,6
1	10	40	3	0,00%	10	1,2	99,059	66,9
1	10	40	4	0,00%	10	1,7	347,102	42
1	10	40	5	0,00%	10	1	104,93	26
1	15	10	3	0,00%	10	1,7	12,728	64,9
1	15	10	4	0,00%	10	21,5	26,375	87,8
1	15	10	5	0,00%	10	5,3	20,638	56,1
1	15	20	3	0,00%	10	2,8	41,871	92
1	15	20	4	0,00%	10	2,2	68,494	83,5
1	15	20	5	0,00%	10	4,9	32,402	51,1
1	15	40	3	0,00%	10	1,3	255,897	105,7
1	15	40	4	0,04%	9	35,6	2176,816	170,7
1	15	40	5	0,57%	9	9	1278,615	85,3

FG : l'écart relatif entre la meilleure solution trouvée et la meilleure borne inférieure fournie.
#Opt : le nombre d'instances résolues à l'optimalité.
#N : le nombre de noeuds dans l'arbre de branchement.
#N : le nombre de variables générées.
TT : le temps total de calcul en secondes (limité à deux heures).

La qualité de la relaxation linéaire étant très bonne, la taille de l'arbre de branchement est très petite : 3,85 noeuds en moyenne, 275 noeuds dans le pire des cas.

Nous voyons également que le nombre minimal m de tournées n'a pas d'impact significatif sur le comportement de l'algorithme et donc que la difficulté de résolution semble davantage liée à la taille du graphe. m étant corrélé au nombre de clients, on ne peut toutefois pas construire d'instances telles que m soit très grand alors que le graphe reste de taille raisonnable.

On sait que l'augmentation de $|N|$ entraîne une combinatoire des anneaux-étoiles très importante. De plus, d'un point de vue pratique, à chaque itération, la formulation

TAB. 7.10 – *Résolution exacte pour* $|N| = 2$

Instances				Problème Maître				
$\|N\|$	$\|U\|$	$\|W\|$	m	FG	$\#Opt$	$\#N$	$\#Var$	TT
2	10	10	3	0,00%	10	2,4	3,314	45
2	10	10	4	0,00%	10	1,3	4,035	44,1
2	10	10	5	0,00%	10	1,9	1,951	26,2
2	10	20	3	0,00%	10	2,6	9,862	54,6
2	10	20	4	0,00%	10	1,3	15,547	48,1
2	10	20	5	0,00%	10	1,6	151,005	31,5
2	10	40	3	0,00%	10	2,2	384,873	66,3
2	10	40	4	0,00%	10	2,4	64,706	49,3
2	10	40	5	0,00%	10	1	153,356	28,7
2	15	10	3	0,00%	10	2,4	16,939	96
2	15	10	4	0,00%	10	1,8	14,011	71,5
2	15	10	5	0,00%	10	8,9	21,565	69,1
2	15	20	3	0,00%	10	1,4	31,231	101
2	15	20	4	0,00%	10	13,7	57,36	105,3
2	15	20	5	0,00%	10	1,3	29,376	64,2
2	15	40	3	0,00%	10	2,9	191,746	127,6
2	15	40	4	0,00%	10	2	502,91	100,3
2	15	40	5	0,00%	10	12,1	337,512	74,4

auxiliaire est appelée $|N|$ fois, c'est-à-dire une fois pour chaque dépôt. On aurait donc pu s'attendre à une baisse de performance quand $|N|$ augmente. Or il apparaît que ce n'est pas le cas, voir même qu'à taille de graphe identique, certaines instances à $|N| = 2$ génèrent moins de variables que certaines instances à $|N| = 1$.

En limitant à 2 heures CPU le temps d'exécution maximal, nous avons pu résoudre des instances allant jusqu'à 4 dépôts, 25 clients et 40 sommets Steiner. Nous avons pu résoudre optimalement 757 des 760 instances que nous avons générées. Pour la première (*resp.* seconde) des trois instances non résolues, nous pouvons cependant fournir une solution réalisable dont la valeur est à au plus 0,39% (*resp.* 5,65%) de l'optimum. Pour une seul de ces instances (à 4 dépôts, 25 clients et 40 sommets Steiner), nous n'avons pas pu fournir de solution réalisable dans la limite des 2 heures de calcul.

