UNIVERSITÉ DE MONTRÉAL

LE PROBLÈME DU POSTIER CHINOIS CUMULATIF

NIKOLAJ VAN OMME
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL)
MAI 2011

UNIVERSITÉ DE MONTREAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

LE PROBLÈME DU POSTIER CHINOIS CUMULATIF

présentée par: M. VAN OMME Nikolaj, M. Sc.
en vue de l'obtention du diplôme de: Philosophiæ Doctor
a été dûment acceptée par le jury constitué de:

M. ROUSSEAU Louis-Martin, Ph.D., président.
M. GENDREAU Michel, Ph.D., membre et directeur de recherche.
M. SORIANO Patrick, Ph.D., membre et co-directeur de recherche.
M. SOUMIS François, Ph.D., membre.
M. VIGO Daniele, Ph.D., membre externe.

Aux autres,
à tous les autres ...

Remerciements

Je remercie mes deux directeurs, Messieurs Michel Gendreau et Patrick Soriano pour leur aide, leur patience et leurs conseils judicieux tout au long du processus ayant mené à la réalisation de cette thèse.

Je remercie Monsieur Jean-Marc Rousseau pour son soutien financier à une partie non négligeable de mes études ainsi que Monsieur Gilles Savard qui s'est toujours gentiment enquis de mon état.

Ma rencontre en 2008 avec Monsieur Ángel Corberán et Monsieur José María Sanchís a été déterminante. Notre collaboration fructueuse fut un véritable plaisir sans cesse renouvelé et j'espère avoir l'honneur et le bonheur de pouvoir continuer à travailler avec eux sur ce sujet difficile qu'est le problème du postier chinois cumulatif. Je tiens à préciser que les noms folkloriques attribués aux inégalités valides sont de mon cru et que j'en assume l'entière responsabilité. Mil gràcies!

Messieurs Eric Springuel et Walter Rei, avec qui j'ai eu le bonheur de partager un bureau, reçoivent aussi toute ma gratitude. Nos longues discussions aussi bien en recherche opérationnelle que sur la vie en générale m'ont fortement inspiré et continuent de m'inspirer encore maintenant.

Mademoiselle mari Albareda Sambola qui m'a fait partager sa passion de la recherche opérationnelle acquière toute mon estime ainsi que des remerciements qui vont bien au-delà de ces quelques mots.

Je remercie également Sylvain Crouzet pour son soutien et son amitié.

Une thèse ne se fait pas tout seul. Au CIRRELT, toute une équipe dévouée nous aide. J'aimerais mentionner certaines personnes qui m'ont particulièrement aidé : Monsieur Daniel Charbonneau, Monsieur Luc Rocheleau, Monsieur Serge Bisaillon, Monsieur François Guertin, Madame Lucie-Nathalie Cournoyer ainsi que Madame Lucie L'Heureux. À l'École Polytechnique aussi, j'ai bénéficié d'un soutien important de la part de Madame Diane Bernier,

Madame Suzanne Guindon et Marie-Carline Lajeunesse avec qui j'ai toujours eu beaucoup de plaisir à discuter. Je remercie chacune et chacun d'entre-vous chaleureusement.

J'aimerais sincèrement remercier toute l'équipe dynamique de SCIP, le solveur utilisé. J'aimerais mentionner que non seulement j'ai adoré travaillé avec ce logiciel mais aussi que l'équipe a toujours été prompte à répondre à mes questions dans des délais qui souvent ne dépassaient pas les quelques heures. La façon dont les (rares) bugs sont traités est aussi exemplaire : ils sont éliminés dans les quelques jours suivant leur découverte.

Ces remerciements n'auraient aucun sens si je ne mentionnais Madame Catherine de Borghrave et Monsieur Albert Carton. Leur amitié indéfectible depuis maintenant 20 ans est toujours un don du ciel et une source intarissable de joie pour moi. Je les remercie vivement pour tout ce qu'ils ont fait pour moi et croyez-moi ce n'est pas rien. Béni soit le jour de notre rencontre.

Plusieurs personnes m'ont supporté financièrement pendant ces longues études. Je vous remercie vivement. Un grand merci donc à Albert Carton, Catherine de Borghgrave, Patrick Soriano, Séverine Dégée, Micheline Roelandt, Sylvain Crouzet, Sabine Delhaye, Liliane Plouvier et Jo-Jo Delmuth. Sans vous, je n'aurais jamais été capable de mener ces études à terme.

Je tiens à remercier amicalement encore une fois Messieurs Abdullah el Blahbla et Ernferst von Blüblü pour leur présence discrète mais néanmoins fort appréciée.

Thanks Mo for being so cool.

Я благодарю Бассем и Хлоя и я надеюсь, что в один прекрасный день мы будем
идти в Россию.

أشكر السيدة هند عطوي لضحكتها المعدية. سنلتقي قريباً إنشاء الله.

Mes remerciements vont aussi à Jimmy Skelling, Ruddy Avalos ainsi qu'aux nouveaux arrivants bien sympathiques que sont Ola Jabali et Marco Veneroni.

Finalement, je remercie mademoiselle Dania El-Khechen pour son soutien constant et énergisant pendant une partie non négligeable de mes études. Merci du fond du cœur d'être présente à mes côtés et de me supporter dans tous les sens du terme.

Résumé

Le sujet de cette thèse est le problème du postier chinois cumulatif (PPCC). Dans ce problème, nous considérons l'importance du moment où une arête est traitée complètement. Cette façon de procéder introduit un caractère cumulatif et dynamique dans le coût réel des arêtes, ce qui a pour effet de changer la structure du problème du postier chinois. Nous démontrons que ce problème est fortement NP-difficile et réductible à une version du problème de voyageur de commerce cumulatif. Ce problème est, à notre connaissance, nouveau. Nous continuons ici l'étude entreprise dans notre mémoire de maîtrise (van Omme, 2003). Notre but dans cette thèse est de résoudre exactement ce problème à l'aide des outils de la programmation linéaire en nombres entiers.

Notre contribution est de plusieurs ordres. Premièrement, nous développons une vingtaine de modèles différents. Dans cette thèse, nous étudions les huit meilleurs et les comparons aussi bien empiriquement que théoriquement entre eux et démontrons toutes les relations de dominance entre eux. L'aboutissement de nos recherches est le modèle L8. Deuxièmement, nous résolvons ce modèle L8 à l'aide d'un algorithme de séparation et évaluation progressive (*Branch and Cut* — algorithme BC1). Nous développons plusieurs outils dont nous présentons ici trois branchements, sept pré-traitements, six familles de coupes dont trois que nous généralisons. Ces outils nous permettent déjà de battre le solveur CPLEX par un facteur de 3 à 58 sur nos graphes de référence. Troisièmement, nous développons une meilleure variante du modèle L8 : le modèle L8$^+$ et utilisons une approche avec génération de colonnes (*Branch, Price and Cut* — algorithme BPC1). Dans la foulée, nous développons cinq familles de coupes et nous généralisons quatre d'entre-elles. Cette nouvelle approche, plus rapide que la première d'un facteur de 2 à 4, nous permet d'être de 2 à 133 fois plus rapide que le solveur CPLEX en utilisant le modèle L8$^+$ sur nos graphes de référence. Quatrièmement, nous améliorons notre approche de génération de colonnes (*Branch, Price and Cut* — algorithme BPC2) avec une évaluation implicite du dual.

Les plus grandes instances du PPCC que nous arrivons à résoudre dans un délai maximal d'une heure comprennent des graphes de 11 sommets et/ou de 55 arêtes, ce qui correspond approximativement à des instances du problème du voyageur de commerce cumulatif à 110 sommets.

Abstract

The subject of this Ph.D. thesis is the Cumulative Chinese Postman Problem (CCPP). We focus on the delay of the service of each arc. This introduces a cumulative and dynamic aspect in the objective function therefore changing the structure of the Chinese Postman Problem. We prove that this problem is strongly NP-hard and reducible to a version of the Cumulative Traveling Salesman Problem. This problem is, to our knowledge, entirely new. The study of this problem was initiated in our master thesis (van Omme, 2003). Our main goal in this thesis is to solve this problem exactly with the help of the tools of linear integer programming.

Our contribution is manifold. First, we develop twenty different models. However, in this thesis, we only discuss and compare theoretically and experimentally the best eight models. We prove all dominance relations among them. Model L8 stands out as the best model. Secondly, we solve this model L8 with a Branch and Cut (algorithm BC1). Throughout our study, we develop several tools among which three branching rules, seven presolving algorithms, six families of cuts (three of them generalized). These tools alone allow us to solve the problem faster than CPLEX by a factor of 3 to 58 on our test graphs. Thirdly, we develop an improved model $L8^+$ and use a column generation approach - a Branch, Price and Cut (algorithm BPC1). We also develop five new families of cuts (four of them generalized). This new approach is faster than the previous one by a factor of 2 to 4 and is faster than CPLEX with the new model $L8^+$ by a factor of 2 to 133 on our test graphs. Fourthly, we improve our Branch, Price and Cut algorithm (algorithm BPC2) by using an implicit evaluation of the dual.

The largest instances for which we are able to solve the CCPP in less than one hour include graphs with 11 nodes and/or 55 edges which correspond approximately to instances of the Cumulative Traveling Salesman Problem with 110 nodes.

Table des matières

Dédicace iii

Remerciements iv

Résumé vi

Abstract vii

Table des matières viii

Liste des tableaux xiv

Liste des figures xvi

Liste des annexes xx

Liste des sigles et abréviations xxi

Chapitre 1 INTRODUCTION 1
- 1.1 Motivations 1
  - 1.1.1 Le problème initial 2
  - 1.1.2 Clients versus Serveur 3
  - 1.1.3 Le temps : un facteur crucial 4
- 1.2 Définitions et concepts de base 5
  - 1.2.1 Définition du problème du postier chinois cumulatif 5
  - 1.2.2 Conséquences de la définition 6
  - 1.2.3 Justification de la définition choisie 7
- 1.3 Un exemple introductif 8
- 1.4 Terminologie 9
  - 1.4.1 Les chemins partiels 10
  - 1.4.2 Les liaisons inter-clients 10
  - 1.4.3 Les diamètres 11
  - 1.4.4 Coûts relatifs et cumulatifs 12
  - 1.4.5 Une fonction objectif cumulative ? 13

1.4.6 Une solution modèle 13
1.5 Le modèle L1 14
1.5.1 Familles de variables pour les arêtes 14
1.5.2 Familles de variables pour les sommets 15
1.5.3 Le modèle 16
1.5.4 La résolution de l'exemple introductif 16
1.6 Choix des logiciels et déroulement des tests expérimentaux 18
1.7 Objectifs de recherche 19
1.8 Plan de la thèse 19

Chapitre 2 REVUE DE LITTÉRATURE 21
2.1 Le problème du postier chinois classique 21
2.1.1 Le cas des graphes non dirigés et dirigés 22
2.1.2 Le cas des graphes mixtes 23
2.1.3 D'autres cas 24
2.2 La version hamiltonienne du PPCC : le problème de latence minimale (PLM) 24
2.2.1 Définition 25
2.2.2 Nomenclature 26
2.2.3 La complexité 26
2.2.4 Les approches exactes 27
2.2.5 Les approximations 30
2.2.6 Les cas polynomiaux 32
2.3 Conclusions 32

Chapitre 3 PROPRIÉTÉS DU PROBLÈME 33
3.1 Le PPCC est NP-difficile 33
3.2 Différences avec le problème du postier chinois classique 34
3.2.1 Un chemin eulérien n'est pas optimal 34
3.2.2 Le graphe augmenté n'est pas d'un grand secours 34
3.2.3 Le problème ne peut pas être subdivisé en sous-problèmes sur des sous-graphes 36
3.3 Un problème similaire au PLM ? 36
3.3.1 Une transformation du PPCCAR en PLM 38
3.4 Cas polynomiaux 41

3.4.1 Le cercle et la ligne droite 41
3.4.2 L'arbre unitaire 42
3.4.3 La bande unitaire 43
3.5 La conjecture sur les grilles unitaires 43
3.5.1 Conjecture pour la bande unitaire de largeur 2 44
3.6 Conclusions 44

Chapitre 4 COMPARAISONS DE DIFFÉRENTS MODÈLES 47
4.1 Présentation des différents modèles 48
4.1.1 L2 49
4.1.2 L3 52
4.1.3 L4 54
4.1.4 L5 57
4.1.5 L6 59
4.1.6 L7 62
4.1.7 L8 64
4.2 Comparaisons sommaires des modèles 69
4.2.1 Comparaison du nombre de variables et de contraintes 70
4.2.2 Comparaison des relaxations linéaires 71
4.2.3 Comparaison des temps de résolution du PPCC 73
4.3 Comparaisons analytiques des bornes obtenues par relaxation 74
4.3.1 $z_{\text{rel}}(\text{L1}) \leq z_{\text{rel}}(\text{L4})$ 74
4.3.2 $z_{\text{rel}}(\text{L3}) \leq z_{\text{rel}}(\text{L5})$ 75
4.3.3 $z_{\text{rel}}(\text{L3}) \leq z_{\text{rel}}(\text{L1})$ 76
4.3.4 $z_{\text{rel}}(\text{L3}) \leq z_{\text{rel}}(\text{L4})$ 77
4.3.5 $z_{\text{rel}}(\text{L2}) \leq z_{\text{rel}}(\text{L4})$ 78
4.3.6 $z_{\text{rel}}(\text{L6}) \leq z_{\text{rel}}(\text{L8})$ 82
4.3.7 $z_{\text{rel}}(\text{L7}) \leq z_{\text{rel}}(\text{L8})$ 83
4.3.8 $z_{\text{rel}}(\text{L4}) \leq z_{\text{rel}}(\text{L8})$ 85
4.3.9 Relations de dominance entre les modèles 87
4.4 Conclusions 88

Chapitre 5 RÉSOLUTION DU MODÈLE L8 : PREMIÈRE APPROCHE 90
5.1 Discussion sommaire du modèle L8 91
5.1.1 Une surabondance de variables et de contraintes 91

5.1.2 La dimension du modèle L8 92
5.2 Conventions pour le modèle L8 94
5.2.1 Les sommes 94
5.2.2 Les valeurs spécifiques 95
5.2.3 Les ensembles 95
5.3 Recherche d'une solution initiale 96
5.4 Pré-traitements des variables 97
5.4.1 Le pré-traitement *mandatory edge* 98
5.4.2 Le pré-traitement *shortest path minimum number of edges* 98
5.4.3 Le pré-traitement *reachable edge* 99
5.4.4 Le pré-traitement *reachable shortest path* 100
5.4.5 Le pré-traitement *depot* 100
5.4.6 Le pré-traitement *backward compatibility* 101
5.4.7 Le pré-traitement *forward compatibility* 102
5.5 Les inégalités de parité 103
5.5.1 Deux lemmes importants 104
5.5.2 Les inégalités *vertex* 106
5.5.3 Les inégalités *clique* 106
5.5.4 Les inégalités *generalized clique* 107
5.5.5 Les inégalités *odd-cut* 108
5.5.6 Les inégalités *generalized odd-cut* 111
5.5.7 Les inégalités *odd-set* 117
5.5.8 Les inégalités *even-set* 118
5.5.9 Équivalences entre inégalités de parité 118
5.6 Les inégalités *co-circuit* 122
5.6.1 Les inégalités *co-circuit* 122
5.6.2 Les inégalités *generalized co-circuit* 123
5.7 Les règles de branchement 128
5.7.1 Le branchement *fix max edge flow direction* 128
5.7.2 Le branchement *fix max flow two vertices* 130
5.7.3 Le branchement *parity* 131
5.8 Résultats expérimentaux 134
5.8.1 Les pré-traitements sur les variables 135
5.8.2 Les règles de branchement 137
5.8.3 Les inégalités valides 141

5.8.4 Comparaisons des temps de résolution 142
5.9 Conclusions 144

Chapitre 6 RÉSOLUTION DU MODÈLE L8 : DEUXIÈME APPROCHE 145
6.1 Génération de colonnes 146
6.1.1 Introduction 146
6.1.2 Le modèle L8$^+$ 148
6.1.3 Le dual du modèle L8$^+$ 149
6.1.4 Le problème du *pricing* 150
6.2 Les inégalités de flots 151
6.2.1 Les inégalités *no joint flow* 152
6.2.2 Les inégalités *generalized no joint flow* 152
6.2.3 Les inégalités *one-way* 154
6.2.4 Les inégalités *generalized one-way* 154
6.2.5 Les inégalités *no bridge flow* 155
6.2.6 Les inégalités *generalized no bridge flow* 156
6.2.7 Les inégalités *no same fruit salad** 158
6.2.8 Les inégalités *generalized no same fruit salad** 159
6.2.9 Les inégalités *no-loop* 159
6.3 Résultats expérimentaux : BPC1 163
6.3.1 Les pré-traitements sur les variables 163
6.3.2 Résolution de la relaxation à la racine 165
6.3.3 Les inégalités valides 168
6.3.4 Comparaisons des temps de résolution 171
6.4 Évaluation implicite du dual 174
6.5 Résultats expérimentaux : BPC2 177
6.5.1 Les pré-traitements sur les variables 177
6.5.2 Comparaison entre BPC1 et BPC2 pour le nombre de variables générées 178
6.5.3 Comparaisons des temps de résolution 178
6.6 Conclusions 179

Chapitre 7 CONCLUSIONS 181
7.1 Synthèse des travaux 181
7.2 Évaluation des solutions proposées 183
7.3 Améliorations futures 183

7.3.1 Utilisation de la programmation non linéaire en nombres entiers . . . 183
7.3.2 Utilisation d'algorithmes adaptés pour la résolution des PL 184
7.3.3 Meilleures heuristiques 184
7.3.4 Le modèle L8 185
7.3.5 Programmation par contraintes 186

Références 187

Annexes 192
A.1 Les quinze graphes de référence 192
B.1 Les bornes inférieures des huit modèles sur les quinze graphes de référence . 193
C.1 Équivalence entre les modèles L1, L2 et L3 194
C.1.1 Équivalence entre les modèles L1 et L2 194
C.1.2 Équivalence entre les modèles L2 et L3 197
D.1 Modèles supplémentaires non présentés 201
D.1.1 Quelques modèles non présentés 201
D.1.2 Comparaisons sommaires des modèles non présentés 213

Liste des tableaux

Tableau 0.1 Les différents sigles utilisés dans ce document xxi
Tableau 1.1 Version des logiciels utilisés dans cette thèse 19
Tableau 2.1 Différentes appellations pour le problème de latence minimum. 27
Tableau 4.1 Bornes supérieures sur le nombre de variables et de contraintes 70
Tableau 5.1 Nombres exacts et approchés de variables et de contraintes dans le modèle L8. 91
Tableau 5.2 Nombre de variables et de contraintes pour des graphes complets pour le modèle L8. 92
Tableau 5.3 Nombre de contraintes détaillé par type de contraintes pour des graphes complets. 92
Tableau 5.4 Équivalences entre les inégalités de parité 120
Tableau 5.5 Nombre de nœuds et temps pour *fix max flow two vertices* 137
Tableau 5.6 Nombre de nœuds et temps pour *fix max edge flow direction* 138
Tableau 5.7 Nombre de nœuds pour les trois règles de branchement 140
Tableau 5.8 Temps en secondes pour les trois règles de branchement 140
Tableau 5.9 Nombre de nœuds, temps nécessaires (en secondes) et nombres de coupes de parité utilisées pour 5 graphes de référence. 142
Tableau 5.10 Comparaisons des temps pour les *Branch and Cut* 142
Tableau 6.1 Le modèle L8$^+$ détaillé 150
Tableau 6.2 Le dual du modèle L8$^+$ détaillé 151
Tableau 6.3 Une solution optimale du dual du modèle L8$^+$ pour le graphe introductif 151
Tableau 6.4 Comparaisons des écarts relatifs obtenus avec les modèles L8 et L8$^+$. 163
Tableau 6.5 Comparaisons de différentes stratégie de pré-traitement des variables 165
Tableau 6.6 Comparaisons des temps de résolution pour le premier PL à la racine du modèle L8$^+$ 167
Tableau 6.7 Comparaisons pour chacune des nouvelles familles de coupes 169
Tableau 6.8 Temps de résolution en prenant les x coupes les plus efficaces 170
Tableau 6.9 Nombre de coupes acceptées ou rejetées en prenant les 35 coupes les plus efficaces 171
Tableau 6.10 Temps de résolution avec et sans inégalités *gen. co-circuit* 171
Tableau 6.11 Comparaisons des temps entre BC1 et BPC1 173

Tableau 6.12 Pourcentages des variables générées avec BPC1 et BPC2 178
Tableau 6.13 Comparaisons des temps entre BC1, BPC1 et BPC2 179
Tableau 6.14 Temps en secondes des différents ingrédients pour les *Branch, Price and Cut* 179
Tableau A.1 Caractéristiques principales des quinze graphes de référence 192
Tableau B.1 Les *bi* des huit modèles présentés pour les quinze graphes de référence. 193
Tableau D.1 Nombre de variables et de contraintes pour les modèles de l'annexe . 213
Tableau D.2 Comparaisons des relaxations pour les modèles de l'annexe 214

Liste des figures

FIGURE 1.1 L'exemple introductif 9
FIGURE 1.2 Une solution optimale de l'exemple introductif 9
FIGURE 1.3 La construction d'une solution optimale 9
FIGURE 1.4 Un chemin partiel 10
FIGURE 1.5 Une liaison inter-clients 11
FIGURE 1.6 Un diamètre 11
FIGURE 1.7 La terminologie des solutions 12
FIGURE 1.8 Temps de fin de service d'une arête 12
FIGURE 1.9 Les différents coûts pour l'exemple introductif 13
FIGURE 1.10 Le caractère cumulatif du problème 14
FIGURE 1.11 Un diamètre pour le modèle L1 15
FIGURE 1.12 Modèle L1 : une solution optimale pour l'exemple introductif. 17
FIGURE 3.1 Une arête peut être copiée plusieurs fois dans le graphe augmenté . . 35
FIGURE 3.2 L'augmentation minimale pour rendre un graphe unitaire traversable ne sert à rien pour le PPCC 35
FIGURE 3.3 Comment décomposer le problème en sous-problèmes ? 36
FIGURE 3.4 Transformations simples connues 38
FIGURE 3.5 La solution optimale pour le PPCCAR de l'exemple introductif . . . 38
FIGURE 3.6 Première étape de la transformation du PPCCAR en PLM 39
FIGURE 3.7 Deuxième étape de la transformation du PPCCAR en PLM 39
FIGURE 3.8 Troisième étape de la transformation du PPCCAR en PLM 40
FIGURE 3.9 Numérotation des sommets du cercle 41
FIGURE 3.10 Une bande unitaire 43
FIGURE 3.11 Les deux solutions optimales pour la bande unitaire 43
FIGURE 3.12 L'ancienne conjecture pour la grille unitaire 45
FIGURE 3.13 Le cas de la grille unitaire 2×2 46
FIGURE 3.14 Conjecture pour la grille $2 \times l$ 46
FIGURE 4.1 Modèle L2 : une solution optimale de l'exemple introductif. 51
FIGURE 4.2 Modèle L3 : une solution optimale de l'exemple introductif. 53
FIGURE 4.3 Modèle L4 : le graphe auxiliaire. 54
FIGURE 4.4 Modèle L4 : une solution optimale de l'exemple introductif. 57

FIGURE 4.5 Modèle L5 : le graphe auxiliaire. 57
FIGURE 4.6 Modèle L5 : une solution optimale de l'exemple introductif. 60
FIGURE 4.7 Modèle L6 : une solution optimale de l'exemple introductif. 62
FIGURE 4.8 Modèle L7 : une solution optimale de l'exemple introductif. 64
FIGURE 4.9 Modèle L7 : les variables q_S^k. 65
FIGURE 4.10 Modèle L8 : le graphe auxiliaire éclaté. 65
FIGURE 4.11 Modèle L8 : une solution optimale de l'exemple introductif. 69
FIGURE 4.12 Les écarts relatifs pour les huit modèles. 72
FIGURE 4.13 Positionnements relatifs des *bi* pour les huit modèles 72
FIGURE 4.14 Temps de résolution pour les huit modèles sur quatre graphes 73
FIGURE 4.15 Une solution relaxée optimale pour le modèle L4 78
FIGURE 5.1 Un graphe et les arêtes originales de son graphe auxilaire éclaté . . . 93
FIGURE 5.2 Un exemple où une solution est obtenue comme combili de solutions. 93
FIGURE 5.3 L'heuristique *enumeration* en action 97
FIGURE 5.4 Le pré-traitement *mandatory edge*. 98
FIGURE 5.5 Le pré-traitement *shortest path minimum number of edges*. 99
FIGURE 5.6 Le pré-traitement *reachable edge*. 99
FIGURE 5.7 Le pré-traitement *reachable shortest path*. 100
FIGURE 5.8 Le pré-traitement *depot*. 101
FIGURE 5.9 Le pré-traitement *backward compatibility*. 101
FIGURE 5.10 Le pré-traitement *forward compatibility*. 102
FIGURE 5.11 Une illustration de la correspondance entre l'intérieur et la frontière. . 104
FIGURE 5.12 Une illustration des inégalités (5.4). 106
FIGURE 5.13 Une illustration des inégalités (5.5). 107
FIGURE 5.14 Une illustration des inégalités (5.6) dans le cas où $|C|$ est impair. . . 108
FIGURE 5.15 Les inégalités (5.6) dans le cas où $|C|$ est pair. 109
FIGURE 5.16 Une illustration des inégalités (5.7) 109
FIGURE 5.17 Une illustration des inégalités (5.11) 112
FIGURE 5.18 Une illustration des inégalité (5.17). 117
FIGURE 5.19 Une illustration des inégalités (5.18) 118
FIGURE 5.20 Les inclusions des familles d'inégalités de parité. 121
FIGURE 5.21 Une illustration des inégalités (5.20) 122
FIGURE 5.22 Une illustration des inégalités (5.21) 123
FIGURE 5.23 Une illustration des inégalités (5.22). 124
FIGURE 5.24 Le graphe G' utilisé pour séparer les inégalités *generalized co-circuit* (5.22) 126

FIGURE 5.25 Transformation d'une arête pour séparer exactement les inégalités *generalized co-circuit* . . . 126
FIGURE 5.26 Situation typique pour une arête originale dans une solution relaxée fractionnaire . . . 129
FIGURE 5.27 Un exemple qui ne peut pas être séparé par le branchement *fix max edge flow direction* . . . 130
FIGURE 5.28 Remarque sur la règle de branchement *fix max flow two vertices* . . . 132
FIGURE 5.29 Règle de branchement *parity* . . . 132
FIGURE 5.30 La règle de branchement *parity* n'est pas complète . . . 134
FIGURE 5.31 Une comparaison du nombre de variables éliminées par pré-traitement 135
FIGURE 5.32 Gains en temps avec les pré-traitements . . . 136
FIGURE 5.33 *fix max flow two vertices* l'emporte sur *fix max edge flow direction* . . 141
FIGURE 5.34 Comparaison des temps de résolution pour le BC1 . . . 143
FIGURE 5.35 Répartition du temps pour le BC1 . . . 144
FIGURE 6.1 Une illustration des inégalités (6.1). . . . 152
FIGURE 6.2 Une illustration des inégalités (6.2). . . . 153
FIGURE 6.3 Une illustration des inégalités (6.3). . . . 154
FIGURE 6.4 Une illustration des inégalités (6.4). . . . 155
FIGURE 6.5 Une illustration des inégalités (6.5). . . . 156
FIGURE 6.6 Une illustration de la première version des inégalités (6.6). . . . 157
FIGURE 6.7 Une illustration de la deuxième version des inégalités (6.7). . . . 157
FIGURE 6.8 Une illustration de la troisième version des inégalités (6.8). . . . 158
FIGURE 6.9 Une illustration des inégalités (6.9). . . . 159
FIGURE 6.10 Une illustration des inégalités (6.10). . . . 160
FIGURE 6.11 Un p-circuit. . . . 160
FIGURE 6.12 L'idée des inégalités (6.11). . . . 161
FIGURE 6.13 Une illustration des inégalités (6.11). . . . 162
FIGURE 6.14 Une condition à respecter pour trouver une inégalité (6.11) violée. . . 163
FIGURE 6.15 Comparaison des temps de résolution pour le BPC1 . . . 172
FIGURE 6.16 Répartition du temps pour l'algorithme BPC1 . . . 174
FIGURE 6.17 Mise à jour implicite du dual . . . 176
FIGURE D.1 Modèle Q1 : une solution optimale de l'exemple introductif. . . . 203
FIGURE D.2 Graphe auxiliaire pour le modèle L10. . . . 205
FIGURE D.3 Modèle L10 : une solution optimale de l'exemple introductif. . . . 207
FIGURE D.4 Modèle L11 : une solution optimale de l'exemple introductif. . . . 209

FIGURE D.5 Modèle L13 : une solution optimale de l'exemple introductif. 213

Liste des annexes

Annexe A Les quinze graphes de référence 192

Annexe B Les bornes inférieures des huit modèles sur les quinze graphes de référence 193

Annexe C Équivalence entre les modèles L1, L2 et L3 194

Annexe D Modèles supplémentaires non présentés 201

Liste des sigles et abréviations

Voici, classés par ordre alphabétique dans le tableau 0.1, les différents sigles utilisés dans ce document. Cette nomenclature, qui n'est pas très consistante, reflète bien les divers aléas de la recherche.