En conclusion, notre algorithme de Branch-and-Cut-and-Price est efficace. Ceci est dû à l'alliance des capacités de la génération de colonnes à éviter les symétries et des inégalités valides à améliorer la relaxation linéaire du problème maître. La mise en oeuvre d'un tel algorithme est cependant complexe à cause de l'obligation de prise en compte des coûts duaux des inégalités additionnelles. Néanmoins, notre méthode, basée

TAB. 7.11 – *Résolution exacte pour* $|N| = 3$

Instances				Problème Maître				
$\|N\|$	$\|U\|$	$\|W\|$	m	FG	$\#Opt$	$\#N$	$\#Var$	TT
3	10	10	3	0,00%	10	1	15,7	61,4
3	10	10	4	0,00%	10	1	2,523	39,4
3	10	10	5	0,00%	10	1	1,642	29,4
3	10	20	3	0,00%	10	1,5	9,94	61,9
3	10	20	4	0,00%	10	1,2	8,142	45,8
3	10	20	5	0,00%	10	1	6,571	31,7
3	10	40	3	0,00%	10	1,4	123,512	78,1
3	10	40	4	0,00%	10	1	507,007	52,1
3	10	40	5	0,00%	10	1	98,57	30,8
3	15	10	3	0,00%	10	1,4	14,476	108,2
3	15	10	4	0,00%	10	8	15,365	87,9
3	15	10	5	0,00%	10	14	38,395	76,9
3	15	20	3	0,00%	10	2,3	43,023	123,1
3	15	20	4	0,00%	10	3	62,347	90,8
3	15	20	5	0,08%	10	20,1	1354,363	92,5
3	15	40	3	0,00%	10	1	161,722	144,7
3	15	40	4	0,00%	10	7	1133,638	110,4
3	15	40	5	0,00%	10	1,8	707,242	65,7

sur un algorithme de Branch-and-Cut pour résoudre le problème auxiliaire, s'avère efficace pour mettre en oeuvre un telle alliance. Le développement d'outils tels que SCIP rend possible l'implémentation d'un tel algorithme. Il serait intéressant d'appliquer notre méthode à d'autres problèmes comme par exemple le CVRP ou le problème de Coloration de graphe.

7.3 Conclusion

Dans ce chapitre, nous avons détaillé la construction d'un algorithme de Branch-and-Cut-and-Price basé sur les résultats théoriques du chapitre précédent. Les résultats expérimentaux ont montré que l'algorithme de Branch-and-Cut-and-Price était très compétitif et bien plus efficace que l'algorithme de Branch-and-Cut du chapitre 5. De plus, l'introduction d'inégalités valides connues pour leur efficacité dans le problème maître nous a permis d'obtenir dans de très nombreux cas une solution optimale dès la racine de l'arbre de branchement.

TAB. 7.12 – *Résolution exacte pour* $|N| = 4$

Instances				Problème Maître				
$\|N\|$	$\|U\|$	$\|W\|$	m	FG	$\#Opt$	$\#N$	$\#Var$	TT
4	10	10	3	0,00%	10	3,1	4,522	54,7
4	10	10	4	0,00%	10	1	11,435	53,4
4	10	10	5	0,00%	10	1	2,116	34,2
4	10	20	3	0,00%	10	1,6	9,164	51,9
4	10	20	4	0,00%	10	1,2	11,953	53,6
4	10	20	5	0,00%	10	1	10,352	38,3
4	10	40	3	0,00%	10	1,4	88,694	58,9
4	10	40	4	0,00%	10	2	303,667	57,5
4	10	40	5	0,00%	10	1	201,674	38,1
4	15	10	3	0,00%	10	2,6	17,578	131,2
4	15	10	4	0,00%	10	6,3	20,083	115,2
4	15	10	5	0,00%	10	4,7	16,791	72,4
4	15	20	3	0,00%	10	1,4	41,4	137
4	15	20	4	0,00%	10	2,3	32,946	92,4
4	15	20	5	0,00%	10	6,9	225,042	84,6
4	15	40	3	0,00%	10	1,2	291,33	155,8
4	15	40	4	0,00%	10	1,4	354,567	108,5
4	15	40	5	0,00%	10	2	381,95	70,6
4	20	40	5	0,00%	10	1	521,32	127,7
4	20	40	7	0,00%	10	2,8	1391,54	101,7
4	25	40	5	0,00%	10	2,6	1660,31	222
4	25	40	7	0,06%	9	16,1	3161,72	199,6

Nous avons montré comment l'utilisation d'un algorithme de Branch-and-Cut pour le problème auxiliaire permet de prendre en compte de manière exhaustive des inégalités issues du polytope du Stable dans le problème maître. Remarquons que les principes énoncés ici s'appliquent de manière générique pour les problèmes de génération de colonnes.

Conclusion

Dans cette thèse, nous avons étudié le problème de conception de réseaux fiables de technologie SDH seconde génération. Nous avons montré que ce problème se ramène au problème de couverture des sommets d'un graphe par des anneaux-étoiles non-disjoints.

Nous avons étudié la version mono-dépôt de ce problème, le 1-γ-RSP, et discuté de la caractérisation d'une solution à ce problème. Ayant montré qu'il n'existe pas de formulation PLNE à 2 indices, nous avons proposé trois formulations PLNE à 3 indices avec un nombre polynomial de variables et avons comparé la valeur de leur relaxation linéaire. Nous avons également adapté plusieurs familles d'inégalités valides, inspirées de la littérature du TSP et du CVRP, pour renforcer l'une de ces formulations, dite naturelle.

Une étude polyédrale menée sur le dominant de la formulation naturelle nous a permis de donner des conditions nécessaires et suffisantes pour que certaines classes d'inégalités de la formulation définissent des facettes du polytope. Grâce à cette étude, nous avons pu développer un algorithme de Branch-and-Cut pour résoudre des instances du 1-γ-RSP générées aléatoirement ou issues d'un réseau réel.

Nous avons ensuite étudié la version multi-dépôts de ce problème, le k-γ-RSP. Nous avons proposé une formulation set partitionning résolue par une méthode de génération de colonnes. Nous avons montré que la résolution du problème auxiliaire de cette formulation ne pouvait se faire efficacement par une approche algorithmique. Nous avons donc proposé d'utiliser l'algorithme de Branch-and-Cut développé pour le cas mono-dépôt pour le résoudre.

Nous avons également montré comme l'utilisation d'un Branch-and-Cut pour résoudre le problème auxiliaire permet d'ajouter des contraintes issues des polytope du Stable au problème maître. La bonne qualité de la relaxation linéaire de la formulation set partitionning, ajoutée à l'efficacité des inégalités que nous avons utilisées, nous a permis d'implémenter un algorithme de Branch-and-Cut-and-Price très efficace pour le k-γ-RSP.

Ce travail de recherche ouvre plusieurs perspectives de recherche liées à la fois à la théorie et à l'expérimentation numérique.

La formulation naturelle que nous avons proposée souffre beaucoup de la symétrie des solutions : en effet, l'utilisation d'un troisième indice induit une importante symétrie artificielle. Il serait intéressant de voir l'impact des nouvelles techniques [88, 68] autour des orbitopes et du branchement orbitopal. De plus, une étude plus approfondie du polytope pourrait permettre l'introduction de nouvelles inégalités valides et l'amélioration globale de l'algorithme. Une telle étude se justifierait aussi de par le fait que toute amélioration de l'algorithme de Branch-and-Cut permettrait une amélioration de l'algorithme de Branch-and-Cut-and-Price dédié au cas multi-dépôts.

Une autre perspective très intéressante serait de tester l'efficacité d'un algorithme de Branch-and-Cut-and-Price utilisant toutes les inégalités issues du polytope du Set Partitionning (ou Set Covering) sur un problème pour lequel les approches par génération de colonnes ont donné de bons résultats, par exemple les problèmes de tournées de véhicules, de coloration ou de cutting-stock. Les algorithmes de Branch-and-Cut permettent maintenant de résoudre très efficacement des instances de taille raisonnable pour un certain nombre de problèmes combinatoires NP-complet tels que le TSP, même dans sa version Steiner. L'amélioration de la relaxation linéaire due à l'utilisation des contraintes du polytope du Set Partitionning dans le problème maître pourrait peut être compenser le coût supplémentaire en temps de calcul par rapport à une résolution algorithmique du problème auxiliaire.