Sigle	Sigle en anglais	Définition
PCHC	CHPP	Problème du chemin hamiltonien cumulatif - Cumulative Hamiltonian Path Problem
PLM	MLP	Problème de latence minimum - Minimal Latency Problem
PPC	CPP	Problème du postier chinois - Chinese Postman Problem
PPCA	WPP	Problème du postier chinois asymétrique - Windy postman problem
PPCC	CCPP	Problème du postier chinois cumulatif - Cumulative Chinese Postman Problem
PPCCAR	CCPPWR	Problème du postier chinois cumulatif avec retour au dépôt - Cumulative Chinese Problem with Return to the Depot
PPH	HPP	Problème du postier hiérarchique - Hierarchical Postman Problem
PPR	RPP	Problème du postier rural - Rural Postman Problem
PVC	TSP	Problème du voyageur de commerce - Traveling Salesman Problem
PVCDT	TDTSP	Problème du voyageur de commerce dépendant du temps - Time-Dependent Traveling Salesman Problem

Tableau 0.1 Les différents sigles utilisés dans ce document

Le tableau 2.1 page 27 résume quant à lui les diverses appellations utilisées dans la littérature pour décrire le problème équivalent sur les sommets.

Chapitre 1

INTRODUCTION

1.1 Motivations[1]

Le problème du postier chinois (PPC[2]) est bien connu et résolu dans sa version classique. Petit à petit, des contraintes supplémentaires sont venues s'ajouter pour obtenir une meilleure modélisation du monde réel. Ainsi, nous avons le problème du postier chinois asymétrique (PPCA) où le coût de traversée de l'arête diffère suivant le sens du parcours, le problème du postier rural (PPR) où seulement une partie des arêtes doivent être desservies et puis finalement le problème du postier hiérarchique (PPH) où des classes d'arêtes sont desservies prioritairement à d'autres (voir (Eiselt *et al.*, 1995a,b) ou (Dror, 2000) pour une vue d'ensemble).

Bien que ces problèmes posent encore de fameux défis, dans le cadre de cette thèse, nous nous tournons vers une autre modification de la version classique[3] du problème du postier chinois. Nous n'ajoutons aucune contrainte mais nous modifions la fonction objectif en la rendant cumulative, ce qui donne un problème nouveau et inconnu : le problème du postier chinois cumulatif (PPCC). Jusqu'à ce jour et à notre connaissance aucune étude n'a été faite sur ce problème même si le problème équivalent sur les sommets, le problème de latence minimum[4] (PLM), est étudié surtout depuis les années 1990 (voir (Afrati *et al.*, 1986), (Raghavachari et Veerasamy, 1999), (Bianco *et al.*, 1993), (Minieka, 1989), (Goemans et Kleinberg, 1998) ou (Blum *et al.*, 1994) par exemple).

1. La présentation qui suit reprend en partie celle qui a été faite dans notre mémoire de maîtrise (van Omme, 2003).

2. Les différents sigles utilisés sont rassemblés à la page xxi.

3. En fait, nous définissons un problème qui est en rupture avec la tradition des problèmes de postier chinois. Les différences et leurs motivations sont énoncées dans ce chapitre et le chapitre 3 concernant les caractéristiques du problème.

4. Ce problème est connu sous différents vocables dans la littérature. Nous en reprenons quelques uns dans la section 2.2.2 page 26.

1.1.1 Le problème initial

Le problème initial qui motive l'étude de cette version du problème du postier chinois est un problème de tournée sur les arêtes où le temps de service des arêtes a une importance cruciale. Citons deux exemples d'application.

Le premier exemple traite du déneigement d'une artère. Plus on attend pour déblayer une rue ou une route et plus la circulation sur cette artère sera pénible voire impossible pour les usagers.

Le deuxième exemple concerne des différents traitements des rues d'une ville par la voirie. Le coût moyen pour réparer les nids de poules est moindre que le coût exorbitant pour le resurfaçage[5]. Les rues qui nécessitent un traitement léger demanderont peut-être des soins plus importants si leurs fissures se sont entre-temps transformées en nids de poules et ces derniers en problèmes exigeant un resurfaçage.

Le moment de traitement de l'arête est donc important dans ces deux exemples.

Dès lors, dans le problème du postier chinois cumulatif (PPCC), nous voulons minimiser les temps de service[6] de toutes les arêtes, c'est-à-dire le temps total d'attente depuis le début de la tournée jusqu'au traitement complet de l'arête pour toutes les arêtes. Notre choix de fonction objectif est de sommer ces temps de service pour toutes les arêtes, ce qui introduit un caractère cumulatif dans la fonction objectif comme nous le verrons plus loin.

Dans une approche plus réaliste, nous pourrions associer différents coûts pour le passage et le traitement des arêtes.

Pour le déneigement, nous pourrions envisager un coût différent pour le passage d'une rue déneigée (coût le plus faible) ou non (coût plus élevé) et ce passage prendrait moins de temps que le traitement effectif de cette rue (coût le plus élevé).

Dans le cas du traitement des rues d'une ville par la voirie, nous pourrions aussi imaginer des coûts différents suivant le service d'une rue, le passage sur une rue traitée ou non bien

5. Le resurfaçage consiste à enlever puis à remplacer la surface de roulement en asphalte ou en béton bitumeux.

6. Ce temps de service est aussi appelé la *latence* de l'arête ou encore son *coût cumulatif*.

que la nécessité de traiter différemment le passage sur une rue traitée ou non peut sembler plus artificiel que dans le premier exemple.

Ce problème initial, où trois coûts différents sont associés à une arête (un coût pour le service, un coût pour le passage lorsque l'arête a été traitée et un coût pour le passage sur l'arête non traitée) nous semble difficile à aborder dans une première étude. Nous nous limitons dans un premier temps à une version avec seulement un coût par arête.

Dans une première approche et par souci de simplicité, nous nous intéressons au problème non orienté avec un seul véhicule sans capacité pour effectuer la tournée. Nous considérons aussi d'abord des problèmes où le temps de passage est le même que celui du service. Cette hypothèse correspond avec notre souci de privilégier le temps de fin de service global. Nous réfutons[7] ainsi la possibilité de pouvoir passer sur une arête sans la servir puisque le passage est aussi coûteux que le service.

1.1.2 Clients versus Serveur

Notre motivation n'était pas tant de modifier la fonction objectif que d'appréhender le problème du postier chinois sous un nouvel angle : celui des clients. En effet, jusqu'à présent seul le point de vue du fournisseur était pris en compte. Dans le problème du postier chinois (PPC), seule compte l'obtention d'une tournée où l'on dessert chaque arête. Peu importe l'instant où une arête sera desservie dans la tournée pourvu que globalement cette tournée se fasse pour un coût minimal. Ainsi on peut dire que le PPC ne s'occupe que de l'optimisation du côté serveur.

Le PPCC est un bon compromis entre les attentes des clients et ce que peut offrir le fournisseur. Les clients qui peuvent être desservis le plus rapidement, toujours dans un souci de remédier aux situations avant qu'elles ne dégénèrent, auront plus de chance d'être desservis avant les autres. Mais il serait ridicule de vouloir servir à tout prix un client peu coûteux mais très difficile à atteindre. Le coût d'une telle opération se répercuterait sur l'ensemble des clients de manière inutile. Dans le PPCC, les petits clients sont privilégiés mais pas au détriment de l'ensemble des clients. Tout dépendra de la configuration du graphe ainsi que des coûts des arêtes. Comme le coût de service ou de passage est exactement le même, les arêtes les plus proches du dépôt seront desservies d'abord. Ceci évitera au fournisseur de

7. Voir le lemme 2 page 6.

devoir parcourir de grandes distances inutilement pour servir un client peu coûteux mais lointain. Il s'agit d'un compromis entre les considérations des clients et celles du fournisseur.

1.1.3 Le temps : un facteur crucial

Reprenons l'exemple du traitement des rues. Comment s'assurer de forcer le traitement des petits problèmes avant les gros qui sont si gourmands en temps ? Comme le PPCC s'occupe prioritairement du service des arêtes dont le traitement est peu coûteux, la solution est simple : mettre un coût peu élevé lorsque le traitement de l'arête est rapide et beaucoup plus élevé si le service de l'arrête se révèle être long.

Dans le cas de traitement des rues par la voirie, on pourrait imaginer trois coûts (on choisit un de ces coûts pour chacune des arêtes) possibles pour une arête : un petit coût pour le scellement de fissures, un coût moyen pour réparer les nids de poules et puis un coût exorbitant pour le resurfaçage. C'est la même équipe qui s'occupe de tout. Si l'équipe s'occupe d'abord de resurfacer des rues, les citoyens attendront longtemps avant que de petits travaux de réparation soient effectués. Par contre si les travaux de réparations sont entrepris en premier, ils ne prendront pas tellement de temps et cette petite attente supplémentaire ne retardera pas tellement les gros travaux qui de toute façon prendront l'essentiel du temps de la tournée.

Voici un autre exemple où le temps cumulatif de traitement des arêtes est crucial. Prenons l'exemple de l'exploration d'un terrain inconnu par un robot. Quadrillons le secteur à découvrir par un maillage d'arêtes. Le coût de celles-ci sera fixé *a priori* par la connaissance sommaire du terrain que nous avons. Nous pensons à l'exploration de la planète Mars par un robot par exemple. Une arête passant par un cratère ou une montagne aura un coût supérieur à celles en terrain plat. Notre désir est de pouvoir inspecter à l'aide du robot la plus grande partie du terrain le plus vite possible parce que nous craignons que les conditions de transmission soient si mauvaises que nous perdions le contrôle du robot assez vite. Si le robot suit une solution optimale du PPCC, il parcourra d'abord un maximum de petites arêtes facilement accessibles depuis son point de départ. Il accédera aux endroits difficiles seulement par après, s'il est encore fonctionnel.

1.2 Définitions et concepts de base

1.2.1 Définition du problème du postier chinois cumulatif

Soit $G = (\mathrm{V},\mathrm{E})$ un graphe connexe non orienté où V est l'ensemble des n sommets et E l'ensemble des m arêtes du graphe. Soit une fonction c de coûts sur les arêtes $c : \mathrm{E} \to \mathbb{N}_0$. Nous ne considérons que des coûts entiers positifs[8]. Rappelons que notre motivation première est de modéliser le temps de service sur les arêtes.

Un sommet privilégié est choisi dès le départ, c'est le sommet à partir duquel la tournée est effectuée. Un tel sommet sera dénommé *dépôt* dans la suite de ce document.

Le coût d'une arête au sens cumulatif[9] dépend du chemin parcouru depuis le début de la tournée. Pour calculer le coût cumulatif d'une arête a_i traitée en i^e position on additionne le coût cumulatif de l'arête a_{i-1} traitée précédemment avec le coût pour aller de a_{i-1} à a_i plus le coût $c(a_i)$ de a_i :

$$\text{coût cumulatif}(a_i) = \text{coût cumulatif}(a_{i-1}) + \text{coût}(a_{i-1},a_i) + c(a_i).$$

Le coût cumulatif de la première arête se confond avec son coût. À tout chemin commençant par le dépôt et passant par toutes les m arêtes a_i du graphe dans l'ordre $i_1,i_2,\ldots,i_m$, la valeur de la fonction objectif[10] z est la somme de tous les coûts cumulatifs pour toutes les arêtes :

$$z = \text{coût cumulatif}(a_1) + \text{coût cumulatif}(a_2) + \ldots + \text{coût cumulatif}(a_m).$$

De sorte que

$$\begin{array}{rcl|l} z & = & c(a_1)+ & \text{coût cumulatif}(a_1) \\ & & c(a_1) + \text{coût}(a_1,a_2) + c(a_2)+ & \text{coût cumulatif}(a_2) \\ & & c(a_1) + \text{coût}(a_1,a_2) + c(a_2) + \text{coût}(a_2,a_3) + c(a_3)+ & \text{coût cumulatif}(a_3) \\ & & \ldots & \ldots \end{array}$$

ou encore

8. Nous réservons un coût nul sur certaines arêtes pour les différentes transformations du problème.
9. Voir la section 1.4.4 page 12 pour plus de détails.
10. Nous avons remplacé les indices $i_1,i_2,\ldots,i_m$ par $1,2,\ldots,m$ pour simplifier l'écriture.

$$
\begin{aligned}
z \;=\; & m \cdot c(a_1) + \\
& (m-1) \cdot \text{coût}(a_1,a_2) + (m-1) \cdot c(a_2) + \\
& (m-2) \cdot \text{coût}(a_2,a_3) + (m-2) \cdot c(a_3) + \\
& \vdots \\
& (m-i) \cdot \text{coût}(a_i,a_{i+1}) + (m-i) \cdot c(a_{i+1}) + \\
& \vdots \\
& (1) \cdot \text{coût}(a_{m-1},a_m) + (1) \cdot c(a_m)
\end{aligned}
$$

Ce qu'on peut résumer en

$$z = m \cdot c(a_1) + \Sigma_{i=1}^{m-1}(m-i) \cdot \{\text{coût}(a_i,a_{i+1}) + c(a_{i+1})\}.$$

Définition 1 (PPCC) *Le problème du postier chinois cumulatif consiste à trouver un chemin qui, partant d'un sommet privilégié (le dépôt), desserve toutes les arêtes du graphe et ce pour un coût minimal au sens cumulatif, c'est-à-dire qu'on minimise la somme des coûts cumulatif de toutes les arêtes du graphe.*

Nous rappelons que nous n'exigeons pas le retour au dépôt après que la dernière arête ait été desservie[11].

1.2.2 Conséquences de la définition

Il y a deux conséquences directes de cette définition. La première conséquence est que, comme les coûts pour desservir et visiter une arête sont les mêmes, il y a tout avantage à d'abord servir une arête avant de l'utiliser. Pour s'en convaincre, la lectrice[12] n'a qu'à comparer le coût de deux solutions réalisables en tout point identiques sauf que dans la première elle utilise une arête avant de la desservir et dans la seconde elle sert cette arête avant de faire le reste de la tournée. La seconde solution est évidemment moins chère. La deuxième conséquence de la définition est qu'il est plus avantageux d'utiliser des plus courts chemins entre deux arêtes à desservir consécutivement. Pour s'en convaincre, il suffit de penser au début d'une tournée : celle-ci sera moins coûteuse si l'on prend effectivement des plus courts chemins. Ces deux propriétés seront abondamment utilisées par la suite et pour pouvoir les référer aisément, nous en faisons deux lemmes.

11. Ou de manière équivalente, nous considérons que le retour au dépôt a un coût nul.
12. Ici et plus loin, nous utilisons le féminin pour décrire un lecteur ou une lectrice potentiels.

Lemme 2 (Desservir avant d'utiliser) *Dans toute solution optimale, il faut desservir une arête avant de pouvoir l'utiliser.*

Lemme 3 (Utilisation de plus courts chemins) *Dans toute solution optimale, pour desservir une arête* (c,d) *dans le sens de* c *vers* d *depuis une arête* (a,b) *qui a été desservie dans le sens de* a *vers* b*, il faut utiliser un plus court chemin entre les sommets* b *et* c*.*

Grâce au lemme 2, nous savons que le plus court chemin de b à c ne sera constitué que d'arêtes déjà desservie auparavant dans toute solution optimale.

1.2.3 Justification de la définition choisie

Nous l'avons déjà dit, notre définition du problème du postier chinois cumulatif est en rupture avec les définitions classiques des variantes du problème de postier chinois existantes. Dans cette section, nous montrons pourquoi notre définition n'est en rien restrictive.

Le retour au dépôt

Nous n'exigeons pas de retour au dépôt une fois la tournée, c'est-à-dire le traitement de toutes les arêtes, complétée. Cette façon de faire n'est en rien restrictive.

Tout d'abord, on peut transformer facilement ce problème de tournée ouverte en un problème de tournée fermée. Il suffit d'ajouter une arête de coût suffisamment grand pendante au dépôt. Le traitement de celle-ci se fera en dernier vu son coût astronomique et ainsi réalisera le retour au dépôt. Calculer ainsi le coût du retour du véhicule au dépôt semble prohibitif à moins d'accorder autant d'importance au temps de retour du véhicule qu'aux temps de traitement des clients.

Ensuite, dans le cas où le véhicule aurait une capacité limitée qui le forcerait à rentrer au dépôt de temps en temps, notre définition reste d'actualité. Le coût de traitement de la première arête après le retour au dépôt comptabiliserait le retour au dépôt.

Finalement, le retour au dépôt exigé dans le PLM ne fait pas la distinction entre le dépôt et les autres sommets. Cette distinction est plus facile à faire dans notre cas où les clients sont représentés par des arêtes et le dépôt par un sommet. Nous voulons vraiment distinguer les deux.

Le choix du dépôt

Dans le PPCC, nous donnons un choix de dépôt comme partie intégrante du problème, *i.e.* le choix du dépôt est fixé et une solution optimale débutera forcément par ce dépôt[13]. Ceci n'est clairement pas restrictif du point de vue pratique car souvent les dépôts existent et il n'est pas question de les relocaliser ailleurs. D'un point de vue théorique, cette définition n'est pas restrictive non plus puisque si le choix du dépôt correspond à une réalité du problème, il suffit de créer un dépôt fictif reliant tous les dépôts possibles par une arête de coût nul pour trouver le meilleur dépôt possible. Une solution optimale du PPCC sur le graphe avec ce dépôt fictif parcourra d'abord toutes les arêtes de coût nul avant de véritablement commencer la tournée. Le premier sommet de la première arête non fictive desservie sera le meilleur choix de dépôt possible.

Bref, d'un point de vue théorique, le choix du dépôt ainsi que le retour au dépôt pour un coût nul n'est en rien restrictif. Nous pouvons aisément trouver le meilleur dépôt possible ou forcer le retour au dépôt. D'un point de vue pratique par contre, cette façon de faire permet de mieux coller à la réalité de l'existence d'un dépôt.

1.3 Un exemple introductif

La figure 1.1 représente un graphe avec trois arêtes a,b et c de coûts[14] respectifs 1, 50 et 1. Une solution optimale est représentée sur la figure 1.2. La tournée commence par l'arête c puis rebrousse chemin sur cette même arête avant de desservir les arêtes a et b. On parlera de la solution $c - c - a - b$ ou, si on ne tient compte que de l'ordre dans lequel les arêtes sont desservies, de la solution $c - a - b$. Cette solution a un coût de

$$\begin{array}{rcl|l} z & = & 1+ & \text{coût cumulatif}(c) \\ & & 1+1+1+ & \text{coût cumulatif}(a) \\ & & 1+1+1+0+50 & \text{coût cumulatif}(b) \\ & = & 57 & \end{array}$$

13. Le dépôt est aussi fixé à l'avance dans le problème de latence minimum.
14. On peut penser à une durée de service sur ces arêtes.

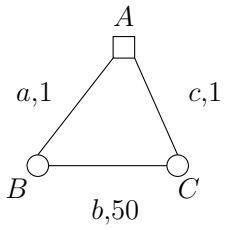


FIGURE 1.1 Un exemple introductif. Cet exemple de graphe tout simple nous permet déjà de comprendre la complexité du PPCC. Le dépôt se situe en haut et est représenté par le carré. Les valeurs après la virgule correspondent au coût pour servir et passer sur l'arête correspondante.

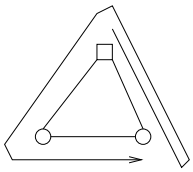


FIGURE 1.2 Une solution optimale de l'exemple introductif. Une solution optimale de coût $1+3+53=57$ pour le PPCC de la figure 1.1.

1.4 Terminologie

Dans la suite de ce document, nous utiliserons quelques concepts clefs que nous définissons ici. Principalement, il s'agit de décortiquer la construction d'une solution optimale étape par étape. Les modèles présentés au chapitre 4 ainsi que ceux présentés dans l'annexe D page 201 reposent tous sur cette décomposition d'une solution optimale.

Reprenons notre exemple introductif de la page 8 dont la figure 1.3 représente la construction d'une solution optimale.

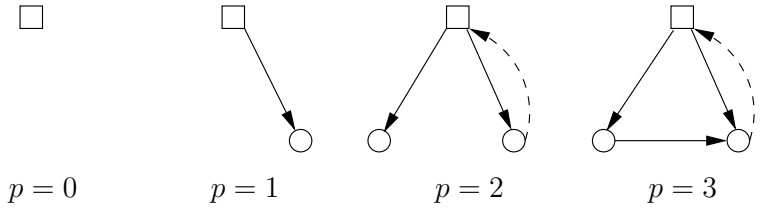


FIGURE 1.3 La construction, étape par étape, de la solution optimale de la figure 1.2.

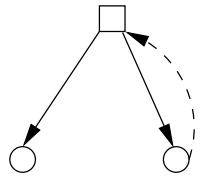
Cette solution optimale est construite en autant d'étapes qu'il y a d'arêtes plus une. Un indice p parcourra les étapes 0 à m. Parce que le dépôt est une donnée *a priori* du problème, une étape $p = 0$ permettra d'initialiser la solution, *i.e.* de faire partir la solution à partir du dépôt donné.

1.4.1 Les chemins partiels

Découpons et définissons les différentes parties de la construction d'une solution optimale. Tout d'abord arrêtons-nous à une étape p donnée, c'est-à-dire juste après qu'une arête soit nouvellement desservie. Appelons *chemin partiel*[15] la partie de la tournée jusqu'à l'arête nouvellement desservie.

Définition 4 *Un* chemin partiel *est une partie de la tournée du dépôt jusqu'à et y compris une arête nouvellement desservie.*

La figure 1.4 représente un chemin partiel pour notre exemple introductif.



$p = 2$

FIGURE 1.4 Le chemin partiel de l'étape $p = 2$ pour la solution optimale de la figure 1.2.

1.4.2 Les liaisons inter-clients

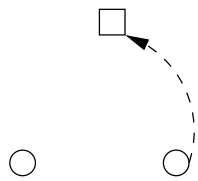
Nous appellerons les arêtes parcourues entre deux arêtes desservies l'une après l'autre, des *liaisons inter-clients*.

Définition 5 *Une* liaison inter-clients *entre deux arêtes desservies successivement est la partie de la tournée comprise entre ces deux arêtes.*

Le lemme 3 nous permet de déduire que dans toute solution optimale, une liaison inter-clients est constituée d'un plus court chemin.

15. A ne pas confondre avec le concept de *solution partielle* qui sera introduit au chapitre 4 pour les modèles L4 et L8.

La figure 1.5 représente la *liaison inter-clients* à l'étape $p = 2$ pour notre solution optimale de l'exemple introductif.



$p = 2$

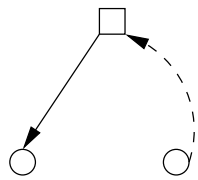
FIGURE 1.5 La liaison inter-clients de l'étape $p = 2$ pour la solution optimale de la figure 1.2.

1.4.3 Les diamètres

Soit une liaison inter-clients. Si nous lui rajoutons l'arête nouvellement desservie, nous obtenons alors ce que nous appelons *un diamètre*.

Définition 6 *Un* diamètre *d'un chemin partiel à une étape p est la partie de la tournée qui raccorde le dernier sommet du chemin partiel à l'étape $p - 1$ au dernier sommet du chemin partiel de l'étape p.*

La figure 1.6 représente le *diamètre* du chemin partiel à l'étape $p = 2$ pour notre solution optimale de l'exemple introductif.



$p = 2$

FIGURE 1.6 Le diamètre de l'étape $p = 2$ pour la solution optimale de la figure 1.2.

La figure 1.7 représente les trois concepts, *chemin partiel*, *liaison inter-clients* et *diamètre*, ensemble ainsi que les liens entre eux.

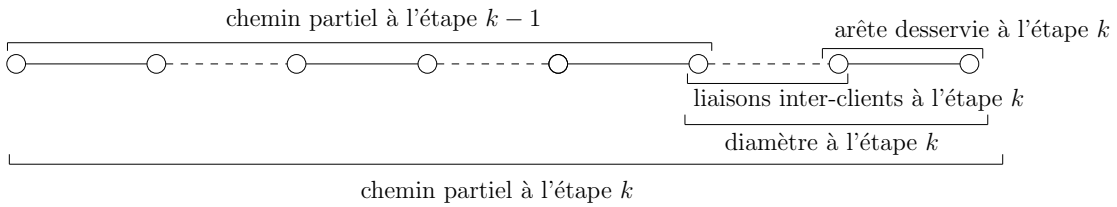


FIGURE 1.7 La terminologie des solutions. On peut voir une solution réalisable comme une alternance d'arêtes desservies et de liaisons inter-clients. Dans cet exemple-ci, $k = 4$.