Bibliographie

[1] T. Achterberg, "SCIP: solving constraint integer programs", *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, "Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems", *Mathematical Programming*, pages 91–153, 2003.

[3] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, and D. Naddef, "Separating capacity constraints in the CVRP using tabu search", *European Journal of Operational Research*, 106(2-3):546–557, 1998.

[4] M. Baïou and A.R. Mahjoub, "Steiner 2-edge connected subgraph polytopes on series-parallel graphs", *SIAM Journal on Discrete Mathematics*, 10(3):505–514, 1997.

[5] M. Baïou and A.R. Mahjoub, "The Steiner traveling salesman polytope and related polyhedra", *SIAM Journal on Optimization*, 13:498, 2002.

[6] R. Baldacci, N. Christofides, and A. Mingozzi, "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts", *Mathematical Programming*, 115(2):351–385, 2008.

[7] R. Baldacci and M. Dell'Amico, "Heuristic algorithms for the multi-depot ring-star problem", *European Journal of Operational Research*, 203(1):270–281, 2010.

[8] R. Baldacci, M. Dell'Amico, and J.J. Gonzalez Salazar, "The capacitated m-ring-star problem", *Operations Research*, 55(6):1147, 2007.

[9] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation", *Operations Research*, pages 723–738, 2004.

[10] R. Baldacci, A. Mingozzi, and R. Roberti, "New route relaxation and pricing strategies for the vehicle routing problem", *Operations research*, 59(5):1269–1283, 2011.

[11] M.L. Balinski and R.E. Quandt, "On an integer program for a delivery problem", *Operations Research*, 12(2):300–304, 1964.

[12] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, "An application of combinatorial optimization to statistical physics and circuit layout design", *Operations Research 36*, pages 493–513, 1988.

[13] F. Barahona and A.R. Mahjoub, "On the cut polytope", *Mathematical Programming 36*, pages 157–173, 1986.

[14] F. Barahona and A.R. Mahjoub, "Facets of the balanced (acyclic) induced subgraph polytope", *Mathematical Programming 45*, pages 21–34, 1989.

[15] F. Barahona and A.R. Mahjoub, "On two-connected subgraph polytopes", *Discrete Mathematics*, 147(1):19–34, 1995.

[16] P. Bauer, "The circuit polytope: Facets", *Mathematics of Operations Research*, 22(1):110–145, 1997.

[17] F. Bendali, I. Diarrassouba, M. Didi Biha, A.R. Mahjoub, and J. Mailfert, "A branch-and-cut algorithm for the k-edge-connected subgraph problem", *Networks*, 55:13–32, 2010.

[18] F. Bendali, I. Diarrassouba, A.R. Mahjoub, and J. Mailfert, "On the k-edge-disjoint 3-hop-constrained paths polytope", discrete optimization", *Discrete Optimization*, 7:222–233, 2010.

[19] M. Didi Biha and A.R. Mahjoub, "k-edge connected polyhedra on series-parallel graphs", *Operations research letters*, 19(2):71–78, 1996.

[20] M. Didi Biha and A.R. Mahjoub, "The k-edge connected subgraph problem : Polytopes and critical extreme points", *Linear algebra and its applications*, 381:117–139, 2004.

[21] N. Boland, J. Dethridge, and I. Dumitrescu, "Accelerated label setting algorithms for the elementary resource constrained shortest path problem", *Operations Research Letters*, 34(1):58–68, 2006.

[22] M.A. Boschetti, V. Maniezzo, M. Roffilli, and A.B. Röhler, "Matheuristics: Optimization, simulation and control", In *Hybrid Metaheuristics*, pages 171–177. Springer, 2009.

[23] Q. Botton, B. Fortz, L. Gouveia, and M. Poss, "Benders decomposition for the hop-constrained survivable network design problem", *to appear in INFORMS Journal on Computing.*

[24] P. Chardaire, J.L. Lutton, and A. Sutter, "Upper and lower bounds for the two-level simple plant location problem", *Annals of Operations Researc*, 86:117–140, 1999.

[25] S. Chopra, "The k-edge-connected spanning subgraph polyhedron", *SIAM Journal on Discrete Mathematics*, 7(2):245–259, 1994.