1.4.4 Coûts relatifs et cumulatifs

Soit une tournée passant par toutes les arêtes et traitant celles-ci dans l'ordre a_1,a_2, . . . ,a_{i-1},a_i,a_{i+1}, . . . ,a_n. Comme le décrit la figure 1.8, le temps de fin de service de l'arête a_i est égal au temps de fin de service de l'arête a_{i-1} plus la durée pour aller de a_{i-1} à a_i plus encore le temps de service de l'arête a_i. Pour la première arête il s'agit de son coût initial. Ce temps de fin de service d'une arête est ce que nous appellerons le *coût cumulatif* d'une arête. Nous utiliserons aussi deux autres synonymes pour le coût cumulatif : *coût réel* et *coût effectif*. En anglais, on parle de « latency »de l'arête, terme que nous reprenons par « latence »en français.

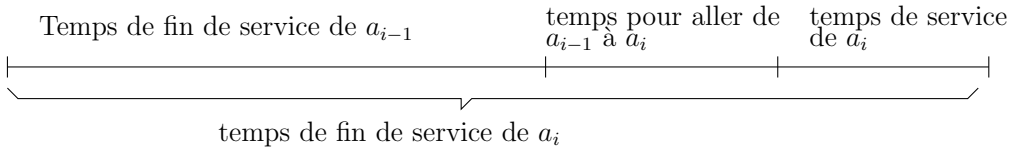


FIGURE 1.8 Temps de fin de service d'une arête. Temps de fin de service de l'arête a_i = temps de fin de service de l'arête a_{i-1} + durée pour aller de a_{i-1} à a_i + service de a_i.

Le coût cumulatif d'une arête devient dynamique dans le sens où il dépend du chemin suivi pour atteindre l'arête. Deux chemins différents aboutissant à la même arête donneront sans doute des coûts cumulatifs différents pour cette arête. En fait, le coût cumulatif ou latence d'une arête n'est rien d'autre que le coût d'un chemin partiel défini par cette arête, c'est-à-dire le coût du plus court chemin pour atteindre cette arête auquel nous additionnons le coût de l'arête.

Le *coût relatif* d'une arête quant à lui est simplement la différence entre le coût cumulatif de cette arête et le coût cumulatif de l'arête desservie précédemment. Pour la première arête

desservie le coût relatif se confond avec son coût. L'idée du coût relatif est de comptabiliser ce que la nouvelle arête coûte vraiment relativement à l'arête desservie précédemment. Si nous utilisons notre terminologie, le coût relatif d'une arête n'est rien d'autre que le coût du diamètre correspondant.

La figure 1.9 résume les différents coûts pour l'exemple introductif de la section 1.3 page 8.

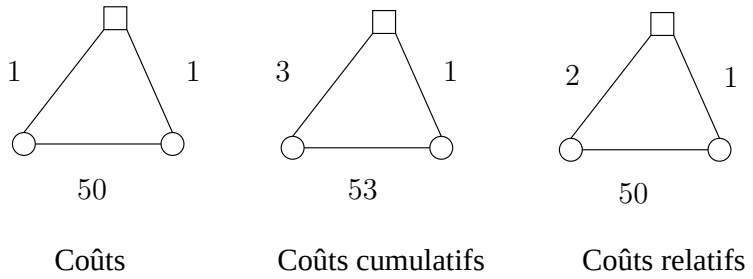


FIGURE 1.9 Les différents coûts pour l'exemple introductif. Les coûts cumulatifs ou latence des arêtes correspondent aux temps de service depuis le début de la tournée alors que les coûts relatifs des arêtes correspondent aux coûts des diamètres.

1.4.5 Une fonction objectif cumulative ?

Pourquoi parle-t-on de coût *cumulatif* et du problème du postier chinois *cumulatif* ? Dans le problème qui nous préoccupe, nous additionnons tous les temps de fin de service et nous essayons de minimiser cette somme. L'addition des fins de temps de service introduit un caractère cumulatif dans la fonction objectif. En effet, le coût des arêtes est additionné plusieurs fois comme le montre la figure 1.10. Le coût cumulatif d'une arête résulte de l'addition répétée des coûts des arêtes desservies précédemment.

1.4.6 Une solution modèle

Rappelons que les deux lemmes 2 et 3 nous permettent de ne considérer que des solutions qui soient telles que

1. une arête est desservie avant d'être parcourue et
2. chaque liaison inter-clients est constituée d'un plus court chemin entre les arêtes desservies entre deux étapes consécutives et ces plus courts chemins eux-mêmes sont constitués uniquement par des arêtes déjà desservies.

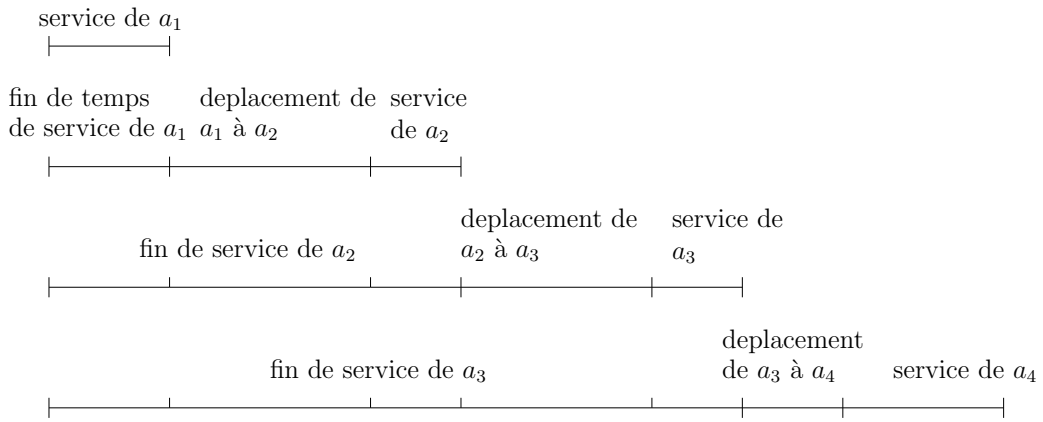


FIGURE 1.10 Le caractère cumulatif du problème. L'aspect cumulatif provient de l'accumulation des coûts. Par exemple le coût associé au service de la première arête se retrouve compté m fois dans la valeur de la fonction objectif.

Nous appelons de telles solutions des solutions *modèles*. La plupart du temps, les modèles présentés au chapitre 4 ne considèrent que ce type de solutions.

1.5 Le modèle L1

Un premier modèle qui vient très naturellement à l'esprit est le modèle L1 proposé dans notre mémoire de maîtrise[16] (voir van Omme, 2003). Il s'agit d'un programme linéaire en variables binaires qui indiquent si oui ou non des diamètres font partie de la solution optimale.

Nous modélisons la construction d'une solution en exigeant qu'à chaque étape un nouveau diamètre soit construit. Pour faire cela, nous utilisons quatre familles de variables binaires : deux pour les arêtes et deux pour les sommets. Ce sont des variables indicatrices qui nous disent si une arête ou un sommet font partie ou non d'un diamètre à l'étape k.

1.5.1 Familles de variables pour les arêtes

$$v_e^k = \begin{cases} 1 & \text{si l'arête } e \text{ est desservie à l'étape } k. \\ 0 & \text{sinon.} \end{cases}$$

16. La version proposée ici est légèrement différente.

$$d_e^k = \begin{cases} 1 & \text{si l'arête } e \text{ est utilisée dans le diamètre à l'étape } k. \\ 0 & \text{sinon.} \end{cases}$$

1.5.2 Familles de variables pour les sommets

$$V_s^k = \begin{cases} 1 & \text{si le sommet } s \text{ est le dernier sommet du diamètre à l'étape } k. \\ 0 & \text{sinon.} \end{cases}$$

$$D_s^k = \begin{cases} 1 & \text{si le diamètre traverse le sommet } s \text{ (} s \text{ n'est ni le premier} \\ & \text{ni le dernier sommet du diamètre) à l'étape } k. \\ 0 & \text{sinon.} \end{cases}$$

Les variables D_s^k sont des variables dépendantes.

La figure 1.11 illustre la construction d'un diamètre à l'étape k.

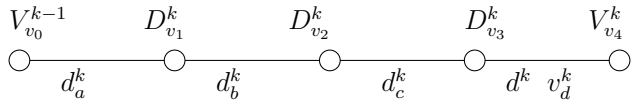


FIGURE 1.11 Un diamètre pour le modèle L1. Il est constitué d'un plus court chemin entre v_0 et v_3 suivi de la nouvelle arête (v_3,v_4) desservie. Toutes les variables représentées sont à 1.

1.5.3 Le modèle

$$\min_{v_e^k, d_e^k, V_s^k, D_s^k} \sum_{e \in \mathrm{E}} c(e) \left[\sum_{k=1}^{m} (m-k+1) \cdot d_e^k \right]$$

s.à :

$$\sum_{k=1}^{m} v_e^k = 1 \qquad \forall\, e \in \mathrm{E} \tag{1.1}$$

$$\sum_{e \in \mathrm{E}} v_e^k = 1 \qquad \forall\, k \in \{1, \ldots, m\} \tag{1.2}$$

$$v_e^k \leq d_e^k \qquad \forall\, e \in \mathrm{E}, \forall\, k \in \{1, \ldots, m\} \tag{1.3}$$

$$\sum_{e; s \in e} d_e^k = 2D_s^k + V_s^{k-1} + V_s^k \qquad \forall\, s \in \mathrm{V}, \forall\, k \in \{1, \ldots, m\} \tag{1.4}$$

$$\sum_{s \in \mathrm{V}} V_s^k = 1 \qquad \forall\, k \in \{1, \ldots, m\} \tag{1.5}$$

$$V_A^0 = 1 \qquad \text{Le dépôt } A \tag{1.6}$$

$$V_s^0 = 0 \qquad \forall\, s \in \mathrm{V} \setminus \{A\} \tag{1.7}$$

(1.1) indique que chaque arête est desservie exactement une fois alors que (1.2) précise qu'une arête est desservie à l'étape k. (1.3) impose qu'on ne peut utiliser l'arête e dans un diamètre que si elle a déjà été desservie. (1.4) fait le raccord entre les trois familles de variables pour construire un diamètre à l'étape k. Soit le sommet $s \in \mathrm{V}$ est l'extrémité d'un diamètre. Dans ce cas, soit $V_s^{k-1} = 1$ ou $V_s^k = 1$ et il n'y a qu'une seule arête e adjacente à s qui appartient à ce diamètre : $\sum_{e; s \in e} d_e^k = 1$. Soit le sommet s est un sommet intérieur au diamètre. Dès lors, $D_s^k = 1$ et il y a deux arêtes pendantes à s dans ce diamètre : $\sum_{e; s \in e} d_e^k = 2$. Soit finalement, le sommet s ne fait pas partie d'un diamètre à la période k et les deux membres de l'égalités sont nuls. (1.5) contrôle qu'il n'y ait qu'un seul sommet pour terminer un diamètre à l'étape k. La tournée commence en un sommet privilégié (le dépôt), ceci est indiqué par (1.6) et (1.7).

1.5.4 La résolution de l'exemple introductif

CPLEX[17] trouve la solution optimale de la figure 1.12.

17. Le tableau 1.1 page 19 reprend l'ensemble des versions des logiciels utilisés dans cette thèse.

k	diamètre (d)	dernier sommet (V)	arête visitée (v)	Sommet traversé(D)	Diamètre
0	/	A $(V_A^0 = 1)$	/	/	A
1	a $(d_a^1 = 1)$	B $(V_B^1 = 1)$	a $(v_a^1 = 1)$	/	A a B
2	a c $(d_a^2 = 1)$ $(d_c^2 = 1)$	C $(V_C^2 = 1)$	c $(v_c^2 = 1)$	A $(D_A^2 = 1)$	A a c B C
3	b $(d_b^3 = 1)$	B $(V_B^3 = 1)$	b $(v_b^3 = 1)$	/	B b C

FIGURE 1.12 Une solution optimale du modèle L1 pour l'exemple introductif. Toutes les variables non nulles sont affichées.

1.6 Choix des logiciels et déroulement des tests expérimentaux

Nous avons essayé tant que possible d'utiliser des logiciels gratuits pour une utilisation académique et dont le code source est disponible. Nous avons choisi le système de résolution (*framework*) SCIP (Achterberg, 2009) développé activement au département d'optimisation du *Konrad-Zuse-Zentrum für Informationstechnik Berlin* à Berlin. Il est un des logiciels non commerciaux les plus rapides sur le marché[18]. Comme l'indique son site officiel `scip.zib.de`, SCIP est une plateforme pour développer à la fois un *branch-cut-and-price*[19] et faire de la programmation par contraintes.

SCIP, bien que supportant 7 solveurs de PL différents[20], est surtout couplé avec le solveur de PL SoPlex, lui aussi développé au département d'optimisation du *Konrad-Zuse-Zentrum für Informationstechnik Berlin* à Berlin. SoPlex est aussi gratuit pour une utilisation académique et son code source est aussi disponible. S'il n'est pas très rapide[21], SoPlex s'est révélé être d'une stabilité à toute épreuve et pour cette raison nous le considérons comme notre solveur de référence. De temps en temps, nous utilisons aussi le solveur CPLEX pour résoudre nos PL afin de pouvoir comparer de façon plus équitable la performance de nos algorithmes avec les résultats obtenus par CPLEX lorsqu'il résout le modèle L8 et ses variantes.

Tout le code a été développé en C++ et compilé par g++ sous Ubuntu en 64 bits. Tous les tests expérimentaux ont été fait sur la même machine : un ordinateur portable Sony VAIO VGN-FW160D doté de deux processeurs Intel Core2 Duo P8400 à 2,26 Ghz avec 3,8 GB de RAM sous le noyeau 2.6.28-19-generic de linux (Ubuntu 9.04 - jaunty). Nous avons forcé tous les logiciels à n'utiliser qu'un seul processeur à la fois car SCIP ne fonctionne pas en parallèle[22].

Les modèles testés dans le chapitre 4 ont été écrits en GNU MathProg Language[23], puis transformés en fichier CPLEX LP avec l'option `--wcpxlp` du solveur GLPSOL et enfin, ces

18. Voir les études comparatives de Hans Mittelmann sur le site `plato.asu.edu/ftp/milpf.html`.
19. Nous verrons par la suite que nous avons dû adapter SCIP pour notre approche de *Branch, Price and Cut*. Voir l'introduction du chapitre 6 page 145 pour plus de détails.
20. CPLEX, Gurobi (en version beta), XPress-MP, Mosek, SoPlex, QSopt et CLP.
21. Voir les études comparatives de Hans Mittelmann sur le site `plato.asu.edu/ftp/lpfree.html`.
22. Voir la remarque à ce sujet au bas de la page 142.
23. *GNU MathProg Language* (GMPL) est un sous-ensemble du langage AMPL. Voir `www.gnu.org/software/glpk/` ou `www.im.pwr.wroc.pl/~przemko/opty/gmpl.html` par exemple.

fichiers ont été traités par appel des différents solveurs.

Le tableau 1.1 reprend les différentes versions des logiciels utilisés dans cette thèse.

Tableau 1.1 Version des logiciels utilisés dans cette thèse.

Logiciels	Versions
CPLEX	12.2.0.0
GLPSOL	4.29
SCIP	1.2.0
SoPlex	1.4.2
CLP	1.12
g++	4.3.3
Boost	1.43.0

Mentionnons que toutes les idées présentées dans cette thèse ont été implémentées à moins d'une mention explicite contraire.

1.7 Objectifs de recherche

Les objectifs de la recherche sont de concevoir un algorithme efficace pour résoudre exactement le PPCC. Dans le cadre de cette thèse, nous cherchons à résoudre un modèle linéaire en nombres entiers en utilisant les outils classiques de la programmation linéaire en nombres entiers.

Nous considérons les solveurs de programmes linéaires (PL) comme des boîtes noires auxquelles nous ne touchons pas.

1.8 Plan de la thèse

Nous présentons tout d'abord l'état de l'art concernant le PPC et le PLM dans le chapitre 2 sur la revue de littérature. Dans le chapitre 3, nous détaillons certaines propriétés intrinsèques du PPCC. Dans le chapitre 4, nous présentons nos huit meilleurs modèles et les comparons entre eux aussi bien expérimentalement que théoriquement. Notre modèle le plus abouti est le modèle L8 que nous utilisons par la suite dans une approche de séparation et évaluation progressive (*Branch and Cut*) au chapitre 5. Au chapitre 6, nous modifions

quelque peu ce modèle pour obtenir le modèle L8$^+$ que nous solutionnons cette fois-ci avec deux approches de génération de colonnes. Finalement, le chapitre 7 nous permet de conclure.

Chapitre 2

REVUE DE LITTÉRATURE

Il n'existe, à notre connaissance, aucun article sur le sujet. Le PPCC est un sujet complètement vierge d'investigation.

Alors que le problème équivalent sur les sommets est le sujet d'étude de bien des articles surtout depuis les années 1990 (voir la section 2.2), on peut se demander pourquoi le cas du postier chinois n'a pas suscité d'intérêt. Nous n'avons pas de réponse définitive à cette question. On peut se demander si le prestige dont jouit le problème du voyageur de commerce (PVC) n'a pas tendance à éclipser son petit frère sur les arêtes.

En effet, nous pensons que l'obligation de contraindre la tournée dans le cas du postier chinois à passer par les arêtes non seulement brise la belle symétrie dont bénéficie le PVC[1] mais surtout complique sérieusement le problème. Nous conjecturons que la version cumulative du problème sur les arêtes est plus difficile que celle sur les sommets.

Le problème du postier chinois est bien connu et nous ne passons en revue que les résultats les plus significatifs dans la section 2.1. Pour plus de détails, nous référons la lectrice principalement aux articles de H. A. Eiselt, M. Gendreau et G. Laporte (Eiselt *et al.*, 1995a,b) et au livre *Arc Routing: Theory, Solutions and Applications - Linear Programming Based Methods for Solving Arc Routing Problems* (Dror, 2000). Le problème de latence minimale, l'équivalent de notre problème sur les sommets, est quant à lui bien plus récent et pas très connu. Nous le disséquons donc plus en détails dans la section 2.2. Finalement, nous concluons dans la section 2.3.

2.1 Le problème du postier chinois classique

Nous ne traitons que des graphes connexes. On distincte classiquement deux cas : les versions sur des graphes dirigés et non-dirigés et la version sur des graphes mixtes, *i.e.* avec

1. Le PVC est en fait un problème d'ordonnancement (voir la section 2.2).

des arêtes (non dirigées) et des arcs (dirigés). Cette distinction est essentiellement due à la complexité différente de ces deux cas. Le premier cas sur les graphes totalement dirigés ou non-dirigés est polynomialement résolvable alors que le cas sur les graphes mixtes est NP-difficile.

L'approche de résolution est la même pour les trois cas. Dans un premier temps, on augmente le graphe, *i.e* on lui rajoute autant d'arcs et d'arêtes que nécessaire, de façon à le rendre eulérien, *i.e.* le graphe possède un circuit eulérien passant une et une seule fois par chacune de ses arêtes et chacun de ses arcs. Une fois cette augmentation obtenue, il ne reste plus qu'à déterminer un tel cycle eulérien.

2.1.1 Le cas des graphes non dirigés et dirigés

Le cas des graphes non dirigés et dirigés se résout polynomialement.

Pour les graphes non dirigés, la condition nécessaire et suffisante est que tous les sommets soient de degré pair[2]. Pour les graphes dirigés, il faut non seulement que tous les sommets soient de degré pair mais en plus, il faut exiger que le nombre d'arêtes entrantes soit égal au nombre d'arêtes sortantes et ce pour tous les sommets (Ford et Fulkerson, 1962). On peut alors entrer et sortir de n'importe quel sommet ce qui permet à la tournée de passer par toutes les arêtes et tous les arcs puisqu'il n'y a aucun endroit où elle pourrait être arrêtée et si l'on prend un arc pour entrer on peut emprunter un arc pour sortir.

Une fois cette conditions remplie, trouver effectivement un circuit eulérien ne pose pas vraiment de problème non plus. Dans le cas non dirigé, il suffit de parcourir le graphe. On commence la tournée n'importe où et il suffit de suivre les arêtes dans le graphe jusqu'à ce qu'on l'ait complètement parcouru ou formé un cycle. Dans ce dernier cas, il faut recommencer à parcourir un autre cycle et fondre les deux cycles en un seul cycle plus grand et ainsi de suite. C'est l'algorithme *End-Pairing* d'Edmonds et Johnson (Edmonds et Johnson, 1973). Le cas dirigé est un peu plus compliqué mais se traite en temps polynômial aussi (algorithmes de Fleury adapté (Dror, 2000, p. 5) ou de van Aardenne-Ehrenfest et de de Bruijn (van Aardenne-Ehrenfest et de Bruijn, 1951)). Il faut en effet faire attention à ne pas se retrouver bloqué en un sommet sans pouvoir en sortir via un arc non encore utilisé alors

2. De manière folklorique, on attribue la condition nécessaire à Euler (Euler, 1736) et la condition suffisante à Hierholzer (Hierholzer, 1873).

que la tournée n'est pas encore complétée.

Dans le cas non dirigé, l'augmentation minimale peut aisément se trouver à l'aide d'un algorithme de couplage parfait alors que dans le cas dirigé un algorithme de flot minimum fait l'affaire.

2.1.2 Le cas des graphes mixtes

Si pour le cas mixte le principe est toujours le même, à savoir : trouver une augmentation minimale pour obtenir un graphe eulérien à partir du graphe original, puis trouver un circuit eulérien, la complexité, elle, est différente. En effet, le cas mixte est NP-difficile. Une autre différence de taille est que si pour les cas dirigé et non dirigé on exige seulement la connexité, dans le cas mixte on exige la connexité forte (c'est-à-dire que tous les sommets soient joignables deux à deux).

La condition nécessaire et suffisante pour que le graphe augmenté soit eulérien est un petit peu plus compliqué aussi : il faut toujours pouvoir entrer et sortir d'un sommet mais comme dans ce cas-ci nous avons des arêtes non dirigées, on ne peut se contenter d'énumérer des conditions seulement sommet par sommet. En effet, pour permettre l'entrée ou la sortie d'un sommet il faudra peut-être exiger qu'une arête non dirigée pendante au sommet soit dirigée d'une certaine façon. Or cette orientation ne convient peut-être pas à l'autre sommet qu'elle relie. Il faut donc des conditions plus globales et celles-ci sont appelées les *conditions d'équilibre ensembliste* d'un graphe (*well balanced set conditions*). Si le graphe possède des sommets tous de degré pair et que les conditions d'équilibre ensembliste du graphe sont respectées alors il est eulérien et vice-versa.

Il existe plusieurs approches exactes pour trouver l'augmentation minimale qui rendra le graphe eulérien mais aucune ne fait l'unanimité. De toute façon, comme le problème est NP-difficile, il faut se tourner vers les heuristiques. La combinaison des deux heuristiques MIXED1 et MIXED2 de Frederickson (Frederickson, 1979) permet d'obtenir une heuristique qui donne dans le pire cas 5/3 fois la valeur optimale.

Une fois le graphe augmenté trouvé, il faut encore déterminer la tournée. Ceci se fait en

trois étapes :

- rendre le graphe symétrique (c'est-à-dire orienter les arêtes pour que le nombre d'arcs entrants soit égal au nombre d'arcs sortant pour tous les sommets),
- orienter les arêtes restantes et
- maintenant que le graphe est orienté, nous nous retrouvons dans le cas orienté dont on peut utiliser les méthodes.

Bref, on voit que le cas mixte n'est pas de tout repos. La lectrice intéressée est guidée à nouveau vers les deux article déjà mentionnés d'Eiselt, Gendreau et Laporte (Eiselt *et al.*, 1995a,b) pour des informations plus détaillées.

2.1.3 D'autres cas

A partir de ces trois cas de base (non dirigé, dirigé et mixte), il existe une multitude de variantes. Il suffit d'ajouter des contraintes supplémentaires. D'un intérêt reconnu sont le cas du problème du postier chinois asymétrique où le coût de l'arête dépend du sens dans laquelle elle est parcourue (le coût le plus élevé symbolise le parcours de l'arête le vent de face alors que le coût le plus modeste symbolise le parcours le vent dans le dos) et le problème du postier chinois hiérarchique où des classes d'arêtes ou d'arcs prioritaires à desservir sont définies. Dans le cas général, ces deux versions sont NP-difficiles (Eiselt *et al.*, 1995a,b).

2.2 La version hamiltonienne du PPCC : le problème de latence minimale (PLM)

Le problème équivalent sur les sommets est connu surtout depuis les années 1990 sous différents vocables. La section 2.2.2 reprend les plus usités. Dans ce document nous utiliserons l'appellation *problème de latence minimum* (PLM) qui semble dominer actuellement.

Dans les sections suivantes, nous résumons les résultats importants sur le PLM. Tout d'abord, nous définissons le problème dans la section 2.2.1 et puis nous passons brièvement en revue les différentes dénominations du problème existantes dans la littérature dans la section 2.2.2. La complexité est traitée dans la section 2.2.3, les approches exactes dans la section 2.2.4, les approximations dans la section 2.2.5 et finalement les cas polynômiaux dans

la section 2.2.6. Nous ne discutons pas des heuristiques ni de l'obtention de bornes pour ce problème.

2.2.1 Définition

Ce problème a été véritablement introduit à la communauté scientifique par Afrati *et al.* dans (Afrati *et al.*, 1986) en 1986 sous le nom du *problème du réparateur itinérant*. Nous traduisons la définition du problème telle qu'écrite dans cet article.

> Considérons la situation suivante : n machines localisées en différents points de la carte doivent être réparées et il n'y a qu'un seul réparateur. On nous donne le temps $t[ij]$ requis par le réparateur pour se déplacer de la machine i à la machine j. Nous faisons l'hypothèse que le temps de réparation est insignifiant (ou de façon équivalente, le même pour toutes les machines). Nous appelons cette variante du temps de séjour moyen du problème du voyageur de commerce le problème du réparateur itinérant (PRI).
>
> De manière formelle, une instance du PRI peut être décrite comme suit : Soient une matrice des distances $t[ij]$ entre n locations et une location distincte de départ s, trouvez une permutation $\pi(0) = s, \pi(1), \ldots, \pi(n-1)$ telle que la fonction objective suivante soit minimisée :
>
> $$c = \sum_{i=1}^{n-1} \sum_{j=1}^{i} t[\pi(j-1), \pi(j)] = \sum_{i=1}^{n-1} (n-i) t[\pi(i-1), \pi(i)]$$

Remarquons que, tout comme dans notre définition du problème du PPCC, le retour au dépôt n'est pas comptabilisé dans cette définition. Nous attirons l'attention de la lectrice sur le fait qu'il n'y a pas vraiment de conscensus sur le retour au dépôt. Certains articles exigent ce retour (Fischetti *et al.*, 1993; Blum *et al.*, 1994; Goemans et Kleinberg, 1998; Archer et Williamson, 2003; Archer *et al.*, 2008) alors que d'autres non (Afrati *et al.*, 1986; Arora et Karakostas, 2003).

Le PLM est un cas particulier[3] du *problème du voyageur de commerce dépendant du temps* (PVCDT) ou *Time-Dependent Traveling Salesman Problem* en anglais (TDTSP). Il s'agit d'une généralisation du problème de voyageur de commerce (voir par exemple Chaudhuri *et al.*, 2003; van Aardenne-Ehrenfest et de Bruijn, 1951) où le coût pour aller du sommet v au sommet w ne dépend pas uniquement de la localisation des sommets v et w mais aussi de leur position dans le tour hamiltonien. Plus spécifiquement, si l'arête e de coût p est traversée en i^e position alors son coût pour le PVCDT est de $c(p,i)$. L'objectif est de minimiser la somme des coûts de visite de tous les sommets. Le PVC, le problème du voyageur de commerce, tout comme le PLM sont des cas particuliers du PVCDT. Dans le premier cas, on prend $c(p,i) = p$ (*i.e.* le coût est indépendant du temps) et dans le deuxième cas, $c(p,i) = (n - i) * p$ avec n étant le nombre de sommets.

Un problème d'ordonnancement tout à fait similaire au PLM est le problème d'ordonnancement $1/s_{jk}/\sum C_j$[4], c'est-à-dire le problème d'ordonnancement à une machine avec temps de préparation dépendant de la séquence et minimisation du temps de séjour moyen.

2.2.2 Nomenclature

Le PLM possède différentes appellations dans la littérature. Nous en regroupons les principales dans le tableau 2.1.

Ces dernières années, il semblerait que le vocable *Minimum Latency Problem* l'emporte. Même s'il ne reflète pas aussi bien que *Problème du voyageur de commerce avec coûts cumulatifs (PVCC)* le fait qu'il s'agisse bien d'un problème de tournées sur les sommets, nous respectons la tendance actuelle et nous désignons ce problème sous le nom du *problème de latence minimale (PLM).*

2.2.3 La complexité

Le PLM est NP-difficile et cela a été démontré par Sahni et Gonzalez en 1976 (Sahni et Gonzalez, 1976). Dans ce même fabuleux petit article de 11 pages, ils démontrent que

3. Le PLM est aussi un cas particulier du *Multiple Traveling Repairman Problem* (mTRP) qui lui-même est un cas particulier du *Cumulative Vehicle Routing Problem* (CumVRP). Voir par exemple (Jothi et Raghavachari, 2007) pour le mTRP et (Kara *et al.*, 2008) pour le CumVRP. Nous ne détaillons pas ces problèmes car ils sont bien trop généraux que pour résoudre raisonnablement le PLM.

4. Les C_j correspondent aux temps de complétion des tâches j.

Version anglaise	Version française	Exemples
Traveling Repairman Problem (TRP)	Problème du réparateur itinérant (PRI)	Afrati *et al.* (1986)
Traveling Salesman Problem with cumulative costs (TSPCC)	Problème du voyageur de commerce avec coûts cumulatifs (PVCCC)	Bianco *et al.* (1993)
Minimum Latency Problem (MLP)	Problème de latence minimale (PLM)	Blum *et al.* (1994)
Delivery man Problem (DMP)	Problème du livreur (PL)	Fischetti *et al.* (1993)
Scheduling Problem $1/s_{jk}/\sum C_j$	Problème d'ordonnancement $1/s_{jk}/\sum C_j$	Bigras *et al.* (2008)

TABLEAU 2.1 Différentes appellations pour le problème de latence minimum.

tout comme pour le PVC dans le cas général, il n'existe pas d'algorithme polynomial pour approximer le problème (à moins que P=NP). C'est pourquoi la grande majorité des auteurs imposent que l'inégalité triangulaire pour les distances (ou le temps) soit respectée. En 1993, Papadimitriou et Yannakakis (Papadimitriou et Yannakakis, 1993) démontrent que ce problème est MAX SNP-difficile et que donc par conséquent il est peu probable que ce problème possède un schéma d'approximation dans le cas général.

La complexité du PLM sur les arbres a longtemps été une question ouverte qui a été résolue en 2002 par Sitters (Sitters, 2002) qui a démontré que même sur les arbres avec seulement des coûts $0-1$ sur les arêtes, le PLM est NP-difficile. Ceci nous démontre que la fonction objectif cumulative introduit une complexité assez forte.

2.2.4 Les approches exactes

Il existe diverses approches exactes. Tout d'abord citons les premières approches proposées pour résoudre le problème du voyageur de commerce dépendant du temps dont le problème de latence minimum est un cas particulier. Par exemple, grâce à un algorithme d'énumération par séparation et évaluation, Picard et Queyranne (Picard et Queyranne, 1978) résolvent des problèmes comprenant jusqu'à 20 villes. Citons aussi l'approche par décomposition de Benders d'un modèle mixte en nombres entiers de Vander Wiel et Sahi-

nidis en 1996 (Wiel et Sahinidis, 1996). Ils peuvent ainsi résoudre des problèmes avec 15 sommets.

Jusqu'à récemment, le problème du voyageur de commerce dépendant du temps n'était pas bien maîtrisé et il fallait se tourner vers des algorithmes plus spécifiques au PLM pour obtenir de meilleurs résultats. Ces derniers concernent la résolution de problèmes avec une soixantaine de sommets au maximum. Les algorithmes proposés peuvent être regroupés en deux catégories : ceux de la programmation dynamique et ceux de l'énumération implicite. Nous présentons quelques résultats avant de revenir sur les récentes résolutions du problème du voyageur de commerce dépendant du temps.

En 1990, Lucena présente une nouvelle formulation du PLM et résout son modèle par une méthode d'énumération implicite où des bornes inférieures sont obtenues par relaxation lagrangienne dans (Lucena, 1990). Il arrive à résoudre des instances avec au plus 30 villes.

Dans (Simchi-Levi et Berman, 1991), Simchi-Levi et Berman décrivent un algorithme d'évaluation et séparation progressive basé sur la relaxation d'un problème d'arbres de recouvrement minimum.

Fischetti, Laporte et Martello proposent une énumération implicite des solutions pour résoudre optimalement le PLM dans (Fischetti *et al.*, 1993) en 1993. Ils étendent la théorie des matroïdes aux matroïdes cumulatifs et en déduisent de nouvelles bornes inférieures. Ils résolvent des problèmes contenant jusqu'à 60 sommets, ce qui était le meilleur résultat obtenu jusqu'à tout récemment.

Bianco, Mingozzi et Ricciardelli proposent dans (Bianco *et al.*, 1993) en 1993 deux algorithmes exacts. Le premier est une méthode de séparation et évaluation progressive alors que le deuxième est un algorithme de programmation dynamique. Dans les deux cas, ils obtiennent l'optimalité pour des problèmes à 35 sommets et arrivent à résoudre parfois des problèmes allant jusqu'à 60 sommets.

van Eijl en 1995 dans (van Eijl, 1995) propose un algorithme de plans coupants pour résoudre un modèle mixte en nombres entiers pour le PLM. Il résout à optimalité des problèmes comprenant jusqu'à 15 sommets.

En 2000, Wu propose dans (Wu, 2000) un algorithme de programmation dynamique pour résoudre le PLM dans le cas général en temps $O(n^2 2^n)$ avec n étant le nombre de sommets. Son intérêt n'est pas tant de résoudre le cas général que de trouver des cas particuliers où son algorithme de programmation dynamique peut s'appliquer en temps polynomial. Wu, Huang et Zhan reprennent l'algorithme de Wu en 2004 (Wu *et al.*, 2004) en ajoutant une élimination de sous-tours et résolvent ainsi des problèmes comprenant jusqu'à 23 sommets.

Récemment, en 2007, Sarubbi et Luna (Sarrubi et Luna, 2007) proposent un nouveau modèle de flots pour modéliser le PLM. Ils ne développent aucune méthode particulière pour résoudre le problème et laissent le soin à CPLEX de résoudre leur modèle. Ils rapportent avoir résolu des problèmes jusqu'à 30 sommets.

Dans le domaine de l'ordonnancement citons par exemple (Stecco *et al.*, 2008) où des instances jusqu'à 50 jobs sont résolues à l'aide d'un algorithme d'évaluation et séparation progressive ou encore (Bigras *et al.*, 2008) où là aussi des instances avec 50 jobs sont résolues à optimalité à l'aide d'un algorithme d'énumération implicite.

Revenons maintenant au problème du voyageur de commerce dépendant du temps. Tout récemment, des progrès immenses ont été faits pour résoudre ce problème. Abeledo, Fukasawa, Pessoa et Uchoa proposent un algorithme *branch-cut-and-price* dans (Abeledo *et al.*, 2010). Ils résolvent presque toutes les instances de la TSPLIB (Reinelt, 1991) qui ont 107 sommets ou moins en moins de 48 heures sur un ordinateur relativement modeste. Mieux, ils comparent leurs résultats avec ce qui était le meilleur code jusqu'à présent de Fischetti, Laporte et Martello (Fischetti *et al.*, 1993) et battent celui-ci sur les plus grandes instances mais pas les plus petites de la TSPLIB.

Notons aussi que beaucoup d'efforts ont été fournis pour le développement de bons modèles linéaires pour le problème du voyageur de commerce dépendant du temps et pour le PLM. Citons les articles récents (Gouveia et Voß, 1995) et (Godinho *et al.*, 2010b) pour le problème du voyageur de commerce dépendant du temps ou encore (Méndez-Díaz *et al.*, 2008), (Ezzine *et al.*, 2010) et (Godinho *et al.*, 2010a) pour le PLM.

2.2.5 Les approximations

Dans toute cette sous-section, un algorithme d'approximation-x désigne un algorithme polynomial dont la borne de pire cas est la valeur optimale du problème multipliée par x. C'est la notion classique du ratio de garantie. Soient f la fonction objectif, I une instance du problème, $Opt(I)$ une solution optimale pour l'instance I et $A(I)$ la solution donnée par l'algorithme d'approximation-x A alors

$$x = \sup_I \left\{ \frac{f(A(I))}{f(Opt(I))} \right\}$$

C'est-à-dire qu'on est certain que quelle que soit l'instance I du problème,

$$f(A(I)) \leq x \cdot f(Opt(I)).$$

Le premier algorithme d'approximation pour le PLM a été proposé par Blum et al. en 1994 (Blum *et al.*, 1994). Il s'agit d'un algorithme d'approximation-144 dans le cas des métriques générales. L'idée est de visiter les sommets les plus proches du dépôt d'abord. Si on sait comment résoudre le *rooted k-MST* , c'est-à-dire résoudre le problème de l'arbre couvrant minimum de k sommets incluant un sommet désigné, pour tous les k, il est naturel de penser à concaténer ces tours. C'est l'approche de Blum et al. qui ont montré qu'une telle approche donne un algorithme d'approximation-8 pour autant qu'on ait une procédure qui nous donne en un temps polynomial ces tours. Malheureusement dans le cas général, le k-MST est NP-difficile. Blum et al. montrent que si on a un algorithme d'approximation-c pour le k-MST on obtient un algorithme d'approximation-8c pour le MLP. Mais en 1994, il n'existait par d'algorithme d'approximation pour le k-MST. Il faut donc modifier cette approche, en particulier il faut prendre d'autres tours à concaténer. Blum et al. ont contourné le problème en introduisant des *approximateurs-PVC-(α,β)* ((α,β)-TSP-approximators).

Soit un PLM sur p sommets débutant en A. Nous avons alors la

Définition 7 *Un approximateur-PVC-(α,β) est un algorithme polynomial capable de trouver, étant donnés des bornes ϵ et L, une tournée débutant en A de longueur au plus égale à βL qui visite au moins $(1-\alpha\epsilon)p$ sommets s'il existe une tournée de longueur L qui visite $(1-\epsilon)p$ sommets.*

L'idée des approximateurs-PVC-(α,β) est que si le nombre maximum de sommets à pouvoir être visités en un temps L est de $p-g(L)$ ($g(L) = \epsilon p$), alors l'approximateur-PVC-(α,β)

trouve un sous-tour comprenant au moins $p - \alpha g(L)$ sommets pour un coût βL. α et β sont tous les deux plus grands que un.

Blum et al. montrent comment remplacer une sous-routine pour trouver la solution au k-MST par un approximateur-PVC-(α,β). Ils obtiennent alors un algorithme d'approximation-$8\lceil\alpha\rceil\beta$ pour le MLP. Dans leur article, ils obtiennent un approximateur-PVC-(3,6) à partir d'un algorithme d'approximation-2 pour le prize-collecting TSP de Goemans et Williamson (Goemans et Williamson, 1995) et donc obtiennent un algorithme d'approximation-144 pour le MLP.

Goemans et Kleinberg (Goemans et Kleinberg, 1998) améliorent cette approche en 1998. D'une part, ils démontrent comment concaténer mieux les sous-tours et d'autre part, ils exhibent un approximateur-PVC-(2,3) à partir d'une relaxation linéaire de l'algorithme d'approximation-2 pour le prize-collecting TSP de Goemans et Williamson, obtenant ainsi un algorithme d'approximation-$2 \cdot 3 \cdot 3{,}5912 \simeq 21{,}55$ du MLP.

L'utilisation de l'algorithme d'approximation-3 de Garg (Garg, 1996) pour le k-MST permet de faire descendre la borne à $1 \cdot 3 \cdot 3{,}5912 \simeq 10{,}78$. Arora et Karakostas ont développé un algorithme d'approximation-$(2 + \epsilon)$ du k-MST en 2002 (Arora et Karakostas, 2000), ce qui permet de faire descendre la borne à $7.18 + \epsilon$.

En 2003, Archer and Williamson (Archer et Williamson, 2003) ont réussi à obtenir une borne légèrement supérieure (9,28) mais avec un algorithme beaucoup plus rapide ($\Theta(n^4 \cdot \log(n))$ fois plus rapide !). L'idée est qu'ils ne considèrent pas la sous-routine pour approximer le k-MST comme une boîte noire mais plutôt interagissent avec celle-ci, ce qui leur permet d'appeler cette sous-routine nettement moins souvent. En 2008, Levin se joint à eux et ensemble ils font descendre la borne à 7,18 (Archer *et al.*, 2008).

La meilleure borne actuelle est due à Chaudhuri, Godfrey, Roa et Talwar (Chaudhuri *et al.*, 2003). Ils réussissent à faire descendre la borne à ≈ 3.59 dans le cas des métriques générales. Il est à noter que si le graphe est de poids unitaire, Koutsoupias, Papadimitriou et Yannakakis proposent un algorithme d'approximation-1.662 dans (Koutsoupias *et al.*, 1996) en 1996. Récemment, en 2010, Archer et Blasiak ont réussi à faire descendre la borne à 3,03 pour les arbres (Archer et Blasiak, 2010).

Mentionnons aussi un *quasi-polynomial time approximation scheme* (QPTAS) pour certaines métriques particulières dans (Arora et Karakostas, 2003).

2.2.6 Les cas polynomiaux

Afrati et al. démontrent en 1986 (Afrati *et al.*, 1986) comment résoudre le cas de la ligne droite par un algorithme de programmation dynamique en $O(n^2)$. Peu après, en 1989, Minieka (Minieka, 1989) prouve qu'une recherche en profondeur donne une solution optimale pour l'arbre unitaire. Dans leur article (Blum *et al.*, 1994), Blum et al. reprennent l'algorithme d'Afrati et al. pour le cas des arbres de diamètre plus petit ou égal à trois mais ne parviennent pas à l'étendre au-delà. En 1996, Koutsoupias, Papadimitriou et Yannakakis proposent un algorithme polynomial pour les arbres dont le nombre de feuilles est fixé à l'avance dans (Koutsoupias *et al.*, 1996). García et al. (García *et al.*, 2002) proposent un algorithme en $O(n)$ pour le cas de la ligne droite en 2001.

En 2000, Wu propose dans (Wu, 2000) un algorithme de programmation dynamique pour le cas général en temps $O(n^2 2^n)$ avec n étant le nombre de sommets. En fait, il s'agit essentiellement du même algorithme de programmation dynamique proposé par Held et Carp (Held et Carp, 1962) pour résoudre le problème du voyageur de commerce. Wu s'intéresse au cas de la ligne droite qu'il résout en temps $O(n^2)$, et plus généralement au cas où il y a k réparateurs itinérants. Dans ce dernier cas, il démontre comment adapter son algorithme pour obtenir la résolution de la ligne droite en $O(n^3)$ lorsque $k = 2$ et en temps $O(n^4)$ lorsque $k > 2$.

2.3 Conclusions

Le PPCC ressemble bien plus à son petit frère sur les sommets, le problème de latence minimum, qu'aux problèmes classiques de postier chinois. Il existe une littérature abondante concernant le PLM tant sur les approches exactes qu'approximées. Les plus gros problèmes résolus exactement ne contiennent au maximum qu'une centaines de sommets, ce qui semble indiquer une difficulté intrinsèque au problème.

Le problème de latence minimum est MAX SNP-difficile mais dans le cas des métriques générales, le meilleur algorithme d'approximation avec borne de pire cas est un algorithme d'approximation-3.59.

Chapitre 3

PROPRIÉTÉS DU PROBLÈME

And now for something completely different.

Monty Python

Nous décrivons dans ce chapitre quelques propriétés du problème : sa complexité, les différences fondamentales avec la version classique du problème du postier chinois, quelques familles de graphes pour lesquelles la résolution du PPCC se fait en temps polynomial. Nous regardons aussi de plus près la filiation entre le PPCC et son cousin sur les sommets, le PLM. Finalement, nous revenons sur la conjecture énoncée dans notre mémoire de maîtrise (Voir van Omme, 2003, pp. 58-61) et qui traite du cas de la grille rectangulaire unitaire sans trou.

Les sections 3.1, 3.2 et 3.4 présentent des résultats qui ont été démontrés dans notre mémoire de maîtrise et s'en inspirent largement. Par manque de place, nous rassemblons ces résultats sans les détailler. Dans la section 3.3 nous présentons une transformation simple entre notre problème et le problème équivalent sur les sommets. Mentionnons aussi que la conjecture sur le cas de la grille unitaire rectangulaire sans trou reprise dans la section 3.4 est une version modifiée de la conjecture énoncée dans notre mémoire de maîtrise.

3.1 Le PPCC est NP-difficile

Dans notre mémoire de maîtrise (Voir van Omme, 2003, pp. 29-31), nous avons démontré que le PPCC est NP-difficile et même fortement NP-difficile. Nous nous contentons de ces deux résultats car notre objectif premier dans cette thèse est de résoudre exactement ce problème. Nous n'avons donc pas essayé de voir dans quelle mesure ce problème s'approxime facilement ou pas.

3.2 Différences avec le problème du postier chinois classique

Comme pour le problème de latence minimum (PLM) (Voir Bianco *et al.*, 1993), le PPCC se différencie nettement du problème classique du postier chinois. Un changement local dans la structure du graphe ou dans le coût de ses arêtes, même minimal comme le rajout d'une seule arête ou le changement du coût d'une seule arête, peut modifier globalement la structure d'une solution optimale.

Le paradigme du postier chinois « *on cherche une augmentation minimale du graphe pour le rendre eulérien et puis on trouve un tour eulérien* »n'est plus applicable ici et pas seulement parce que nous n'exigeons pas de retour au dépôt. La différence entre les deux problèmes est vraiment essentielle. Illustrons-là par trois aspects.

3.2.1 Un chemin eulérien n'est pas optimal

L'exemple introductif malgré sa petite taille illustre déjà très bien certaines difficultés inhérentes au PPCC. Le graphe de l'exemple introductif de la figure 1.1 page 9 est eulérien mais la solution optimale de la figure 1.2 n'est pas un chemin eulérien[1].

3.2.2 Le graphe augmenté n'est pas d'un grand secours

Le concept de graphe augmenté, en tout cas dans sa version classique, n'a pas l'air de nous être d'un grand secours. On peut construire aisément des exemples où le graphe augmenté possède $m - 1$ copies d'une arête. La figure 3.1 représente un tel exemple.

Même dans le cas d'un graphe unitaire, où tous les coûts sont égaux à un, le concept de graphe augmenté n'a pas l'air d'être d'une grande utilité. La figure 3.2 montre un exemple pour lequel l'augmentation minimale[2] au sens classique ne donne aucune information sur la solution optimale.

1. Le fait que le chemin eulérien dans le cas de cet exemple introductif est aussi un cycle eulérien n'est en rien significatif.

2. Comme nous ne demandons pas de retour au dépôt, l'augmentation minimale doit être comprise au sens des graphes traversables, *i.e.* des graphes qui possèdent un chemin eulérien.

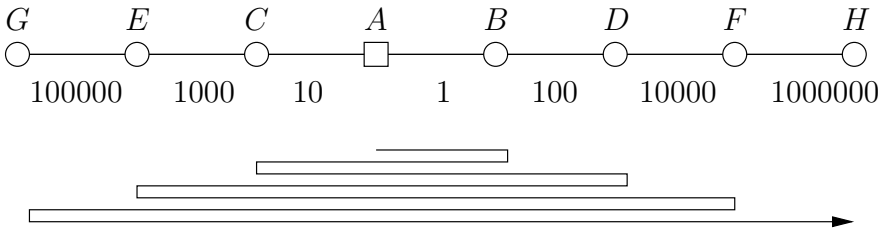


FIGURE 3.1 Une arête peut être copiée plusieurs fois dans le graphe augmenté. Il y a six copies de l'arête AB dans le graphe augmenté qui donne la solution optimale $A - B - C - D - E - F - G - H$.

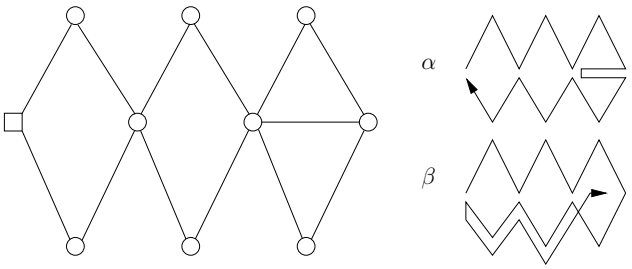


FIGURE 3.2 L'augmentation minimale pour rendre un graphe unitaire traversable ne sert à rien. La solution α a un coût de 97 alors que la solution β ne coûte que 95 mais demande une augmentation de quatre arêtes.

3.2.3 Le problème ne peut pas être subdivisé en sous-problèmes sur des sous-graphes

La figure 3.3 nous montre qu'il n'est pas possible de séparer aisément un graphe en parties disjointes pour traiter chacune des parties séparément comme cela pourrait être le cas pour traiter la version classique du problème.

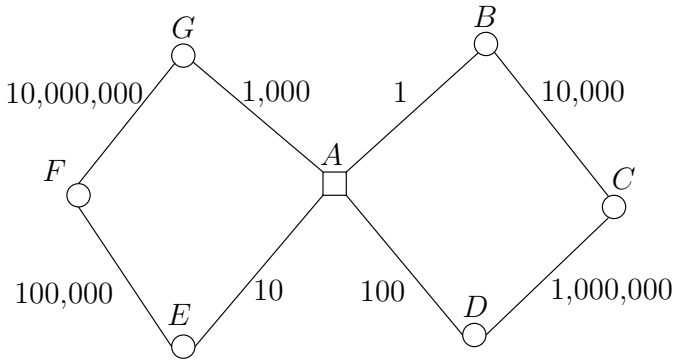


FIGURE 3.3 Comment décomposer le problème en sous-problèmes ? La solution optimale consiste à parcourir les arêtes par ordre de grandeur croissant. Il n'y a pas moyen de séparer le problème en traitant séparément le cycle $A - B - C - D$ et le cycle $A - E - F - G$ par exemple.

3.3 Un problème similaire au PLM ?

Dans (Laporte, 1995) et (Dror, 2000, Chapitre 8, pp. 277-326), plusieurs versions du problème du postier chinois (PPC) sont transformées en problèmes équivalents sur les sommets. Vu la similarité *a priori* entre le PLM et le PPCC, on peut se demander quels sont les rapports qu'entretiennent ces deux problèmes. Nous n'avons pas réussi à établir de lien direct, c'est-à-dire une réduction en temps polynomial de l'un à l'autre qui applique toute solution de l'un en une solution de l'autre[3]. Les quelques résultats préliminaires obtenus semblent indiquer que les deux problèmes sont à la fois relativement similaires mais aussi fondamentalement différents.

Définissons d'abord les différents problèmes que nous pouvons situer les uns par rapport aux autres.

3. Nous cherchons une transformation simple d'un problème à l'autre et non une réduction qui ne se soucierait que d'appliquer une solution optimale sur une solution optimale.

Le problème existant équivalent sur les sommets est le *problème de latence minimum (PLM)* dont la définition est donnée à la section 2.2.1 page 25. Ce problème consiste en la recherche d'une tournée passant une et une seule fois par tous les sommets du graphe et qui minimise la somme des latences[4] de tous les sommets. Le sommet de départ, soit le dépôt, est donné *a priori* et le retour au dépôt est exigé dans le sens où la tournée doit se terminer au dépôt et où la latence du dépôt est comptabilisée dans le coût de la fonction objectif.

En fait, le retour au dépôt dans le PLM est sans doute une caractérisation de ce problème qui le rend essentiellement différent du nôtre. Nous considérons dès lors aussi la version sur les sommets qui ne demande pas un retour au dépôt. Cette version, sans le retour au dépôt, est la version considérée dans l'article d'Afrati *et al* (Afrati *et al.*, 1986)[5] mais qui, assez curieusement, n'a pas été retenue par la suite. Pour faire la distinction avec la version qui a finalement prédominé, celle avec retour au dépôt, nous appelons le problème de trouver un chemin passant une et une seule fois par tous les sommets et commençant par un sommet donné sans le retour au dépôt, le *problème du chemin hamiltonien cumulatif (PCHC)*.

Il y a notre version, le problème du postier chinois cumulatif (PPCC). Rappelons encore une fois que le dépôt est donné *a priori* et qu'une fois toutes les arêtes desservies nous n'exigeons pas le retour au dépôt. Ce non-retour est sans doute la principale difficulté à surmonter pour transformer simplement le PPCC en PLM.