[26] W. Cook and J.L. Rich, "A parallel cutting-plane algorithm for the vehicle routing problem with time windows", *Computational and Applied Mathematics Department, Rice University, Houston, TX, Technical Report*, 1999.

[27] G. Cornuéjols, J. Fonlupt, and D. Naddef, "The traveling salesman problem on a graph and some related integer polyhedra", *Mathematical programming*, 33(1):1–27, 1985.

[28] G. Cornuejols and F. Harche, "Polyhedral study of the capacitated vehicle routing problem, on the p-median polytope", *Mathematical Programming*, 60(1-3):21–52, 1991.

[29] J. Current and H. Pirkul, "The hierarchical network design problem with transshipment facilities", *European Journal of Operations Research*, 52:338–347, 1991.

[30] G. Dahl, "Notes on polyhedra associated with hop-constrained paths", *Operations research letters*, 25(2):97–100, 1999.

[31] G. Dahl, D. Huygens, A.R. Mahjoub, and P. Pesneau, "On the k edge-disjoint 2-hop-constrained paths polytope", *Operations Research Letters*, 34:577–582, 2006.

[32] G. Desaulniers, F. Lessard, and A. Hadjar, "Tabu search, partial elementarity and generalized k-path inequalities for the vehicle routing problem with time windows", *Transportation Science*, 42(3):387–404, 2008.

[33] M. Desrochers, J. Desrosiers, and M.M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows", *Operations research*, 40(2):342–354, 1992.

[34] M. Desrochers and G. Laporte, "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints.", *Operations Research Letters*, 10(1):27–36, 1991.

[35] C.S. Thayse Dias, G.F. de Sousa Filho, E.M. Macambira, F.C. Lucidio dos Anjos, and M.H.C. Fampa, "An efficient heuristic for the ring star problem", In *Experimental Algorithms*, pages 24–35. Springer, 2006.

[36] E.W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische mathematik*, 1(1):269–271, 1959.

[37] J. Edmonds, "Maximum matching and a polyhedron with 0,1-vertices", *Journal of Research National Bureau of Standards*, pages 125–130, 1965.

[38] O. Ekin-Karazan, P. Fouilhoux, A.R. Mahjoub, O. Özkök, and H. Yaman, "Survivability in hierarchical telecommunications networks", In *INOC 2009 International Network Optimization Conference*, 2009.

[39] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems", *Networks*, 44(3):216–229, 2004.

[40] T.A. Feo and M.G.C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem", *Operations research letters*, 8(2):67–71, 1989.

[41] M. Fischetti, J.J. González Salazar, and P. Toth, "Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem", In *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation, 169–173*. Citeseer, 1995.

[42] M.L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing", *Networks*, 11(2):109–124, 1981.

[43] J. Fonlupt and A.R. Mahjoub, "Critical extreme points of the 2-edge connected spannning subgraph polytope", In *Integer Programming and Combinatorial Optimization*, pages 166–182. Springer, 1999.

[44] B. Fortz and M. Labbe, "Polyhedral results for two-connected networks with bounded rings", *Mathematical Programming*, 93:27–54, 2002.

[45] B. Fortz, M. Labbe, and F. Maffioli, "Solving the two-connected network with bounded meshes problem", *Operations Researc*, 58:866–877, 2000.

[46] B. Fortz, A.R. Mahjoub, S.T. McCormick, and P. Pesneau, "Two-edge connected subgraphs with bounded rings: Polyhedral results and branch-and-cut", *Mathematical Programming*, 105:85–111, 2006.

[47] P. Fouilhoux, O. Karasan, A.R. Mahjoub, O. Ozkok, and H. Yaman, "Survivability in hierarchical telecommunications networks", *Networks*, 59:37–58, 2012.

[48] P. Fouilhoux and A. Questel, "The non-disjoint m-ring-star problem: polyhedral results and sdh/sonet network design", *Combinatorial Optimization*, pages 93–104, 2012.

[49] R. Fukasawa, H. Longo, J. Lysgaard, M.P. de Aragão, M. Reis, E. Uchoa, and R.F. Werneck, "Robust branch-and-cut-and-price for the capacitated vehicle routing problem", *Mathematical programming*, 106(3):491–511, 2006.