Finalement nous considérons le problème sur les arêtes avec retour au dépôt. Appelons *problème du postier chinois cumulatif avec retour (PPCCAR)* le problème du postier chinois cumulatif dans lequel on exige et comptabilise le retour au dépôt.

La figure 3.4 résume les transformations connues entre ces différents problèmes. La transformation du PPCC vers le PCHC est essentiellement la même que la transformation du PPCCAR vers le PLM. Nous donnons cette dernière transformation dans la section suivante.

4. De manière similaire à la définition de la latence d'une arête, la latence d'un sommet est définie comme le temps écoulé depuis le début de la tournée jusqu'à ce sommet.

5. C'est cet article qui a véritablement introduit le problème à la communauté scientifique.

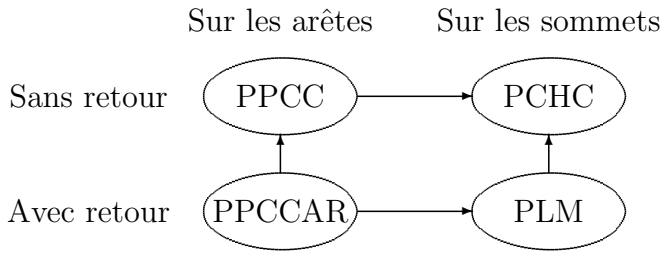


FIGURE 3.4 Les flèches indiquent les transformations simples connues pour passer d'un problème à un autre. Pour passer des versions avec retour aux versions sans retour, il suffit d'ajouter une arête avec un coût exorbitant ou un sommet fortement éloigné du dépôt.

3.3.1 Une transformation du PPCCAR en PLM

Nous présentons la transformation du PPCCAR, le problème du postier chinois cumulatif avec retour, en PLM, le problème de latence minimum. La transformation du PPCC (sans retour) en PCHC est essentiellement la même.

Reprenons l'exemple introductif de la page 8 mais cette fois-ci résolvons le problème du postier chinois cumulatif avec retour au dépôt. Dans ce cas-ci, la solution est la même comme nous le montre la figure 3.5.

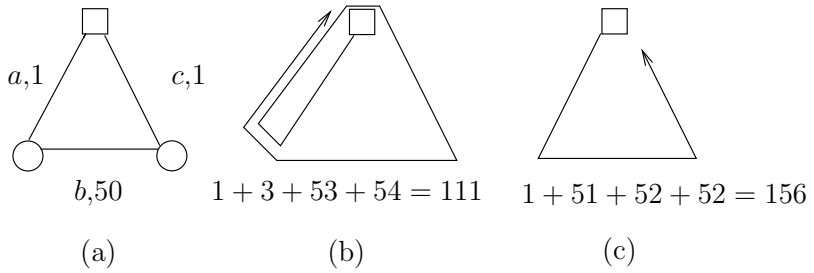


FIGURE 3.5 La solution optimale pour le PPCCAR de l'exemple introductif. Pour le problème du postier chinois cumulatif avec retour (PPCCAR), le retour au dépôt est comptabilisé dans le quatrième terme des sommes de (b) et (c). La solution optimale est représentée en (b).

La transformation se fait en 3 étapes.

Étape 1

Soit un graphe $G = (\mathrm{V},\mathrm{E})$. Transformons-le en un graphe $\hat{G} = (\mathrm{V},\hat{\mathrm{E}})$ comme sur la figure 3.6. Pour ce faire, nous remplaçons chaque arête a du graphe G par une paire d'arcs de sens opposés a_1 et a_2 dans $\hat{G}$.

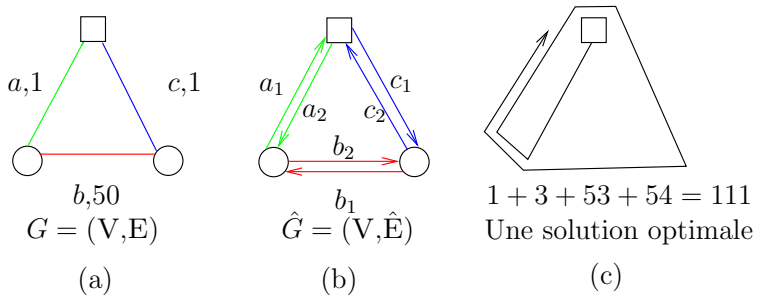


FIGURE 3.6 Première étape de la transformation du PPCCAR en PLM. Le graphe $G = (\mathrm{V},\mathrm{E})$ est transformé en un graphe $\hat{G} = (\mathrm{V},\hat{\mathrm{E}})$ où chaque arête a de G a été remplacée par une paire d'arcs de sens opposés a_1 et a_2. Une solution optimale est représentée en (c).

Étape 2

La deuxième étape transforme le problème du postier chinois cumulatif sur les paires d'arcs du graph $\hat{G}$ en un problème de voyageur de commerce cumulatif généralisé sur le graphe $\tilde{G}$. Le graphe $\tilde{G} = (\tilde{\mathrm{V}},\tilde{\mathrm{E}})$ est obtenu en doublant tous les sommets de $\hat{G}$ et en introduisant deux sommets fictifs s_1 et s_2 comme sur la figure 3.7.

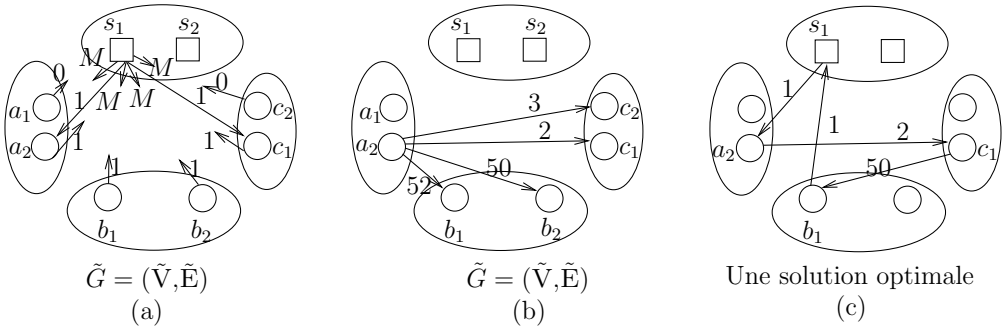


FIGURE 3.7 Deuxième étape de la transformation du PPCCAR en PLM. En (a), le coût des arcs sortants de s_1 ou s_2 correspond au coût de l'arête représentée par le sommet correspondant si cette arête est accessible dans le sens désigné par le sommet, un coût exorbitant M sinon. Les coûts des arcs entrant en s_1 ou s_2 correspondent au plus courts chemins entre ces sommets et le dépôt dans le graphe G. En (b), le coût des arcs d'un sommet x à un sommet y représentant deux arêtes différente du graphe G est le coût du plus court chemin entre ces deux sommets sur le graphe G. Une solution optimale du problème de voyageur de commerce cumulatif généralisé de coût $1 + 3 + 53 + 54 = 111$ est représentée en (c).

On force le départ de la tournée en s_1. Les coûts des arcs entrants et sortants des sommets s_1 et s_2 sont les mêmes. Pour les arcs sortants, on prend le coût des arêtes directement accessibles depuis le dépôt dans le graphe G lorsque c'est possible ou bien un coût exorbi-

tant M qui empêche la tournée de commencer par des arêtes non directement accessibles depuis le dépôt. Ainsi dans notre exemple, le sommet s_1 est relié au sommet a_2 pour un coût de 1 alors qu'il est relié au sommet c_2 pour un coût M car il est inaccessible depuis le dépôt.

De manière plus générale, un arc du sommet x au sommet y représentant deux arêtes différentes du graphe G a pour coût le plus court chemin pour aller de x vers y dans le graphe G. Nous obtenons ainsi un graphe que nous pouvons solutionner par un problème de voyageur de commerce cumulatif généralisé.

Étape 3

Dans la dernière étape de la transformation, on transforme le problème de voyageur de commerce cumulatif généralisé en un problème de latence minimum par la transformation de Noon et Bean (Noon et Bean, 1991). Cette dernière transforme un problème de voyageur de commerce généralisé en un problème de voyageur de commerce. On peut l'appliquer telle quelle car les coût cumulatifs sont simplement doublés. Détaillons la transformation de Noon et Bean. Pour chaque sous-ensemble de sommets, x_1 et x_2, on relie ces sommets par un cycle de coût nul. La figure 3.8 illustre cette construction.

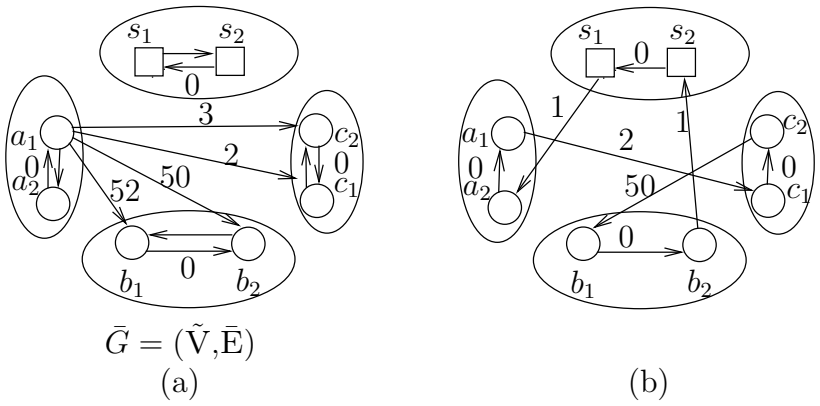


FIGURE 3.8 Troisième étape de la transformation du PPCCAR en PLM. Le graphe $\bar{G} = (\tilde{V},\bar{E})$ est obtenu à partir du graphe $\tilde{G} = (\tilde{V},\tilde{E})$ en rajoutant des arcs de coût nul entre chaque pair de sommets représentants la même arête. Les coûts sortants d'un sommet x_1 sont les coûts des arcs correspondants sortants du sommet x_2 et vice-versa du graphe $\tilde{G}$. Une solution optimale est représentée en (b). Elle commence en s_1 et son coût est de $1 + 1 + 3 + 3 + 53 + 53 + 54 + 54 = 2 \cdot 111$.

Les coûts sur les arcs sortants d'un sommet x_1 sont les coûts des arcs sortants du sommet correspondant x_2 et vice-versa dans le graphe $\tilde{G}$. La transformation de Noon et Bean

fonctionne car la valeur de la fonction objectif ainsi obtenue pour des solutions équivalentes dans le graphe $\tilde{G}$ de la deuxième étape est simplement doublée.

3.4 Cas polynomiaux

Nous reprenons les quatre cas résolvables[6] en temps polynomial qui ont été introduits dans notre mémoire de maîtrise. Nous ne reprenons que les résultats et nous nous inspirons largement de l'exposé fait dans notre mémoire. La lectrice est renvoyée au mémoire pour une démonstration et une explication plus détaillée de ces résultats (van Omme, 2003, pp 42-57).

3.4.1 Le cercle et la ligne droite

Le PPCC défini sur le cercle et la ligne droite sont résolus en $O(n^2)$ grâce à un algorithme de programmation dynamique d'Afrati et al. (Afrati *et al.*, 1986) présenté pour le PLM sur la ligne droite mais qui s'applique aussi au cas du cercle. Ce dernier cas n'est pas recensé, à notre connaissance, parmi les problèmes résolus du PLM et est donc nouveau.

Nous montrons brièvement comment faire pour le cercle. L'algorithme pour la droite est similaire.

Théorème 8 *Le PPCC défini sur le cercle et sur la droite sont de complexité polynomiale et se résolvent en temps $O(n^2)$.*

La figure 3.9 montre comment nous numérotons les sommets d'un cercle.

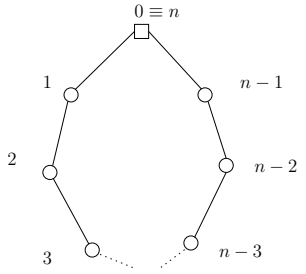


FIGURE 3.9 Numérotation des sommets du cercle. Les sommets sont numérotés à partir de 0 pour le dépôt dans le sens anti-horaire. Le dépôt reçoit aussi le numéro n.

6. Plusieurs autres cas triviaux sont présentés dans la conclusion de ce chapitre.

Nous utilisons une récursion arrière. Soit $f(i,j,k)$ le coût de ce qu'il reste à desservir quand on a déjà tout couvert du dépôt jusqu'au sommet i à gauche et du dépôt jusqu'au sommet j à droite. L'indice k nous indique si nous avons terminé par la droite ($k = 0$) ou par la gauche ($k = 1$). Pour s'en rappeler plus aisément, $f(i,j,0)$ sera surmonté d'une flèche vers la droite ($\overrightarrow{f(i,j,0)}$) alors que $f(i,j,1)$ le sera d'une flèche vers la gauche ($\overleftarrow{f(i,j,1)}$).

Soit $d(i,j)$ la distance entre les sommets i et j lorsqu'on passe par le dépôt.

Si toutes les arêtes sont desservies, il reste un coût nul : $\overrightarrow{f(i,i,0)} = \overleftarrow{f(i,i,1)} = 0$ alors que s'il reste une seule arête à desservir, nous avons $\overrightarrow{f(i,i+1,0)} = \min \{ \quad c(i,i+1), d(i,i+1) + c(i,i+1) \quad \}$ par exemple.

S'il reste deux arêtes, alors pour un arc de cercle vers la gauche nous avons :

$$\begin{aligned}\overleftarrow{f(i,i+2,1)} = \min \quad \{ \quad & 2 \cdot c(i,i+1) + \overleftarrow{f(i+1,i+2,1)}, \\ & 2 \cdot \{d(i,i+2) + c(i+1,i+2)\} + \overrightarrow{f(i,i+1,0)} \quad \}\end{aligned}$$

En général :

$$\begin{aligned}\overleftarrow{f(i,j,1)} = \min \quad \{ \quad & (j-i) \cdot c(i,i+1) + \overleftarrow{f(i+1,j,1)}, \\ & (j-1) \cdot \{d(i,j) + c(j-1,j)\} + \overrightarrow{f(i,j-1,0)} \quad \}\end{aligned}$$

et

$$\begin{aligned}\overrightarrow{f(i,j,0)} = \min \quad \{ \quad & (j-i) \cdot c(j-1,j) + \overrightarrow{f(i,j-1,0)}, \\ & (j-1) \cdot \{d(i,j) + c(i,i+1)\} + \overleftarrow{f(i+1,j,1)} \quad \}\end{aligned}$$

Nous cherchons $\overleftarrow{f(0,n,1)}$ et $\overrightarrow{f(0,n,0)}$.

3.4.2 L'arbre unitaire

Le PLM sur l'arbre unitaire (coût de un pour toutes les arêtes) est très simplement résolu : n'importe quelle recherche en profondeur donne une solution optimale. Ce résultat,

dû à Minieka (Minieka, 1989) s'étend pour le PPCC sur l'arbre unitaire.

Théorème 9 *Le PPCC dans le cas de l'arbre unitaire est polynomial et résolvable en temps $O(m)$.*

3.4.3 La bande unitaire

Définissons la bande unitaire. Ce type de graphes particuliers est illustré sur la figure 3.10.

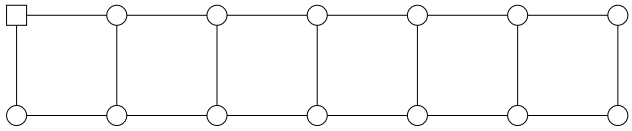


FIGURE 3.10 Une bande unitaire. Une bande unitaire $s = 6$. Tous les coûts sont de 1. Le dépôt est arbitrairement fixé dans le coin supérieur gauche.

Une bande unitaire est un carré suivi d'une succession de trois arêtes qui forment des carrés supplémentaires et est caractérisée par le nombre total des s carrés qui la composent. On parlera d'une bande s pour décrire la bande à s carrés. Toutes les arêtes ont un coût unitaire. Pour une bande $s = k$ (carrés) il y a $3k + 1$ arêtes. Nous plaçons le dépôt dans le coin supérieur gauche.

Théorème 10 *Le PPCC dans le cas de la bande unitaire est polynomial en temps $O(m)$.*

Toutes les solutions pour la bande unitaire ont une structure similaire à celle de la figure 3.11.

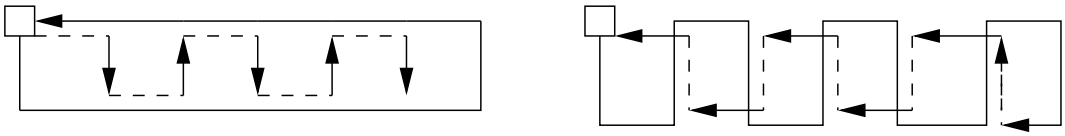


FIGURE 3.11 Les deux solutions optimales pour la bande unitaire. Deux solutions optimales pour la bande unitaire de la figure 3.10.

3.5 La conjecture sur les grilles unitaires

Dans notre mémoire de maîtrise sur le problème du postier chinois cumulatif, nous conjecturions que la grille unitaire rectangulaire sans trou se résolvait en temps polynomial (Voir

van Omme, 2003, pp 58-60). La figure 3.12 présente cette conjecture.

Nous avions remarqué que le cas 2×2 était à exclure de cette conjecture. En effet, la figure 3.13 montre une solution optimale meilleure que celle suggérée par la conjecture.

En fait, c'est toute la bande de largeur $k = 2$ qu'il faut reconsidérer, ce que nous faisons dans la section suivante. C'est la seule modification que nous faisons de cette conjecture. Nous la conservons donc mais pour $3 \leqslant k \leqslant l$.

3.5.1 Conjecture pour la bande unitaire de largeur 2

Nous conjecturons que le PPCC sur la bande unitaire de largeur 2 se résout en temps polynomial et qu'il y a trois cas à considérer. Ces trois cas sont réprésentés sur la figure 3.14.

3.6 Conclusions

Le PPCC est un problème vraiment différent des problèmes de postier chinois classiques et il est fortement NP-difficile. Nous avons vu quelques familles de graphes pour lesquelles ce problème se résout en temps polynomial. Nous pouvons rajouter des cas triviaux comme par exemple les graphes eulériens unitaires, les arbres caterpillars de longueur deux et trois ou les graphes étoiles mais ces cas restent bien peu nombreux.

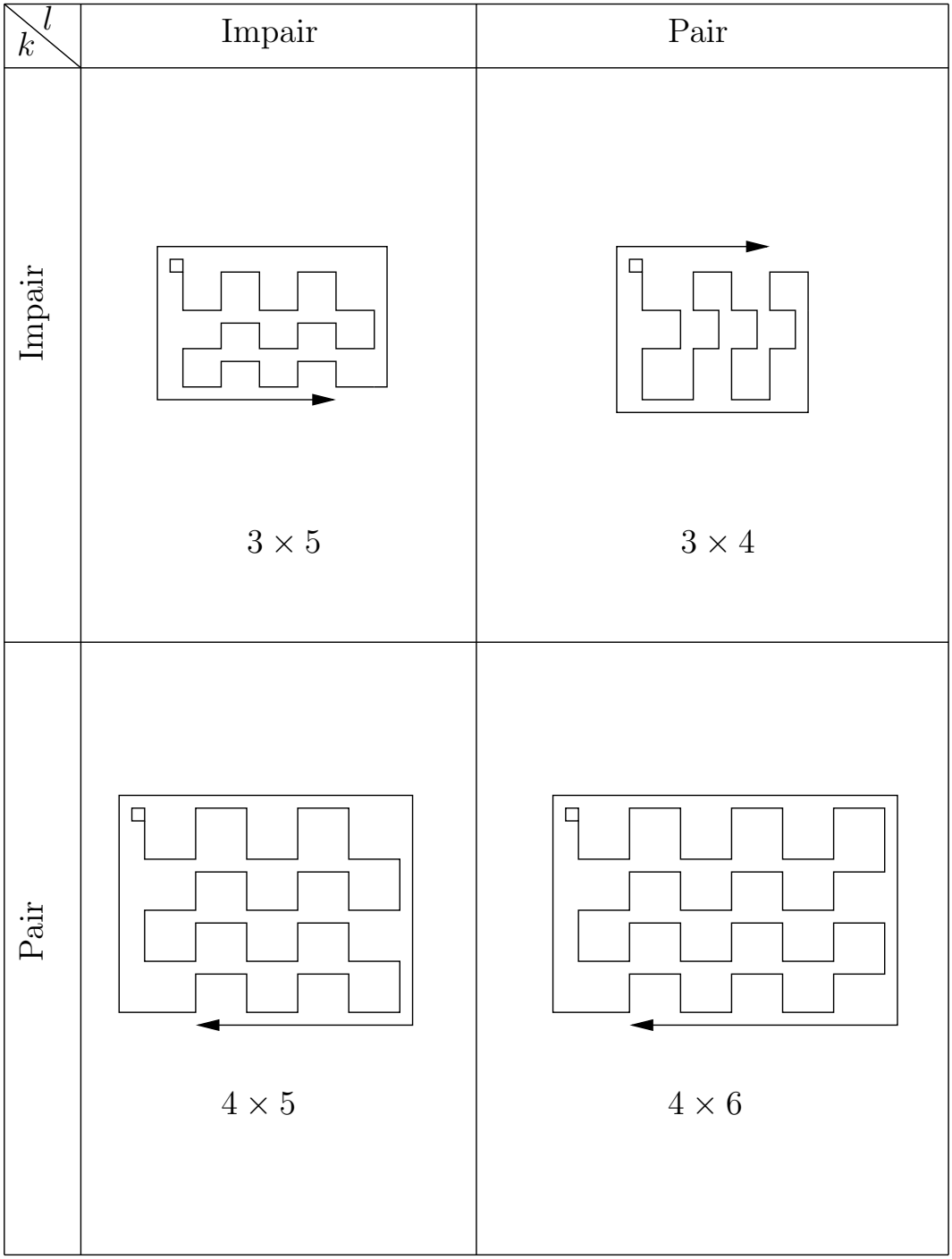
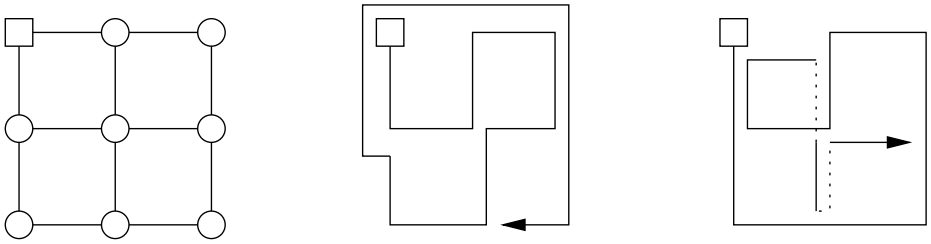


FIGURE 3.12 L'ancienne conjecture pour la grille unitaire. Cette conjecture pour le cas général des grilles unitaires $k \times l$ avec $k \leqslant l$ consiste à parcourir l'intérieur des grilles en zigzaguant puis à terminer en desservant les arêtes de la frontière. Nous modifions légèrement cette conjecture et nous excluons maintenant le cas de la bande $2 \times l$. Ce cas un peu particulier est traité sur la figure 3.14.

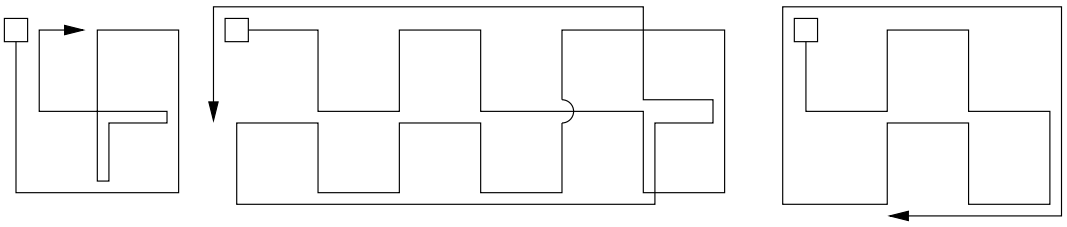


Grille 2×2.

La solution suivant la conjecture pour un coût de 85.

La solution optimale pour un coût de 84.

FIGURE 3.13 Le cas 2×2.



2×2

2×6
$2 \times l, l \geqslant 4$
l pair

2×3
$2 \times l, l \geqslant 3$
l impair

FIGURE 3.14 Les trois cas de la conjecture pour la grille $2 \times l$.

Chapitre 4

COMPARAISONS DE DIFFÉRENTS MODÈLES

The more the merrier

Common wishful thinking

Nous sommes confrontés à deux tâches bien différentes lorsque nous voulons modéliser le problème du postier chinois cumulatif. D'une part, il faut que la solution obtenue soit un chemin partant d'un sommet fixé — le dépôt — et qui doit passer au moins une fois par toutes les arêtes du graphe et d'autre part, il faut calculer le coût cumulatif de ce chemin. Si la première exigence est relativement simple à modéliser, il nettement moins aisé de calculer de façon efficace les coûts cumulatifs. La difficulté croît nettement lorsque nous combinons ces deux exigences en un seul modèle.

Nous avons essayé bien des approches différentes et développé plus d'une vingtaine[1] de modèles. Petit à petit, nous avons amélioré nos modèles jusqu'à obtenir notre meilleure version : le modèle L8 qui représente l'aboutissement de nos recherches. Nous l'introduisons ici sommairement car il sera disséqué plus en détails dans les deux chapitres suivants.

Tous les modèles présentés ici sont des modèles linéaires car tel est le cadre de notre recherche. Ceci dit, le problème du postier chinois cumulatif peut se modéliser aussi par des modèles non linéaires et en fait c'est peut-être un cadre plus naturel pour ce problème. La lectrice est invitée à prendre connaissance du modèle Q1 dans la section D.1.1 page 201 de l'annexe. Ce modèle est un modèle quadratique convexe qui a été écrit spécialement pour être résolu par le solveur CPLEX mais il existe peut-être d'autres modélisations non linéaires mieux adaptées.

1. Sans compter les multiples variantes pour certains de ces modèles. Dans l'annexe D page 201, la lectrice trouvera quelques uns des modèles non repris ici. Ils représentent des approches naturelles qui à première vue s'imposent comme de bonnes approches mais que nous avons abandonnées après une étude plus minutieuse.

Pour comparer l'efficacité des modèles, nous insistons dans ce chapitre surtout sur la comparaison des bornes inférieures obtenues par relaxation des contraintes d'intégralité. Nous utilisons dans tout ce chapitre l'abréviation *bi* ou parlons simplement de *bornes* pour désigner celles-ci et désignons par *solution relaxée* la solution fractionnaire optimale obtenue en relâchant les conditions d'intégralité.

Bien que le critère des *bi* soit important, ce n'est évidemment pas le seul dont il faut tenir compte pour comparer les modèles entre eux. Nous manquons malheureusement de place pour expliciter toutes nos tentatives. Par exemple, nous avons passé beaucoup de temps à chercher des coupes efficaces pour certains modèles.

Ce chapitre est divisé comme suit. Nous présentons d'abord huit modèles parmi les plus intéressants dans la section 4.1 et puis les comparons entre eux d'abord de manière empirique à la section 4.2 puis nous confirmons ces résultats expérimentaux de manière théorique dans la section 4.3. Nous concluons dans la section 4.4.

Le modèle L8 a été obtenu avec la collaboration des professeurs Ángel Corberán de l'Université de Valence et José María Sanchís de l'Université Polytechnique de Valence.

4.1 Présentation des différents modèles

Plusieurs possibilités s'offrent à nous pour modéliser le problème du postier chinois cumulatif mais très vite nous nous sommes rendus compte qu'en surchargeant les modèles d'information nous obtenions des modèles de plus en plus intéressants. Nous présentons d'abord des modèles dépouillés puis de plus en plus chargés d'information. La lectrice pourrait s'étonner que certaines approches relativement naturelles ne soient pas présentées ici. Nous ne prétendons évidemment pas à l'exhaustivité mais lui suggérons de jeter un coup d'œil à l'annexe D page 201. Peut-être y trouvera-t-elle son approche ou du moins une approche similaire ? Les huit modèles présentés ici sont ceux qui possèdent les meilleures *bi* parmi tous les modèles que nous avons développés.

Sauf mention explicite contraire, chaque modèle est présenté avec un nombre minimal de familles de contraintes, c'est-à-dire que si nous retirons une seule de ces familles, le modèle ainsi obtenu ne résout plus le problème du postier chinois cumulatif. Par contre, il se peut très bien qu'au sein d'une même famille ou entre deux familles différentes il y ait une certaine

redondance entre certaines contraintes.

Pour aider la lectrice dans sa compréhension des modèles, nous présentons systématiquement graphiquement une solution optimale détaillée de l'exemple introductif de la section 1.3 page 8. Nous présentons aussi les graphes auxiliaires les plus complexes utilisés dans la construction de certains modèles. Nous suggérons fortement à la lectrice d'étudier d'abord ces figures avant d'essayer de comprendre les modèles.

De manière rigoureuse[2], il faudrait démontrer l'équivalence entre tous les modèles présentés ici. Nous présentons deux de ces démonstrations : l'équivalence entre les modèles L1 et L2 dans la section C.1.1 et l'équivalence entre les modèles L2 et L3 dans la section C.1.2 car il s'agit de deux modèles relativement différents puisque L2 est une approche basée sur les arcs alors que L3 se base sur des sommets. Nous ne montrons l'équivalence que pour des solutions modèles (définies section 1.4.6 page 13) car ce sont les seules que nous devons vraiment considérer. Par manque de place nous n'avons pas mis les démonstrations d'équivalence entre les autres modèles.

Enfin, une dernière petite mise en garde avant de nous lancer dans la présentation des huit modèles. Tous les modèles ont leur propre nomenclature et parfois un même symbole peut exprimer deux choses différentes pour deux modèles distincts. Nous avons préféré garder une certaine cohérence au sein de chaque modèle plutôt que de modifier artificiellement les symboles pour qu'ils aient chacun une fonction unique. Ainsi la lettre E et ses variantes est réservée aux ensemble d'arêtes mais dépendamment des modèles, ces ensembles peuvent être très différents. Nous recommandons donc à la lectrice d'être vigilante. Le graphe original est toujours représenté par $G = (V,E)$ avec V l'ensemble de ses sommets au nombre de $|V| = n$ et E l'ensemble de ses arêtes au nombre de $|E| = m$.

Rappelons que le modèle L1 a été présenté à la section 1.5 page 14.

4.1.1 L2

Nous commençons petit à petit à rajouter de l'information. Pour ce modèle-ci, nous disons explicitement qu'un chemin partiel à l'étape k doit être construit à partir du chemin partiel de l'étape précédente $k - 1$. De plus, nous tenons explicitement compte du sens de passage

2. Nous considérons comme acquis le fait que les modèles proposés résolvent bel et bien le PPCC.

sur les arêtes. Pour ce faire, nous utilisons une approche naturelle de résolution par des flots cumulatifs : nous exigeons qu'à chaque période k un flot de 1 passe par le chemin partiel de la période $k-1$ et par le diamètre de la période k. Nous utilisons un graphe auxiliaire construit comme suit. Nous dédoublons les arêtes $e \in E$ par une paire d'arcs $\{e_1,e_2\}$ et nous relions tous les sommets à un sommet fictif T par un arc de coût nul. Ce graphe auxiliaire dirigé est noté $\tilde{G} = (\mathrm{V} \cup \{T\},A)$. Introduisons une nouvelle notation : $\delta_G^+(i) = \{v \in V \ : (i,v) \in A\}$ et $\delta_G^-(i) = \{v \in V \ : (v,i) \in A\}$.

Les variables

$x_{ij}^k \in \mathbb{Z}$ représente le flot sur l'arc $(i,j) \in A$ appartenant au chemin partiel de la période k. Comme il est possible de repasser plusieurs fois sur un arc, nous devons considérer des variables entières ou nulles. Nous ajoutons aussi des variables x_{iT}^k correspondantes aux arcs de coût nul qui aboutissent en T.

$$y_{ij}^k = \begin{cases} 1 & \text{si l'arête } (i,j) \text{ a déjà été desservie à l'étape } k \text{ (en } k \text{ ou avant).} \\ 0 & \text{sinon.} \end{cases}$$

Le modèle

$$\min_{x_{ij}^k, y_{ij}^k} \sum_{k=1}^{m} \sum_{(i,j)\in \mathrm{E}} c_{ij}\,(x_{ij}^k + x_{ji}^k)$$

s.à :

$$x_{ij}^k \leqslant x_{ij}^l \qquad \forall\, 1 \leqslant k < l \leqslant m, \forall\,(i,j) \in A \qquad (4.1)$$

$$x_{iT}^k + \sum_{j\in \delta_G^+(i)} x_{ij}^k - \sum_{j\in \delta_G^-(i)} x_{ji}^k = \begin{cases} 1 & i \text{ est dépôt} \\ 0 & \text{sinon.} \end{cases} \qquad \forall\, k \in \{1,\ldots,m\}, \forall\, i \in \mathrm{V} \qquad (4.2)$$

$$\sum_{i\in \mathrm{V}} x_{iT}^k = 1 \qquad \forall\, k \in \{1,\ldots,m\} \qquad (4.3)$$

$$x_{iT}^k + \sum_{j\in \delta_G^+(i)} x_{ij}^k \leqslant \sum_{j\in \delta_G^+(i)} x_{ij}^{k+1} \qquad \forall\, k \in \{1,\ldots,m-1\}, \forall i \in \mathrm{V} \qquad (4.4)$$

$$y_{ij}^k \leqslant x_{ij}^k + x_{ji}^k \qquad \forall\, k \in \{1,\ldots,m\}, \forall\,(i,j) \in \mathrm{E} \qquad (4.5)$$

$$x_{ij}^k + x_{ji}^k \leqslant m\, y_{ij}^k \qquad \forall\, k \in \{1,\ldots,m\}, \forall\,(i,j) \in \mathrm{E} \qquad (4.6)$$

$$\sum_{(i,j)\in \mathrm{E}} y_{ij}^k = k \qquad \forall\, k \in \{1,\ldots,m\} \qquad (4.7)$$

et les variables x_{ij}^k sont entières positives ou nulles et les y_{ij}^k binaires.

(4.1) force les chemins partiels à emprunter les mêmes arcs (mais pas forcément dans le même ordre). (4.2) et (4.3) s'occupent de la conservation du flot pour tous les sommets de $\tilde{G}$ et toutes les étapes k. (4.4) impose que si un arc — dont le dernier sommet est $i \in V$ — a été desservi en k, il soit un arc de passage en $k+1$. (4.5) et (4.6) assurent le lien entre les y_{ij}^k et x_{ij}^k. (4.7) certifie que chaque chemin partiel passe par k arêtes exactement. Notons que le choix du dépôt est fait par (4.2).

L'exemple introductif

CPLEX trouve la solution optimale de la figure 4.1.

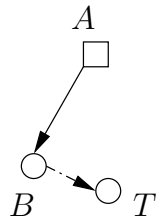
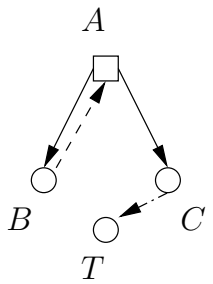
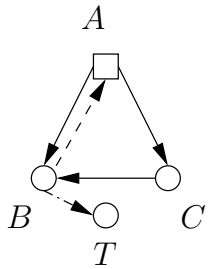
k	arête(s) desservie(s)	flot passant sur	chemins partiels
1	AB $(y_{AB}^1 = 1)$	AB,BT $(x_{AB}^1 = 1)$ $(x_{BT}^1 = 1)$	
2	AB,AC $(y_{AB}^2 = 1)$ $(y_{AC}^2 = 1)$	AB,BA,AC,CT $(x_{AB}^2 = 1)$ $(x_{BA}^2 = 1)$ $(x_{AC}^2 = 1)$ $(x_{CT}^2 = 1)$	
3	AB,AC,BC $(y_{AB}^3 = 1)$ $(y_{AC}^3 = 1)$ $(y_{BC}^3 = 1)$	AB,BA,AC CB,BT $(x_{AB}^3 = 1)$ $(x_{BA}^3 = 1)$ $(x_{AC}^3 = 1)$ $(x_{CB}^3 = 1)$ $(x_{BT}^3 = 1)$	

FIGURE 4.1 Une solution optimale de l'exemple introductif pour le modèle L2.

4.1.2 L3

Une manière de calculer les coûts cumulatifs est l'approche dite du *grand* M. Disons tout de suite que nous ne sommes pas parvenus à l'utiliser efficacement même si cette approche donne des modèles relativement réduits par le nombre de variables et par le nombre de contraintes. Nous l'introduisons ici pour illustrer la beauté de ces modèles[3]. Comme nous le verrons par la suite, les *bi* obtenues sont parmi les plus mauvaises. Ceci s'explique par les grandeurs M disproportionnellement grandes qu'il faut considérer pour borner des coûts cumulatifs ou des coûts de plus courts chemins. Ce phénomène est bien connu.

Comme nous ne considérons que des arcs et pour pouvoir utiliser une écriture plus simple, nous utilisons un graphe auxiliaire (voir la figure 4.2 page 53) formé uniquement de sommets représentants des arcs du graphe original. Cet ensemble de sommets est noté $\bar{V}$ et nous notons le graphe auxiliaire par $\bar{G} = (\bar{V},\emptyset)$. Soit i un arc desservi à la période $k-1$ et j l'arc desservi à la période suivante k, nous notons par $c(i,j)$ le coût d'un plus court chemin entre le dernier sommet de i et le premier sommet de j. Ces distances seront bornées par un grand M :

$$M = \max\{\text{dist}(v_1,v_2) : v_1,v_2 \in V\}.$$

Soit une arête $e \in E$. Nous noterons par e_1 et e_2 les deux arcs correspondants de $\bar{V}$.

Les variables

Deux familles de variables sont utilisées : la première indique si oui ou non un arc donné est desservi à la période k et la seconde permet de calculer exactement le coût d'une liaison inter-clients pour la solution correspondante.

$$y_a^k = \begin{cases} 1 & \text{si on dessert l'arc } a \text{ en } k^e \text{ position.} \\ 0 & \text{sinon.} \end{cases}$$

$z^k =$ coût de la liaison inter-clients entre les arcs desservis en $(k-1)^e$ et k^e position.

3. La lectrice est invitée à décortiquer cette même approche dans un cadre encore plus naturel en considérant le modèle L10 dans l'annexe D.1.1 page 204.

Le modèle

$$\min_{y_a^k, z^k} \; m \cdot \left(\sum_{a \in \bar{V}} c(a) \cdot y_a^1 \right) + \sum_{k=2}^{m} (m-k+1) \left(\sum_{a \in \bar{V}} c(a) \cdot y_a^k + z^k \right)$$

s.à :

$$\sum_{k=1}^{m} (y_{e_1}^k + y_{e_2}^k) = 1 \qquad \forall\, e \in E \tag{4.8}$$

$$\sum_{a \in \bar{V}} y_a^k = 1 \qquad \forall\, k \in \{1, \ldots, m\} \tag{4.9}$$

$$(y_i^{k-1} - 1) \cdot M + y_j^k \cdot c(i,j) \leqslant z^k \qquad \forall\, i,j \in \bar{V},\, \forall\, k \in \{2, \ldots, m\} \tag{4.10}$$

$$\sum_{\substack{(ij) \in \bar{V}: \\ *i \text{ est le dépôt}}} y_{ij}^1 = 1 \qquad \text{le dépôt} \tag{4.11}$$

et les variables y_a^k sont binaires et les variables z^k sont entières positives ou nulles. Notons que les indices i et j représentent des arcs dans les contraintes (4.10) mais représentent des sommets dans les contraintes (4.11).

(4.8) impose que toute arête du graphe original soit desservie alors que (4.9) impose qu'à toute période k déterminée une et une seule arête (un arc) du graphe original soit desservie. (4.10) permet de calculer ce que coûterait une liaison inter-clients si les arcs i et j étaient desservis en $(k-1)^e$ et k^e positions respectivement. Finalement, (4.11) désigne le dépôt.

L'exemple introductif

La construction de la solution optimale obtenue par CPLEX est résumée dans la figure 4.2.

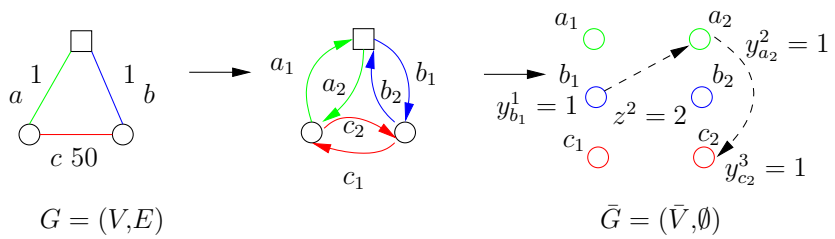


FIGURE 4.2 Une solution optimale de l'exemple introductif pour le modèle L3. On transforme le graphe original en dédoublant les arêtes par des arcs. Les flèches en pointillé ne représentent aucune variable et indiquent seulement l'ordre dans lequel on dessert les trois arcs.

4.1.3 L4

Comme nous le verrons dans la suite, l'approche du modèle L3 est trop grossière pour obtenir de bonnes *bi*. Nous raffinons cette approche en calculant explicitement les coûts des liaisons inter-clients.

Le graphe auxiliaire pour ce modèle est construit comme suit. Nous dédoublons les arêtes par des arcs et nous représentons chacun de ces arcs par un sommet comme sur la figure 4.3. Soit $\bar{V}$ cet ensemble de sommets. Nous relions deux sommets entre eux si les deux arcs correspondants sont tels que la destination de l'un est l'origine de l'autre et nous notons par $\bar{A}$ cet ensemble d'arcs. Le coût d'un tel arc $(i,j) \in \bar{A}$ est le coût de l'arête représentée par l'arc j. Nous rajoutons aussi les arcs suivants dans l'ensemble $\bar{A}$. Pour modéliser le dépôt, nous rajoutons un sommet fictif s que nous relions seulement aux arcs qui débutent par le dépôt avec le coût d'utiliser ceux-ci. Nous construisons aussi des arcs de tous les sommets de $\bar{V}$ vers le sommet fictif s avec un coût nul pour modéliser un retour fictif au dépôt. Nous considérons donc $m + 1$ étapes.

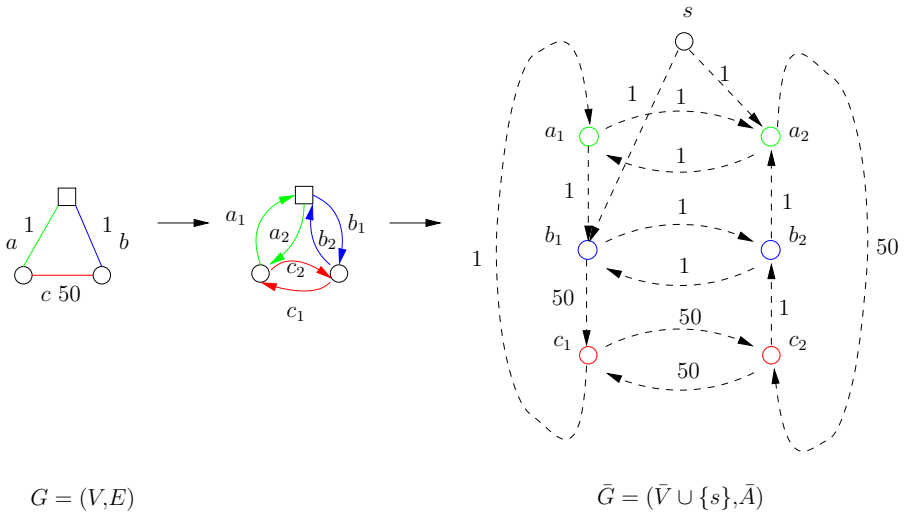


FIGURE 4.3 Le graphe auxiliaire utilisé pour le modèle L4. On transforme le graphe original en dédoublant les arêtes par des arcs. Les arcs de coût nul de retour de chacun des sommets de $\bar{V}$ vers le sommet fictif s ne sont pas représentés.

Les variables

Chaque arête $e \in E$ doit être desservie et est représentée par une paire d'arcs e_1 et e_2. Pour savoir quel est l'arc qui a été desservi, nous utilisons une variable binaire[4] $q^k_{e_i}$ qui indique si oui ou non l'arc e_i a été desservi à la période k :

$$q^k_a = \begin{cases} 1 & \text{si on dessert l'arc } a \text{ à la période } k. \\ 0 & \text{sinon.} \end{cases}$$

Nous rajoutons des variables pour construire un diamètre à chaque étape k :

$$x^k_{ij} = \begin{cases} 1 & \text{si on utilise l'arc } j \text{ après avoir emprunté l'arc } i \text{ dans le diamètre} \\ & \text{de la période } k. \\ 0 & \text{sinon.} \end{cases}$$

4. Nous utilisons une variable $q^k_{e_i}$ plutôt qu'une variable $y^k_{e_i}$ pour pouvoir la distinguer plus facilement lorsque viendra le temps de démontrer certaines relations de dominance entre modèles à la section 4.3 page 74.

Le modèle

$$\min_{x_{ij}^k} \sum_{k=1}^{m} (m-k+1) \cdot \left(\sum_{(i,j)\in\bar{E}} c_{ij} \cdot x_{ij}^k \right)$$

s.à :

$$\sum_{k=1}^{m} \left(q_{e_1}^k + q_{e_2}^k \right) = 1 \qquad \forall\, e \in E \tag{4.12}$$

$$\sum_{a\in\bar{V}} q_a^k = 1 \qquad \forall\, k \in \{1,\ldots,m\} \tag{4.13}$$

$$\sum_{k=1}^{m} \sum_{\substack{i\in\bar{V}:\\(i,a)\in\bar{A}}} x_{ia}^k = \sum_{k=2}^{m+1} \sum_{\substack{j\in\bar{V}:\\(a,j)\in\bar{A}}} x_{aj}^k \qquad \forall\, a \in \bar{V} \tag{4.14}$$

$$\sum_{\substack{i\in\bar{V}:\\(i,a)\in\bar{A}}} x_{ia}^k \leq \sum_{\substack{j\in\bar{V}:\\(a,j)\in\bar{A}}} x_{aj}^k + q_a^k \qquad \forall\, a \in \bar{V},\ \forall\, k \in \{1,\ldots,m\} \tag{4.15}$$

$$q_a^k \leq \sum_{\substack{i\in\bar{V}:\\(i,a)\in\bar{A}}} x_{ia}^k \qquad \forall\, a \in \bar{V},\ \forall\, k \in \{1,\ldots,m\} \tag{4.16}$$

$$q_a^k \leq \sum_{\substack{j\in\bar{V}:\\(a,j)\in\bar{A}}} x_{aj}^{k+1} \qquad \forall\, a \in \bar{V},\ \forall\, k \in \{1,\ldots,m\} \tag{4.17}$$

$$\sum_{a\in\bar{V}:\exists(s,a)\in\bar{A}} x_{s,a}^1 = 1 \qquad \text{Le dépôt.} \tag{4.18}$$

et les variables x_{ij}^k et q_a^k sont binaires.

Chaque arête du graphe original doit être desservie grâce à (4.12) et (4.13) impose qu'à toute période k une arête soit desservie. Nous y ajoutons les contraintes de conservation de flot (4.14) ainsi que des contraintes de propagation individuelle de flots (4.15). Le liens entre les deux familles de variables est assuré par (4.16) et (4.17). Le dépôt quant à lui est spécifié par (4.18).

L'exemple introductif

La construction de la solution optimale obtenue par CPLEXest résumée dans la figure 4.4.

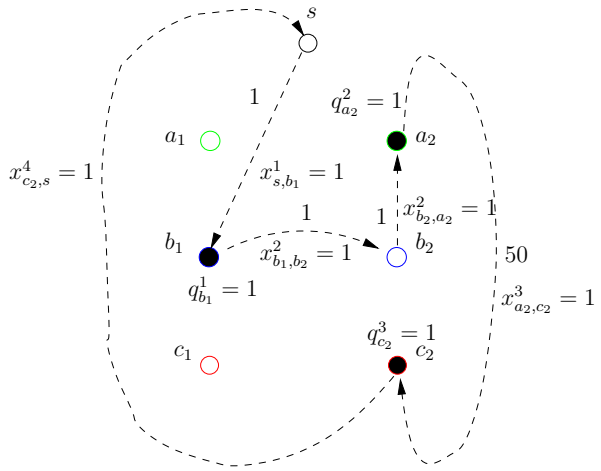


FIGURE 4.4 Une solution optimale de l'exemple introductif pour le modèle L4.

4.1.4 L5

Pour ce modèle, nous exploitons le fait qu'une solution optimale doit nécessairement passer par des plus courts chemins comme nous l'avons démontré pour le lemme 3 à la page 7. La figure 4.5 montre le graphe auxiliaire utilisé. Nous décomposons le graphe en ses arêtes en dédoublant ses sommets et nous ajoutons pour chaque pair de sommets (i,j) représentant un plus court chemin possible, une arête (en trait discontinu sur la figure 4.5) dont le coût est celui du plus court chemin. Le nouveau nom des sommets est obtenu comme suit : une arête (α_i,β_i) du graphe auxiliaire représentant une arête originale (α,β) reçoit un indice i différent pour chaque arête originale.

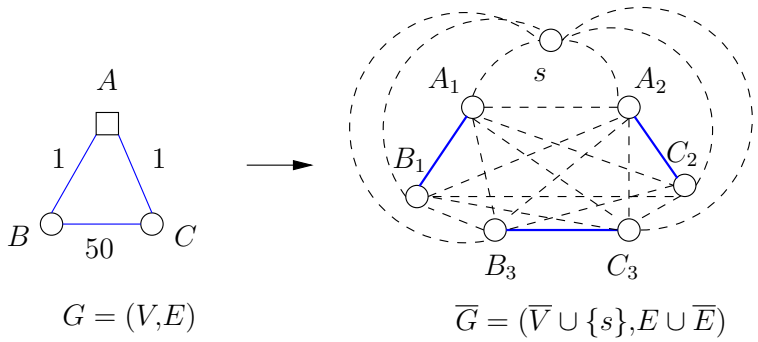


FIGURE 4.5 Le graphe auxiliaire du modèle L5.

Un sommet fictif s est rajouté pour permettre la création de la tournée depuis le dépôt. Le nouveau graphe ainsi obtenu est $\overline{G} = (\overline{\mathrm{V}} \cup \{s\}, \mathrm{E} \cup \overline{\mathrm{E}})$ avec $\overline{\mathrm{V}}$ l'ensemble des sommets dédoublés de G et $\overline{\mathrm{E}}$ l'ensemble des arêtes représentants les plus courts chemins. Les arêtes entre les sommets de $\overline{V}$ et s sont de coût nul.

Nous modélisons la construction d'une solution optimale en exigeant qu'à chaque étape un nouveau chemin partiel soit construit à partir du chemin partiel de l'étape précédente.

Les variables

Nous utilisons une famille de variables binaires représentants des parties de chemins partiels que nous séparons en deux catégories pour plus de clarté : la première concerne les arêtes $\in E$ du graphe et la deuxième correspond aux plus courts chemins $\in \overline{E}$ utilisés dans la construction de ces chemins partiels. Nous utilisons la même lettre pour décrire les deux catégories de variables pour souligner qu'essentiellement ces deux catégories n'en forment qu'une seule.

$$v_e^k = \begin{cases} 1 & \text{si on prend l'arête } e \in \mathrm{E} \text{ dans le chemin partiel de l'étape } k \\ 0 & \text{sinon.} \end{cases}$$

$$v_f^k = \begin{cases} 1 & \text{si on prend l'arête } f \in \overline{\mathrm{E}} \text{ dans le chemin partiel de l'étape } k. \\ 0 & \text{sinon.} \end{cases}$$

Nous ne tenons pas compte de l'orientation des arêtes car le fait d'exiger que le chemin partiel soit composé d'arêtes et de liaisons inter-client couplé avec l'exigence que le premier chemin partiel parte du dépôt permet d'orienter les arêtes automatiquement.

Le modèle

$$\min_{v_e^k, v_f^k} \sum_{k=1}^{m} \left(\sum_{e \in \mathrm{E}} c(e) \cdot v_e^k + \sum_{f \in \overline{\mathrm{E}}} c(f) \cdot v_f^k \right)$$

s.à :

$$\sum_{e \in \mathrm{E}} v_e^k = k \text{ et } \sum_{f \in \overline{\mathrm{E}}} v_f^k = k \qquad \forall\, k \in \{1, \ldots, m\} \tag{4.19}$$

$$v_e^{k-1} \leq v_e^k \text{ et } v_f^{k-1} \leq v_f^k \qquad \forall\, e \in \mathrm{E}, \forall\, f \in \overline{\mathrm{E}},\ \forall\, k \in \{2, \ldots, m\} \tag{4.20}$$

$$\sum_{e \in \mathrm{E}: S \in e} v_e^k \leq 1 \text{ et } \sum_{f \in \overline{\mathrm{E}}: S \in f} v_f^k \leq 1 \qquad \forall\, S \in \mathrm{V} \cup \{s\},\ \forall\, k \in \{1, \ldots, m\} \tag{4.21}$$

$$\sum_{f \in \overline{\mathrm{E}}; f \cap e \neq \emptyset} v_f^k \geq v_e^k \qquad \forall\, e \in \mathrm{E},\ \forall\, k \in \{1, \ldots, m\} \tag{4.22}$$

$$v_f^k \leq \sum_{e \in \mathrm{E}; f \cap e \neq \emptyset} v_e^{k-1} \qquad \forall\, f \in \overline{\mathrm{E}},\ \forall\, k \in \{2, \ldots, m\} \tag{4.23}$$

$$\sum_{f \in \overline{\mathrm{E}}; s \in f} v_f^1 = 1 \qquad \text{Le dépôt} \tag{4.24}$$

et toutes les variables v_e^k et v_f^k sont binaires.