[50] M.R. Garey and D.S. Johnson *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences* WH Freeman, Company, San Francisco, Calif, 1979.

[51] B. Gavish, "Topological design of centralized computer networks: Formulations and algorithms", *Networks*, 12:355–377, 1982.

[52] B. Gavish and S.C. Graves, "The travelling salesman problem and related problems", *Operations Research Center Working Paper; OR 078-78*, 1978.

[53] F. Glover and M. Laguna et al. *Tabu search*, volume 22 Springer, 1997.

[54] R. Gourdin, M. Labbe, and H. Yaman, "Telecommunication and location", *Facility Location: Applications and Theory*, pages 275–305, 2001.

[55] L. Gouveia, "A result on projection for the vehicle routing problem", *European Journal of Operational Research*, 85(3):610–624, 1995.

[56] M. Grötschel, L. Lovasz, and A. Schrijver, "Geometric algorithms and combinatorial optimization", *Springer-Verlag*, 1985.

[57] M. Grötschel and C.L. Monma, "Integer polyhedra arising from certain network design problems with connectivity constraints", *SIAM Journal on Discrete Mathematics*, 3(4):502–523, 1990.

[58] M. Grötschel, C.L. Monma, and M. Stoer *Polyhedral approaches to network survivability* Inst. für Mathematik, 1990.

[59] M. Grötschel, C.L. Monma, and M. Stoer, "Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints", *Operations Research*, 40(2):309–330, 1992.

[60] M. Grötschel, C.L. Monma, and M. Stoer, "Facets for polyhedra arising in the design of communication networks with low-connectivity constraints", *SIAM Journal on Optimization*, 2(3):474–504, 1992.

[61] M. Grötschel, C.L. Monma, and M. Stoer, "Design of survivable networks", *Handbooks in Operations Research and Management Science*, 7:617–672, 1995.

[62] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi, "A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations", *Annals of Operations Research*, 61(1):21–43, 1995.

[63] E.A. Hoshino and C.C. de Souza, "Column generation algorithms for the capacitated m-ring-star problem", In *Computing and Combinatorics*, pages 631–641. Springer, 2008.

[64] E.A. Hoshino and C.C. de Souza, "A branch-and-cut-and-price approach for the capacitated m-ring–star problem", *Discrete Applied Mathematics*, 2011.

[65] E.A. Hoshino and C.C. de Souza, "A branch-and-cut-and-price approach for the capacitated m-ring–star problem", *Discrete Applied Mathematics*, 160(18):2728–2741, 2012.

[66] S. Irnich and D. Villeneuve, "The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$", *INFORMS Journal on Computing*, 18(3):391–406, 2006.

[67] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger, "Subset-row inequalities applied to the vehicle-routing problem with time windows", *Operations Research*, 56(2):497–511, 2008.

[68] V. Kaibel, M. Peinhardt, and M.E. Pfetsch *Orbitopal fixing* Springer, 2007.

[69] S. Kedad-Sidhoum and V.H. Nguyen, "An exact algorithm for solving the ring star problem", *Optimization*, 59(1):125–140, 2010.

[70] H. Kerivin and A.R. Mahjoub, "Design of survivable networks: A survey", *Networks*, 46(1):1–21, 2005.

[71] H. Kerivin, A.R. Mahjoub, and C. Nocq, "(1, 2)-survivable networks: Facets and branch&cut", *The Sharpest Cut, MPS-SIAM Series in Optimization*, pages 121–152, 2004.

[72] J.G. Klincewicz, "Hub location in backbone/tributary network design: a review", *Location Science*, 6:307–335, 1998.

[73] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis, "2-path cuts for the vehicle routing problem with time windows", *Transportation Science*, 33(1):101–116, 1999.

[74] J.B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[75] M. Labbe, G. Laporte, I. Martín Rodriguez, and J.J. Gonzalez Salazar, "The ring star problem: Polyhedral analysis and exact algorithm", *Networks*, 43(3):177–189, 2004.

[76] M. Labbe, H. Yaman, and E. Gourdin, "A branch and cut algorithm for hub location problems with single assignment", *Mathematical Programming*, 102:371–405, 2005.

[77] G. Laporte, Y. Nobert, and M. Desrochers, "Optimal routing under capacity and distance restrictions", *Operations research*, 33(5):1050–1073, 1985.