(4.19) indique que dans chaque chemin partiel à l'étape k il faut avoir k arêtes. (4.20) exige qu'un chemin partiel à l'étape k passe par le chemin partiel à l'étape $k-1$. Un chemin partiel ne passe qu'une seule fois par un sommet et c'est ce que (4.21) demande en ne permettant pas d'avoir plus de deux arêtes pendantes en un sommet. (4.22) indique qu'une arête est connectée par un plus court chemin dans un chemin partiel. Les liaisons inter-clients à deux étapes consécutives doivent être reliées entre elles, ce que permet (4.23). Le choix du dépôt est fait grâce à (4.24)

L'exemple introductif

La construction de la solution optimale trouvée par CPLEX est résumée dans la figure 4.6.

4.1.5 L6

Nous reprenons l'approche et la nomenclature du modèle L5 mais cette fois-ci en ne considérant que des diamètres. Comme nous le verrons plus loin, ce petit changement donne

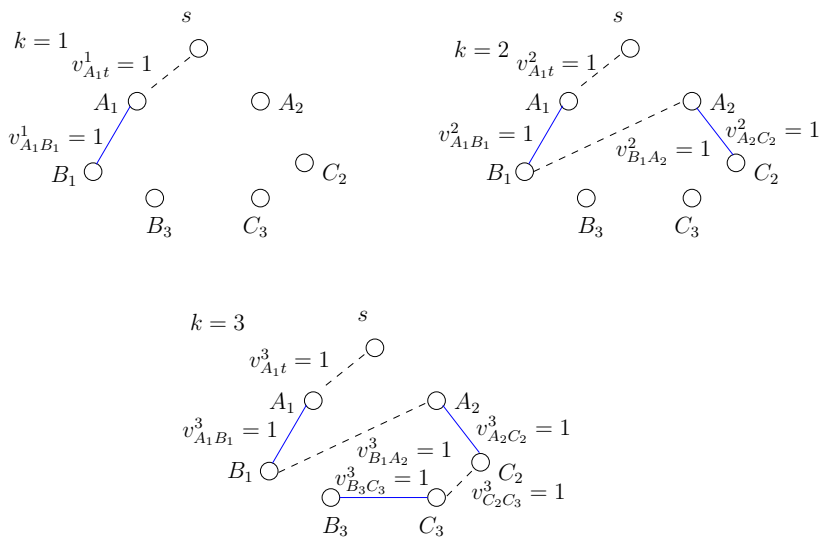


FIGURE 4.6 Une solution optimale de l'exemple introductif par le modèle L5.

deux modèles assez différents.

Les variables

Cette fois-ci, nous marquons la différence entre les variables pour les arêtes originales et les variables pour les liaisons inter-clients. Nous utilisons donc deux familles de variables. Dans le modèle L5, ces deux arêtes font essentiellement partie de chemins partiels alors qu'ici nous avons une solution réalisable construite à partir d'une alternance de ces arêtes. En fait, la véritable raison de les distinguer sera fournie plus tard lorsque nous verrons que la donnée des variables sur un type d'arêtes (arêtes originale du graphe ou arête de plus courts chemins) permet de retrouver les valeurs des variables sur l'autre type d'arêtes.

$$x_e^k = \begin{cases} 1 & \text{si l'arête } e \in \mathrm{E} \text{ est desservie en k}^e \text{ position.} \\ 0 & \text{sinon.} \end{cases}$$

$$y_f^k = \begin{cases} 1 & \text{si on prend le plus court chemin } f \in \overline{\mathrm{E}} \text{ en k}^e \text{ position.} \\ 0 & \text{sinon.} \end{cases}$$

À nouveau, nous ne tenons pas compte de l'orientation des arêtes pour les mêmes raisons que celles énoncées pour le modèle L5. Cet ajout sera fait pour les arêtes originales dans le modèle L7 et pour les plus courts chemins et les arêtes originales dans le modèle L8.

Le modèle

$$\min_{x_e^k, y_f^k} \sum_{k=1}^{m} \left(\sum_{e \in \mathrm{E}} x_e^k \cdot (m-k+1) \cdot c(e) + \sum_{f \in \overline{\mathrm{E}}} y_f^k \cdot (m-k+1) \cdot c(f) \right)$$

s.à :

$$\sum_{k=1}^{m} x_e^k = 1 \qquad \forall\, e \in \mathrm{E} \tag{4.25}$$

$$\sum_{e \in \mathrm{E}} x_e^k = 1 \qquad \forall\, k \in \{1, \ldots, m\} \tag{4.26}$$

$$\sum_{k=1}^{m} y_f^k \leq 1 \qquad \forall\, f \in \overline{\mathrm{E}} \tag{4.27}$$

$$\sum_{f \in \overline{\mathrm{E}}} y_f^k = 1 \qquad \forall\, k \in \{1, \ldots, m\} \tag{4.28}$$

$$\sum_{f \in \overline{\mathrm{E}} : f \cap e \neq \emptyset} (y_f^k + y_f^{k+1}) \geq 2x_e^k \qquad \forall\, e \in \mathrm{E}, \forall\, k \in \{1, \ldots, m-1\} \tag{4.29}$$

$$\sum_{f \in \overline{\mathrm{E}} : f \cap e \neq \emptyset} y_f^m \geq x_e^m \qquad \forall\, e \in \mathrm{E} \tag{4.30}$$

$$\sum_{k=1}^{m} \sum_{f \in \overline{\mathrm{E}} : S \in f} y_f^k \leq 1 \qquad \forall\, S \in \overline{\mathrm{V}} \tag{4.31}$$

$$\sum_{f \in \overline{\mathrm{E}} : t \in f} y_f^1 = 1 \qquad \text{Le dépôt} \tag{4.32}$$

et toutes les variables x_e^k et y_f^k sont binaires.

(4.25) impose que chaque arête du graphe doit être desservie alors que (4.26) demande qu'à chaque période une seule arête soit desservie. Chaque plus court chemin ne peut être utilisé qu'une seule fois au plus (4.27) et pour chaque étape un plus court chemin doit être utilisé (4.28). Il ne reste plus qu'à connecter les plus courts chemins avec les arêtes et faire commencer la solution au dépôt. Chaque arête doit être coincée entre deux plus courts chemins (4.29) sauf la dernière desservie qui ne touche qu'un seul plus court chemin (4.30). En chaque sommet, il ne peut y avoir qu'un seul plus court chemin, ce qu'indique (4.31). Finalement, nous exigeons de partir du dépôt grâce à (4.32).

L'exemple introductif

La construction de la solution optimale trouvée par CPLEX est résumée dans la figure 4.7.

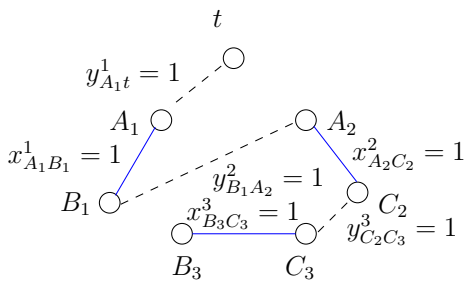


FIGURE 4.7 Une solution optimale de l'exemple introductif pour le modèle L6.

4.1.6 L7

Nous peaufinons notre approche des modèles L5 et L6 dont nous reprenons le graphe auxiliaire de la figure 4.5 page 57 légèrement modifié en dupliquant le sommet fictif : la solution part d'un sommet fictif s pour aboutir en un sommet fictif t. L'ajout du sommet t pour le dépôt nous permet de faciliter l'implémentation du modèle. Nous en reparlerons lorsque nous aborderons le modèle L8 qui utilise lui aussi ces deux sommets fictifs. Cette fois-ci, nous rajoutons de l'information en renforçant le fait de devoir desservir une arête dans un seul sens avec l'introduction de variables q_S.

Les variables

Trois familles de variables sont utilisées. La première famille concerne les liaisons inter-clients (les variables y^k_f) et la deuxième famille concerne les arêtes (les variables x^k_e). Ces deux familles indiques si ces liaisons inter-clients et ces arêtes sont utilisées ou pas dans la construction du diamètre de la période k. La troisième (les variables q_S) permet de renforcer le sens du service sur les arêtes.

$$x^k_e = \begin{cases} 1 & \text{si on dessert l'arête } e \text{ en k}^e \text{ position.} \\ 0 & \text{sinon.} \end{cases}$$

$$y_f^k = \begin{cases} 1 & \text{si on utilise la liaison inter-clients en k}^e \text{ position.} \\ 0 & \text{sinon.} \end{cases}$$

$$q_S = \begin{cases} 1 & \text{si le sommet } S \in \overline{\mathrm{V}} \cup \{s,t\} \text{ est le premier ou dernier sommet} \\ & \text{d'un diamètre.} \\ 0 & \text{sinon.} \end{cases}$$

Notons que la connaissance des valeurs des variables x_e^k ou y_f^k permet de reconstituer les valeurs de l'ensemble de toutes les variables.

Le modèle

$$\min_{x_e^k, y_f^k, q_S} \sum_{k=1}^{m} \left(\sum_{e \in \mathrm{E}} x_e^k \cdot (m-k+1) \cdot c(e) + \sum_{f \in \overline{\mathrm{E}}} y_f^k \cdot (m-k+1) \cdot c(f) \right)$$

s.à :

$$\sum_{k=1}^{m+1} \sum_{f \in \overline{\mathrm{E}}: S \in f} y_f^k = 1 \qquad \forall\, S \in \overline{\mathrm{V}} \cup \{s,t\} \tag{4.33}$$

$$\sum_{k=1}^{m} x_e^k = 1 \qquad \forall\, e \in E \tag{4.34}$$

$$\sum_{f \in \overline{\mathrm{E}}} y_f^k = 1 \qquad \forall\, k \in \{1, \ldots, m\} \tag{4.35}$$

$$\sum_{e \in \mathrm{E}} x_e^k = 1 \qquad \forall\, k \in \{1, \ldots, m\} \tag{4.36}$$

$$q_i + q_j = 1 \qquad \forall\, e \in \mathrm{E}; e = (i,j) \tag{4.37}$$

$$-q_S \leq \sum_{f \in \overline{\mathrm{E}}: S \in f} y_f^k - x_e^k \leq q_S \qquad \forall\, k \in \{1, \ldots, m\}, \forall\, S \in \overline{\mathrm{V}}, S \in e \tag{4.38}$$

$$-1 + q_S \leq \sum_{f \in \overline{\mathrm{E}}: S \in f} y_f^k - x_e^{k-1} \leq 1 - q_S \qquad \forall\, k \in \{2, \ldots, m+1\}, \forall\, S \in \overline{\mathrm{V}}, S \in e \tag{4.39}$$

$$\begin{gathered} q_s = 1, q_t = 0, \\ \sum_{f \in \overline{\mathrm{E}}: s \in f} y_f^1 = 1, \sum_{f \in \overline{\mathrm{E}}: t \in f} y_f^{m+1} = 1 \end{gathered} \tag{4.40}$$

et les toutes les variables x_e^k, y_f^k et q_S sont binaires.

(4.33) impose une liaison inter-clients par sommet alors que (4.34) impose que toutes les arêtes soient desservies. Nous ne pouvons utiliser qu'une seule liaison inter-clients par

période, ce que (4.35) nous confirme. De la même manière, une seule arête doit être desservie par période comme l'indique (4.36). Nous renforçons le sens du service sur les arêtes grâce à (4.37). Le lien entre les liaisons inter-clients et les arêtes est fait avec (4.38) et (4.39). Finalement, (4.40) s'occupe du dépôt matérialisé par les deux sommets fictifs s et t.

L'exemple introductif

La construction de la solution optimale par CPLEX est résumée dans la figure 4.8.

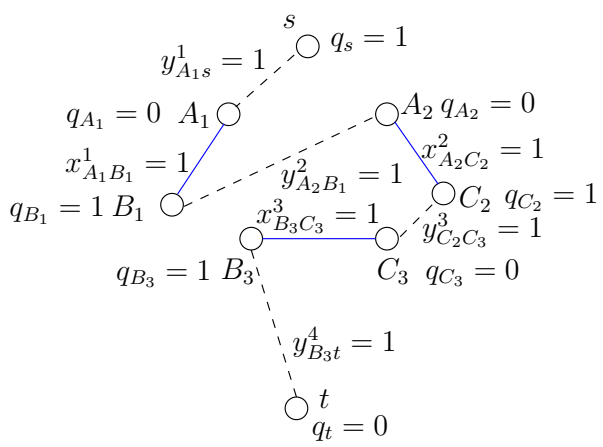


FIGURE 4.8 Une solution optimale de l'exemple introductif pour le modèle L7.

Mentionnons qu'en introduisant des variables q^k_S comme sur la figure 4.9, nous ne sommes pas parvenus ni à obtenir de meilleures bornes ni à obtenir de nouvelles coupes.

$$q^k_S = \begin{cases} 1 & \text{si le sommet } S \in \overline{\mathrm{V}} \cup \{s,t\} \text{ est le dernier sommet} \\ & \text{d'une arête desservie à l'étape } k. \\ 0 & \text{sinon.} \end{cases}$$

La figure 4.9 illustre les variables q^k_S.

4.1.7 L8

Ce modèle est notre modèle principal, celui que nous avons finalement retenu pour résoudre le problème du postier chinois cumulatif. Il a été obtenu conjointement avec les

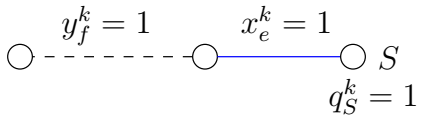


FIGURE 4.9 Les variables q_S^k pour le modèle L7.

professeurs Ángel Corberán de l'Université de Valence et José María Sanchís de l'Université Polytechnique de Valence. Cette fois-ci, nous forçons le sens de passage et sur les arêtes et sur les liaisons inter-clients en utilisant des variables qui tiennent compte du sens de passage. Il a été obtenu à partir du modèle L7 dont nous avons retiré les variables dépendantes x_e^k et q_S. Puis nous avons orienté les variables au sens où nous avons deux jeux de variables pour toute paire de sommets (i,j).

Nous appelons le graphe auxiliaire mixte que nous utilisons dans l'élaboration du modèle L8, le *graphe auxiliaire éclaté*.

Le graphe auxiliaire éclaté

La figure 4.10 montre le graphe original à gauche et le graphe auxiliaire éclaté à droite.

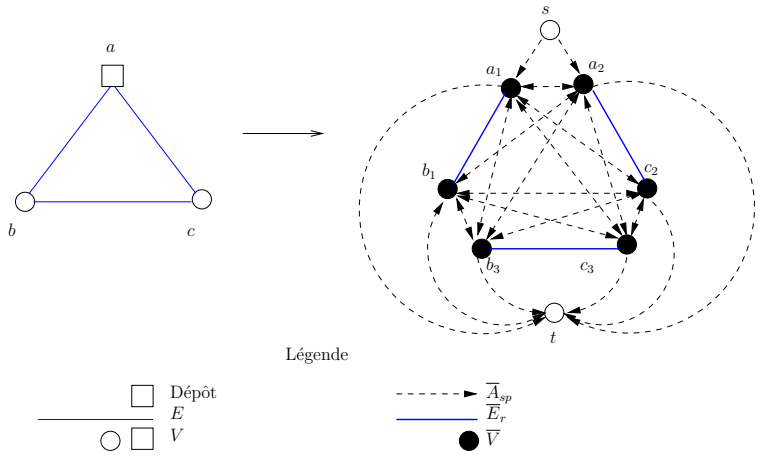


FIGURE 4.10 Le graphe auxiliaire éclaté du modèle L8.

Le graphe auxiliaire éclaté est construit comme suit. Pour chaque arête $e \in$ E, il existe

une arête non dirigée correspondante $\bar{e} \in \overline{E}_r$, de sorte que $|\mathrm{E}| = |\overline{E}_r| = m$. Les sommets correspondants forment l'ensemble $\overline{V}$ et celui-ci est donc de cardinalité $|\overline{V}| = 2m$. Nous rajoutons des arcs dirigés entre chaque paire de sommets de $\overline{V}$ qui ne sont pas reliés pas une arête de $\overline{E}_r$. Ces arcs sont au nombre de $4m(m-1)$ et correspondent aux plus courts chemins. Finalement, deux sommets fictifs s et t ainsi que des arcs dirigés du sommet fictif s vers les sommets correspondants au dépôt et des arcs de chacun des sommets de $\overline{V}$ vers le sommet fictif t sont rajoutés. Il y a deg(dépôt) + $|\overline{V}|$ arêtes dirigées de la sorte. Les arêtes dirigées constituent l'ensemble $\overline{A}_{sp}$.

Au total, dans le graphe auxiliaire éclaté, il y a $2m+2$ sommets, $|\overline{E}_r| = m$ arêtes non dirigées qui correspondent aux arêtes originales et $|\overline{A}_{sp}| = 4m(m-1) + \deg(\text{dépôt}) + |\overline{V}|$ arcs qui correspondent à des plus courts chemins.

Sur la figure 4.10, nous avons $\overline{V} = \{a_1,a_2,b_1,b_3,c_2,c_3\}$. Les sommets s et t ne font pas partie de l'ensemble $\overline{V}$. L'ensemble $\overline{E}_r$ représente les arêtes du graphe auxiliaire éclaté qui correspondent aux arêtes originales : $\overline{E}_r = \{(a_1,b_1),(a_2,c_2),(b_3,c_3)\}$. Parfois nous aurons besoin de connaître l'orientation de ces arêtes, nous utiliserons alors l'ensemble $\overline{A}_r = \{(a_1,b_1),(b_1,a_1),(a_2,c_2),(c_2,a_2),(b_3,c_3),(c_3,b_3)\}$

Quant aux plus courts chemins, les voici :

$$\begin{aligned}\overline{A}_{sp} = \{&(a_1,a_2),(a_2,a_1),(a_1,b_3),(b_3,a_1),(a_1,c_2),(c_2,a_1),(a_1,c_3),(c_3,a_1),(a_2,b_1),(b_1,a_2),\\ &(a_2,b_3),(b_3,a_2),(a_2,c_3),(c_3,a_2),(b_1,b_3),(b_3,b_1),(b_1,c_2),(c_2,b_1),(b_1,c_3),(c_3,b_1),\\ &(b_3,c_2),(c_2,b_3),(c_2,c_3),(c_3,c_2),(s,a_1),(s,a_2),(a_1,t),(a_2,t),(b_1,t),(b_3,t),(c_2,t),(c_3,t)\}\end{aligned}$$

Pour parler facilement de correspondances entre le graphe original et le graphe auxiliaire éclaté, voici deux conventions bien utiles.

Pour tout sommet $i \in \overline{V}$, $*i \in V$ représente le sommet correspondant dans le graphe d'origine. Par exemple sur la figure 4.10, le sommet $*a_1$ est le sommet a du graphe original.

Pour tout sommet $i \in \overline{V}$, $\bar{\imath} \in \overline{V}$ représente le sommet adjacent de l'arête $(i,\bar{\imath}) \in \overline{E}_r$, i.e. $(*i, *\bar{\imath}) \in \mathrm{E}$. Par exemple, sur la figure 4.10, l'arête $(a_1,\overline{a_1})$ est l'arête (a_1,b_1).

Nous voici armés pour décrire le modèle.

Les variables

Pour chaque liaison inter-clients entre les sommets $i,j \in \overline{V}$ qui ne sont pas reliés par une arête de $\overline{E}_r$, il y a deux jeux de variables : y_{ij}^k et y_{ji}^k.

- Pour les sommets $i,j \in \overline{V}$:

$$y_{ij}^k = \begin{cases} 1 & \text{si on utilise la liaison inter-clients } i-j \text{ en } k^e \text{ position,} \\ & \text{ce qui implique que l'arête originale } (j,\bar{j}) \text{ est desservie en } k^e \text{ position .} \\ 0 & \text{sinon.} \end{cases}$$

- Pour les sommets s et $t \notin \overline{V}$:
  - Pour le dépôt, le sommet s est relié uniquement aux sommets représentant le dépôt :

$$y_{sD}^1 = \begin{cases} 1 & \text{si on utilise le dépôt représenté par } D. \\ 0 & \text{sinon.} \end{cases}$$

  - Pour tout sommet $v \in \overline{V}$:

$$y_{vt}^{m+1} = \begin{cases} 1 & \text{si le sommet } v \in \overline{V} \text{ est le dernier parcouru avant } t. \\ 0 & \text{sinon.} \end{cases}$$

Le coût des variables

Soient $d : \mathrm{V} \times \mathrm{V} \to \mathbb{N} : (v_1,v_2) \mapsto d(v_1,v_2)$ la fonction qui retourne le coût des plus courts chemins entre deux sommets de V et $c : \mathrm{V} \times \mathrm{V} \to \mathbb{N}_0 : (v_1,v_2) \mapsto c(v_1,v_2)$ la fonction qui retourne le coût des arêtes du graphe original.

Les arcs reliant s aux sommets représentant le dépôt ainsi que ceux qui relient tous les sommets $v \in \overline{V}$ à t ont un coût nul. Les coûts $\bar{c}$ sur les arcs $(i,j) \in \overline{A}_{sp}$ restants sont les coûts $d(*i, *j)$ des plus courts chemins entre les sommets correspondants dans le graphe original plus le coût $c(*j, *\bar{j})$ de l'arête originale $(*j, *\bar{j}) \in \mathrm{E}$ du graphe original :

$$\bar{c}(i,j) = d(*i, *j) + c(*j, *\bar{j}).$$

Le modèle

$$\min_{y_{ij}^k} \sum_{k=1}^{m} \sum_{(i,j)\in\overline{A}_{sp}} (m-k+1)\cdot \bar{c}(i,j)\cdot y_{ij}^k$$

s.à :

$$y_{sv}^1 + \sum_{k=2}^{m}\Big(\sum_{(v,i)\in\overline{A}_{sp}} y_{vi}^k + \sum_{(j,v)\in\overline{A}_{sp}} y_{jv}^k\Big) + y_{vt}^{m+1} = 1 \qquad \forall\, v\in\overline{V} \qquad (4.41)$$

$$\sum_{(i,j)\in\overline{A}_{sp}} y_{ij}^k = 1 \qquad \forall\, k\in\{1,\ldots,m+1\} \qquad (4.42)$$

$$\sum_{(p,i)\in\overline{A}_{sp}} y_{pi}^k = \sum_{(\bar{\imath},q)\in\overline{A}_{sp}} y_{\bar{\imath}q}^{k+1} \qquad \forall\, i\in\overline{V}, \forall\, k\in\{1,\ldots,m\} \qquad (4.43)$$

et les toutes les variables y_{ij}^k sont binaires.

(4.41) impose que chaque arête originale $(v,\bar{v})$ soit desservie. Il y a une certaine redondance en exigeant que cette contrainte soit respectée pour chaque sommet $v\in\overline{V}$. Le premier terme y_{sv}^1 n'est à rajouter que si $v\in\overline{V}$ représente un dépôt. La contrainte (4.42) dicte qu'à chaque période k, un plus court chemin doit être visité. Seule une contrainte pour une période donnée est véritablement nécessaire. Finalement, les équations de flots (4.43) permettent d'obtenir une tournée. (4.43) est une façon concise d'écrire l'égalité qu'il faut comprendre pour $k=1$ comme $\sum_{(s,i)\in\overline{A}_{sp}} y_{si}^1 = \sum_{(\bar{\imath},q)\in\overline{A}_{sp}} y_{\bar{\imath}q}^2$ et pour $k=m$ comme $\sum_{(p,j)\in\overline{A}_{sp}} y_{pj}^m = \sum_{(\bar{\jmath},t)\in\overline{A}_{sp}} y_{\bar{\jmath}t}^{m+1}$.

Notons que le dépôt est encodé directement dans le choix des arêtes $(i,j)\in\overline{A}_{sp}$.

Ce modèle est très similaire à la formulation à 3 indices de Picard et Queyranne pour le problème du voyageur de commerce dépendant du temps (Picard et Queyranne, 1978).

L'exemple introductif

La construction de la solution optimale par CPLEX est résumée dans la figure 4.11.

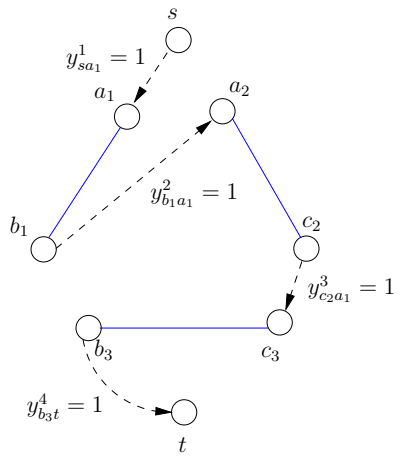


FIGURE 4.11 Une solution optimale détaillée de l'exemple introductif pour le modèle L8.

4.2 Comparaisons sommaires des modèles

Nous faisons ici une première comparaison empirique des modèles présentés avant de passer dans la section 4.3 suivante à des comparaisons théoriques plus détaillées.

Nous utilisons quinze graphes[5] de référence appelés *g1* à *g15*. Les cinq premiers graphes ont été construits pour refléter diverses situations possibles. Passons-les en revue pour en spécifier les caractéristiques. Le graphe *g1* n'est rien d'autre que le graphe de l'exemple introductif de la page 8. Sa caractéristique est d'être un graphe eulérien mais dont le coût exorbitant d'une de ses arêtes empêche un chemin eulérien simple comme solution. Le graphe *g2* a été conçu de telle sorte que la solution optimale passe m fois sur la première arête desservie, $m-1$ fois sur la deuxième arête desservie et ainsi de suite. Le graphe *g3* est une grille unitaire modélisant typiquement nos villes nord-américaines. Le graphe *g4* possède des coûts sur ses arêtes qui diffèrent tous par au moins une puissance de 10 forçant une tournée optimale à desservir les arêtes par coût croissant et a une structure assez spéciale. Quant au graphe *g5*, il possède une arête directement accessible depuis le dépôt mais avec un coût tellement élevé qu'elle doit être desservie en dernier. Les dix autres graphes ont été générés au hasard avec des coûts compris entre 1 et 100 et l'ensemble de ces 15 graphes a été choisi pour illustrer toutes les relations de dominance entre les huit modèles.

5. La lectrice pourra se référer à l'annexe A page 192 pour plus de détails. En particulier, le tableau A.1 reprend les caractéristiques les plus communes de ces quinze graphes.