[78] Y. Lee, B.H. Lim, and J.S. Park, "A hub location problem in designing digital data service networks: Lagrangian relaxation approach", *Location Science*, 4:185–194, 1996.

[79] A.N. Letchford, R.W. Eglese, and J. Lysgaard, "Multistars, partial multistars and the capacitated vehicle routing problem", *Mathematical Programming*, 94(1):21–40, 2002.

[80] A.N. Letchford and J.J. González Salazar, "Projection results for vehicle routing", *Mathematical Programming*, 105(2-3):251–274, 2006.

[81] A.R. Mahjoub, "Two-edge connected spanning subgraphs and polyhedra", *Mathematical Programming*, 64(1-3):199–208, 1994.

[82] A. Mauttone, S.Nesmachnow, A. Olivera, and F. Robledo, "A hybrid metaheuristic algorithm to solve the capacitated m-ring star problem", In *International Network Optimization Conference*, 2007.

[83] K. Menger, "Zur allgemeinen kurventheorie", *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[84] C.E. Miller, A.W. Tucker, and R.A. Zemlin, "Integer programming formulation of traveling salesman problems", *Journal of the ACM (JACM)*, 7(4):329, 1960.

[85] M. Stoer *Design of survivable networks*, volume 1531 Springer-Verlag Berlin, 1992.

[86] Z. Naji-Azimi, M. Salari, and P. Toth, "An integer linear programming based heuristic for the capacitated m-ring-star problem", *European Journal of Operational Research*, 217(1):17–25, 2012.

[87] G.L. Nemhauser and G. Sigismondi, "A strong cutting plane/branch-and-bound algorithm for node packing", *Journ. of the Operations Research Society*, pages 443–457, 1992.

[88] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio, "Orbital branching", In *Integer programming and combinatorial optimization*, pages 104–118. Springer, 2007.

[89] J.A. Moreno Pérez, J.M. Moreno-Vega, and I. Martin Rodriguez, "Variable neighborhood tabu search and its application to the median cycle problem", *European Journal of Operational Research*, 151(2):365–378, 2003.

[90] H. Pirkul and V. Nagarajan, "Locating concentrators in centralized computer networks", *Annals of Operations Research*, 36:247–26, 1992.

[91] R.C. Prim, "Shortest connection networks and some generalizations", *Bell system technical journal*, 36(6):1389–1401, 1957.

[92] W.R. Pulleyblank, "Polyhedral combinatorics", *Handbooks in Operations Research and Management Science*, pages 371–446, 1989.

[93] G. Righini and M. Salani, "New dynamic programming algorithms for the resource constrained elementary shortest path problem", *Networks*, 51(3):155–170, 2008.

[94] D. Ryan and D.B. Foster, "An integer programming approach to scheduling", *Computer Scheduling of Public Transport, A. Wren (Ed.)*, pages 269–280, 1981.

[95] M. Salari, A. Naji-Azimi, and P. Toth, "A variable neighborhood search and its application to a ring star problem generalization", *Electronic Notes in Discrete Mathematics*, 36:343–350, 2010.

[96] J. Sanchez, "A branch and cut algorithm for the steiner ring star problem", *International Journal of Management Science*, 4(1):21–34, 1998.

[97] A. Schrijver, "Combinatorial optimization - polyhedra and efficiency", *Springer*, 2003.

[98] S. Spoorendonk and G. Desaulniers, "Clique inequalities applied to the vehicle routing problem with time windows", *INFOR: Information Systems and Operational Research*, 48(1):53–67, 2010.

[99] J.W. Suurballe, "Disjoint paths in a network", *Networks*, 4(2):125–145, 1974.

[100] J.W. Suurballe and R.E. Tarjan, "A quick method for finding shortest pairs of disjoint paths", *Networks*, 14(2):325–336, 1984.

[101] P. Winter, "Generalized steiner problem in halin graphs", In *Proceedings of the 12th International Symposium on Mathematical Programming*, 1985.

[102] P. Winter, "Generalized steiner problem in series-parallel networks", *Journal of Algorithms*, 7(4):549–566, 1986.

[103] P. Winter, "Steiner problem in networks: A survey", *Networks*, pages 129–167, 1987.