4.2.1 Comparaison du nombre de variables et de contraintes

Une première étude consiste à comparer les paramètres les plus évidents des modèles : le nombre de variables et le nombre de contraintes. Le tableau 4.1 présente un comptage sommaire de ces nombres.

TABLEAU 4.1 Bornes supérieures[a b] sur le nombre de variables et de contraintes pour chacun des modèles proposés

Modèles	**#Variables**	**#Contraintes**
L1	$2m^2 + mn$	$m^2 + mn + 3m$
L2	$4m^2 + mn$	$\frac{m^3}{2} + \frac{3m^2}{2} + 2mn + 2m - n$
L3	$2m^2 + m - 2$	$4m^3 - 10m^2 + 6m$
L4	$4m^2 - 2m + n$	$6m^2 + 4m + 1$
L5	$2m^3 - 2m$	$4m^3 - 4m^2 + 6m + 1$
L6	$2m^3 - m^2 + 2m$	$3m^2 + m + 2n + 1$
L7	$2m^3 + 5m^2 + 4m + 2$	$2mn + 3m + 2n + 2$
L8	$4m^3 - 8m^2 + 6m + n$	$2m^2 + m + n + 1$

[a] Parfois certaines contraintes ou variables facilement éliminables par un pré-traitement sommaire n'ont pas été comptabilisées.
[b] Nous avons systématiquement assumé que le dépôt était de degré n.

Les modèles comprenant moins de variables et de contraintes comme les modèles L1 et L4 peuvent sembler plus attrayants que des modèles ayant nettement plus de variables et de contraintes comme le modèle L8 mais nous ne sommes pas parvenus à utiliser efficacement ces modèles. Comme nous le verrons dans la prochaine section, les *bi* obtenues sont nettement plus mauvaises pour les modèles L1 et L4 que pour le modèle L8 par exemple. Pire, nous ne sommes pas parvenus dans une approche de séparation et évaluation progressive (*Branch and Cut*) à trouver des coupes efficaces alors que le modèle L8 s'y prête bien comme nous le verrons dans les prochains chapitres.

Le nombre de variables des modèles L6-L8 est énorme. Par exemple, pour le graphe complet K_{10} à 10 sommets, le modèle L8 utilise 348 614 variables. Ce nombre astronomique de variables est à mettre en parallèle avec le nombre de variables requis pour toute solution réalisable : 45. Nous y reviendrons par la suite.

4.2.2 Comparaison des relaxations linéaires

Pendant notre recherche d'un meilleur modèle, l'obtention de meilleures *bi* a été un critère important. Les huit modèles choisis sont en fait ceux qui possèdent les meilleures *bi* parmi tous nos modèles.

Introduisons des nouvelles notations pour les *bi*. Notons par z_{rel}(A,G) la *bi* d'un modèle A pour un graphe G donné. Si z_{rel}(B,G) $\leqslant z_{\text{rel}}$(A,G) peu importe le graphe G, nous disons que le modèle A *domine* le modèle B et nous notons cela par $z_{\text{rel}}(\text{B}) \leqslant z_{\text{rel}}(\text{A})$. Notons par z_{opt}(G) la valeur optimale du problème du postier chinois cumulatif pour un graphe G donné.

La figure 4.12 montre les *bi* obtenues par CPLEX pour les huit modèles sur l'ensemble des quinze graphes de référence. Il faut interpréter cette figure comme suit. En abscisse, nous avons les quinze graphes. En ordonnée, nous avons l'écart relatif entre la valeur optimale du problème et la *bi* correspondante obtenue, soit

$$\frac{z_{\text{opt}} - z_{\text{rel}}}{z_{\text{opt}}}.$$

Plus les valeurs sont proches de zéro, meilleures sont les *bi*. Nous pouvons voir au moins deux tendances sur cette figure[6]. La première est que le modèle L8 domine clairement sauf pour le graphe *g3* où il est dominé par le modèle L5. La deuxième est que, à part pour les cinq premiers graphes qui ont des structures très particulières, les modèles L1-L7 ont des comportements relativement similaires.

Dans la figure 4.13, nous avons ordonné les modèles L1-L8 par valeur de plus grande *bi* pour chacun des quinze graphes. Le graphe qui a la meilleure *bi* obtient la première place et a une ordonnée égale à 1 et se trouve donc situé le plus haut. Le modèle qui obtient la deuxième plus grande *bi* obtient la deuxième place et l'ordonnée 2 et ainsi de suite. S'il y a x modèles *ex æquo* à la deuxième place par exemple, nous leur donnons tous la deuxième place et le modèle suivant se voit attribué la $2 + x^{\text{e}}$ place et l'ordonnée correspondante.

Un examen attentif de la figure 4.13 permet de déduire toutes les relations de dominance de la proposition 19 page 87. Nous pouvons nous demander si nous avons bien obtenu toutes les relations de dominance et la réponse est oui comme nous le démontrons dans la section 4.3.

6. La lectrice est renvoyée au tableau B.1 page 193 de l'annexe pour les données arrondies à deux décimales près.

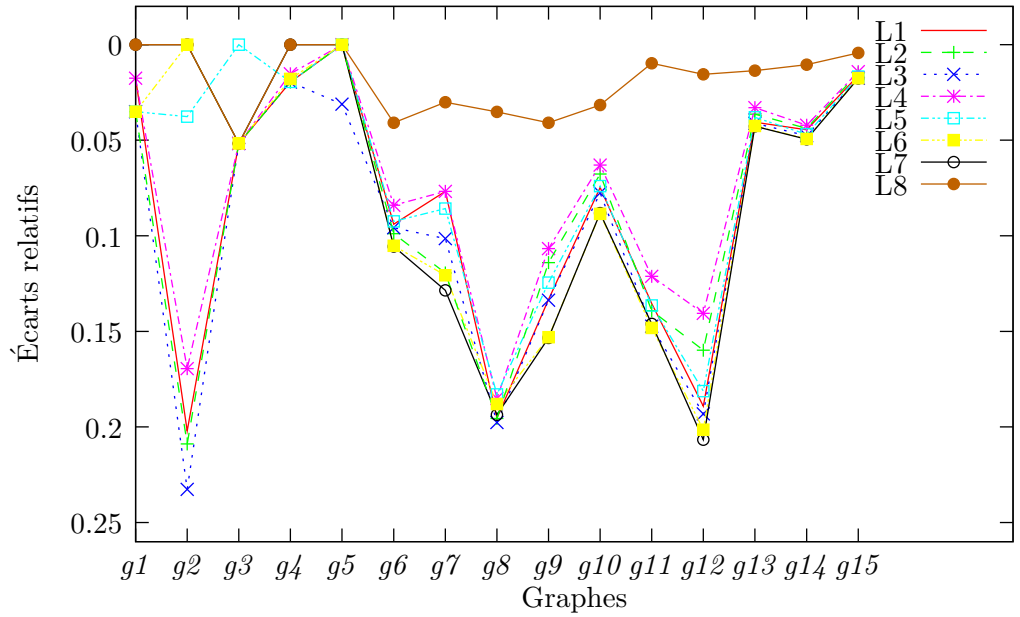


FIGURE 4.12 Les écarts relatifs $\frac{z_{\text{opt}}-z_{\text{rel}}}{z_{\text{opt}}}$ pour les huit modèles sur les quinze graphes de référence. Plus les valeurs sont proches de 0, meilleures sont les *bi*.

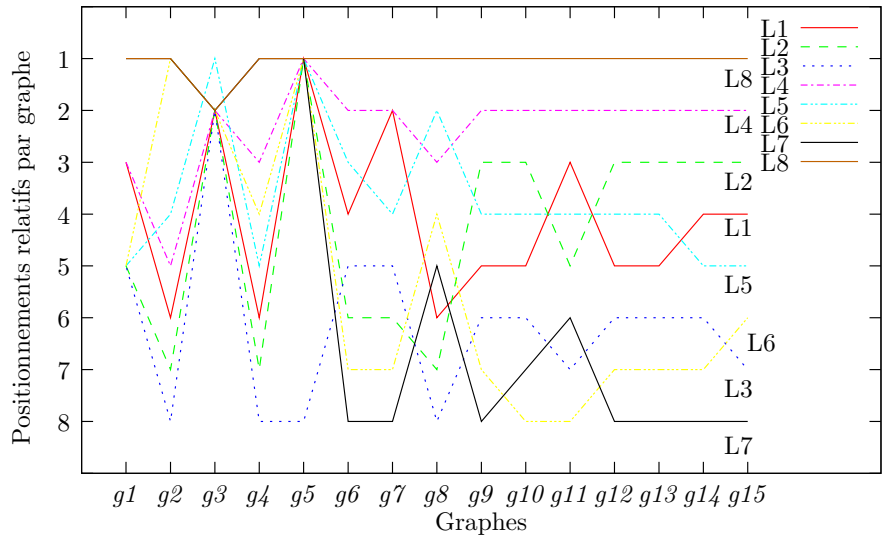


FIGURE 4.13 Positionnements relatifs des *bi* pour les huit modèles.

4.2.3 Comparaison des temps de résolution du PPCC

Nous continuons notre comparaison sommaire en comparant cette fois-ci les temps de résolution pour résoudre le PPCC à l'optimalité. Nous avons utilisé CPLEX sans modification de ses paramètres. La figure 4.14 ne reprend que les temps pour seulement quatre de nos quinze graphes de référence soit parce les temps de résolution sont trop petits (graphes *g1*, *g2*, *g5*, *g7* et *g11*), soit parce que les temps étaient trop longs (graphes *g8-g10* et *g12-g15*). Attirons l'attention de la lectrice sur le fait que l'échelle des ordonnées est logarithmique.

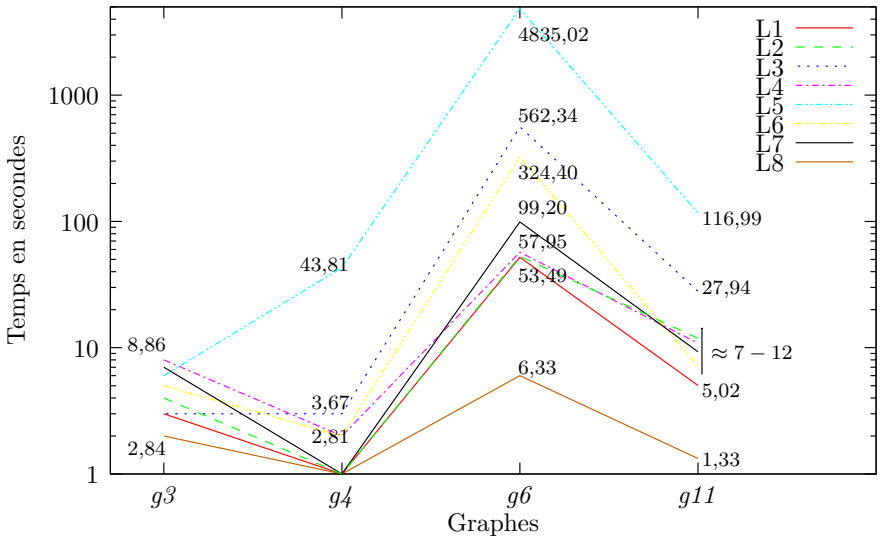


FIGURE 4.14 Temps de résolution en secondes pour les huit modèles sur quatre graphes. L'échelle des ordonnées est logarithmique.

La figure 4.14 nous permet de voir qu'ici aussi, le modèle L8 se comporte mieux que les autres ce qui vu la qualité de ses *bi* n'est guère étonnant. Remarquons aussi que le modèle L5 se démarque par un temps de résolution nettement au-dessus de la moyenne. Par exemple pour le graphe *g6*, il faut approximativement 6 secondes à CPLEX pour résoudre le problème avec le modèle L8 alors qu'il lui faut approximativement 4835 secondes pour résoudre ce même problème avec le modèle L5.

4.3 Comparaisons analytiques des bornes obtenues par relaxation des conditions d'intégralité

Dans cette section, nous démontrons toutes les relations de dominance entre les modèles L1 à L8. Nous invitons d'ailleurs la lectrice à lire sans plus tarder l'énoncé de la propositions 19 page 87 qui résume ces relations. Comme tous les modèles présentés sont linéaires et possèdent les mêmes solutions optimales[7], il nous suffit de démontrer que toute solution[8] d'un modèle A possède une solution équivalente dans un modèle B pour en déduire que $z_{\text{rel}}(\text{B}) \leqslant z_{\text{rel}}(\text{A})$, c'est-à-dire que le modèle A domine le modèle B. Les démonstrations sont toutes basées sur le même schéma. Nous présentons d'abord la transformation qui permet de traduire une solution du modèle A en une solution du modèle B. Puis nous démontrons que la solution ainsi obtenue est bien une solution réalisable pour le modèle B. Souvent nous n'utiliserons qu'un (petit) sous-ensemble de contraintes du modèle A pour démontrer la réalisabilité de la solution pour le modèle B. C'est ce sous-ensemble de contraintes qui permet de mieux comprendre la relation entre les deux modèles.

Nous avons gardé les nomenclatures propres à chacun des modèles y compris quand deux modèles comparés utilisent les mêmes lettres pour désigner différentes familles de variables. Si la lectrice se donne la peine de savourer les démonstrations, le contexte lui permettra de bien saisir de quelles familles il s'agit. Par souci d'économie de papier, certaines équivalences - techniquement exigeantes mais ne représentant qu'un intérêt limité et dont la lectrice peut saisir les nuances rapidement - ont été omises. Terminons en disant que toutes les sommations doivent être comprises dans leur sens le plus étendu. Par exemple, $\sum_p y_{pi}^k$ pour le modèle L8 sous-entend que l'on considère toutes les variables existantes pour un i et un k donnés, y compris les cas extrêmes lorsque $k = 1$ ou $k = m + 1$.

4.3.1 $z_{\text{rel}}(\text{L1}) \leq z_{\text{rel}}(\text{L4})$

Lemme 11 $z_{rel}(L1) \leq z_{rel}(L4)$

Démonstration

Soit $\{x_{ij}^k, q_i^k\}$ une solution relaxée du modèle L4. Construisons une solution correspondante de L1 à partir de celle-ci et vérifions qu'elle vérifie bien toutes les

7. Ceci n'est pas démontré mais considéré comme acquis dans cette thèse. Nous renvoyons la lectrice à l'annexe C page 194 pour un exemple de démonstration de l'équivalence entre les modèles L1, L2 et L3.

8. Parfois nous ne considérerons que des solutions optimales.

contraintes du modèle L1. Nous construisons la solution apparentée comme suit :

$$\begin{aligned} v_{ij}^k &= q_{ij}^k + q_{ji}^k \\ d_{ij}^k &= \sum_{a \in \bar{V}} (x_{a,ij}^k + x_{a,ji}^k) \\ V_v^k &= \sum_{i \in V} q_{iv}^k \end{aligned}$$

(1.1) est respectée grâce à (4.12) :

$$\sum_{k=1}^{m} v_e^k = \sum_{k=1}^{m} (q_{e_1}^k + q_{e_2}^k) \overset{(4.12)}{=} 1.$$

(1.2) est respectée grâce à (4.13) :

$$\sum_{e \in E} v_e^k = \sum_{a \in V} q_a^k \overset{(4.13)}{=} 1.$$

(1.3) $v_e^k \leqslant d_e^k$ devient

$$q_{ij}^k + q_{ji}^k \leqslant \sum_{a \in \bar{V}} (x_{a,ij}^k + x_{a,ji}^k)$$

qui est vrai par (4.16).

(1.5) $\sum_{v \in \mathrm{V}} V_v^k = 1$ est assurée par (4.13) :

$$\sum_{v \in \mathrm{V}} (\sum_{i \in V} q_{iv}^k) = \sum_{a \in \bar{V}} q_a^k \overset{(4.13)}{=} 1.$$

(1.6) et (1.7) s'obtiennent grâce à (4.18). Quant à (1.4) elle est automatiquement respectée puisqu'elle nous donne les variables D_v^k en fonction des variables v_e^k,d_e^k et V_v^k qui respectent les autres contraintes du modèle L1.

□

4.3.2 $z_{\mathbf{rel}}(\mathbf{L3}) \leq z_{\mathbf{rel}}(\mathbf{L5})$

Lemme 12 $z_{rel}(L3) \leq z_{rel}(L5)$

Démonstration

Soit $\{v_e^k, v_f^k\}$ une solution relaxée du modèle L5. Construisons une solution de L3 de telle sorte que

$$y_{a_1}^k + y_{a_2}^k = v_a^k - v_a^{k-1} \geqslant 0.$$

Posons $v_a^0 = 0 \; \forall \, a \in E$. Dès lors, (4.8) est respectée car

$$\sum_{k=1}^{m} (y_{a_1}^k + y_{a_2}^k) = \sum_{k=1}^{m} (v_a^k - v_a^{k-1}) = v_a^m$$

et $v_a^m = 1 \; \forall \, a \in E$ car c'est la seule possibilité pour avoir à la fois $v_a^m \in [0,1]$ et (4.19) $\sum_{e \in \mathrm{E}} v_e^m = m$ puisque $|E| = m$.

(4.9) est respectée aussi grâce à (4.19) car

$$\begin{aligned} \sum_{a \in \bar{V}} y_a^k &= \sum_{e \in E} (y_{e_1}^k + y_{e_2}^k) = \sum_{e \in E} (v_e^k - v_e^{k-1}) \\ &= \sum_{e \in E} v_e^k - \sum_{e \in E} v_e^{k-1} \overset{(4.19)}{=} k - (k-1) = 1. \end{aligned}$$

(4.10) est automatiquement respectée puisqu'il s'agit d'une contrainte qui définit les valeurs des variables z^k.

(4.11) $\sum_{\substack{(ij) \in E, a \in \bar{V} \\ *i \text{ est le dépôt}}} y_{ij,a}^1 = 1$ est l'équivalent de (4.24) $\sum_{f \in \overline{\mathrm{E}}; s \in f} v_f^1 = 1$. En effet, (4.24) impose que $\sum_{\substack{(ij) \in E \\ *i \text{ est le dépôt}}} v_{ij}^1 = 1$ par (4.22) pour $k = 1$ et (4.23) pour $k = 2$. Par conséquent, la configuration du graphe auxiliaire du modèle L3 nous permet de déduire (4.11).

□

4.3.3 $z_{\mathbf{rel}}(\mathbf{L3}) \leq z_{\mathbf{rel}}(\mathbf{L1})$

Lemme 13 *$z_{rel}(L3) \leq z_{rel}(L1)$*

Démonstration

Soit $\{v_e^k, d_e^k, V_v^k, D_v^k\}$ une solution relaxée du modèle L1. Construisons une so-

lution de L3 de telle sorte que

$$y_{ij}^k + y_{ji}^k = v_{ij}^k.$$

Cette solution respecte les contraintes du modèle L3. En effet, (4.8) est respectée grâce à (1.1) :

$$\sum_{k=1}^{m}(y_{ij}^k + y_{ji}^k) = \sum_{k=1}^{m} v_{ij}^k \overset{(1.1)}{=} 1.$$

(4.9) s'obtient avec (1.2) :

$$\sum_{a\in\bar{V}} y_a^k = \sum_{(ij)\in E} v_{ij}^k \overset{(1.2)}{=} 1.$$

(4.11) est respectée par (1.6) et (1.7). (4.10) nous donne une valeur pour les z^k et est donc automatiquement respectée.

□

4.3.4 $z_{\mathbf{rel}}(\mathbf{L3}) \leq z_{\mathbf{rel}}(\mathbf{L4})$

Lemme 14 $z_{rel}(L3) \leq z_{rel}(L4)$

Démonstration

Soit $\{q_a^k, x_{ij}^k\}$ une solution relaxée du modèle L4. Nous pouvons utiliser directement cette solution dans le modèle L3 :

$$y_a^k = q_a^k$$

et (4.8) est l'équivalent de (4.12) tout comme (4.9) est l'équivalent de (4.13).

(4.10) est respectée automatiquement et (4.18) force (4.11).

□

4.3.5 $z_{\mathbf{rel}}(\mathbf{L2}) \leq z_{\mathbf{rel}}(\mathbf{L4})$

Avant de nous lancer dans la démonstration, faisons une remarque sur une propriété des solutions optimales relaxée du modèle L4. La figure 4.15 illustre une telle solution typique du modèle L4 pour le graphe de l'exemple introductif de la page 8.

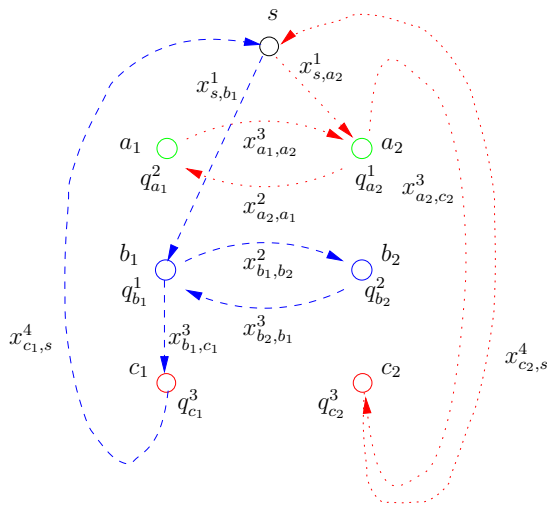


FIGURE 4.15 Une solution relaxée optimale typique pour le modèle L4. Toutes les variables sont à $\frac{1}{2}$.

Soit une solution optimale relaxée. Considérons un sommet $a \in \bar{V}$. Deux cas pour les variables q^k_a se présentent. Soit $q^k_a = 0$. (4.15) impose que $\sum_i x^k_{ia} \leqslant \sum_j x^k_{aj}$. Comme la fonction objectif est linéaire, que nous minimisons et que toutes les autres contraintes du modèle sont respectées à égalité, nous avons

$$\sum_i x^k_{ia} = \sum_j x^k_{aj}. \tag{4.44}$$

Soit $q^k_a = \alpha > 0$. (4.16) et (4.17) imposent que $\alpha \leqslant \sum_i x^k_{ia}$ et $\alpha \leqslant \sum_j x^{k+1}_{aj}$. Par le même raisonnement, nous avons dans toute solution optimale relaxée :

$$\sum_i x^k_{ia} = \alpha = \sum_j x^{k+1}_{aj} = q^k_a. \tag{4.45}$$

Ce sont les seules deux possibilités pour une solution optimale relaxée. Dès lors, celle-ci consiste en une succession d'arcs respectant une certaine conservation de flots au sens de (4.44) et (4.45). Nous appelons ces successions d'arcs des *solutions partielles* pour le

modèle L4. Nous avons deux telles solutions partielles sur la figure 4.15 : une en tirets et une autre en pointillés.

Cette propriété se retrouve transposée sur le graphe original aussi car l'ensemble $\bar{A}$ des arcs permis dans le graphe auxiliaire du modèle L4 sont des arcs entrants et sortants d'un même sommet : $x^k_{bc,de}$ n'existe que si $c = d$. Si $q^k_{uv} = 0$, nous avons (4.44) qui se traduit sur le graphe original par un passage à l'étape k sur l'arc $a = (u,v)$ et sur un certain arc $j = (v,w)$. Donc $q^k_{uv} = 0$ exprime que le sommet $v \in V$ est un sommet interne d'un diamètre si $\sum_a x^k_{a,uv} > 0$. Soit $q^k_{uv} = \alpha > 0$. (4.45) se traduit sur le graphe original par un passage à l'étape k sur $a = (u,v)$ puis un passage à l'étape $k+1$ sur un certain arc $j = (v,w)$. Autrement dit, v est l'extrémité finale d'un diamètre et l'on dessert l'arc (v,w) à l'étape k, ce qui est exactement la définition de $q^k_{u,v}$ lorsque nous considérons l'intégralité. Ce sont ces propriétés qui font que le modèle L4 domine tant de modèles. Nous pouvons maintenant passer à la démonstration du lemme suivant.

Lemme 15 $z_{rel}(L2) \leq z_{rel}(L4)$

Démonstration

Soit $\{x^k_{ij}, q^k_a\}$ une solution relaxée optimale du modèle L4. Nous construisons à partir de celle-ci une solution réalisable pour le modèle L2 comme suit :

$$y^k_{ij} = \sum_{l=1}^{k} (q^l_{ij} + q^l_{ji})$$
$$x^k_{iT} = \sum_{v \in V} q^k_{vi}$$
$$x^k_{ij} = \sum_{l=1}^{k} \sum_{a \in \bar{V}} x^l_{a,ij}$$

Vérifions maintenant qu'une telle solution est bien réalisable pour le modèle L2.

(4.1) est vérifiée par construction.

Pour voir que (4.2) est respectée, considérons deux cas. Premier cas, le sommet i n'est pas le dépôt et pour notre solution, (4.2) devient

$$\underbrace{\sum_{v\in V} q_{vi}^k}_{\alpha} + \underbrace{\sum_{j\in\delta_G^+(i)} (\sum_{l=1}^{k}\sum_{a\in\bar{V}} x_{a,ij}^l)}_{\beta} - \underbrace{\sum_{j\in\delta_G^-(i)} (\sum_{l=1}^{k}\sum_{a\in\bar{V}} x_{a,ji}^l)}_{\gamma} = 0 \tag{4.46}$$

Cette somme est bien nulle. Si i est l'extrémité finale d'un diamètre et que l'on vient de desservir l'arc (v,i) alors $q_{vi}^k = \theta \in \alpha$. (4.16) et notre remarque plus haut imposent qu'il existe un $x_{a,vi}^k = \theta \in \gamma$ et donc les deux termes s'annulent dans (4.46). Nous pouvons donc nous débarrasser de la somme α si nous éliminons deux à deux tous ces termes dans (4.46). Montrons que pour chacun des termes restants de la somme β il existe un terme correspondant dans les termes restants de la somme γ et vice-versa.

Soit $x_{a,ij}^l$ un terme restant de la somme β. Soit i est l'extrémité finale d'un diamètre à l'étape $l-1$, dans ce cas nous avons vu qu'il existe un terme $x_{b,a}^{l-1}$ et par la remarque plus haut, les deux termes s'annulent dans (4.46). Soit[9] i est un sommet interne d'un diamètre à l'étape l. Nous sommes dans le cas où $\sum_v q_{vi}^l = 0$. Soit $x_{ui,iv}^l$ un terme de la somme β. Alors il existe forcément un $x_{b,ui}^l$ correspondant et de même valeur dans la somme γ qui l'annule dans (4.46) par la remarque plus haut.

Nous pouvons faire le même raisonnement pour tout terme de la somme γ de sorte que nous venons de démontrer que (4.46) est respectée lorsque le sommet i n'est pas le dépôt.

Considérons maintenant le second cas : i est le dépôt. La seule différence avec un sommet qui n'est pas le dépôt est qu'à l'étape 1, nous héritons des termes $\sum_v x_{s,iv}^1 = 1$ par (4.18). Autrement dit, (4.2) sera toujours respectée et égale à 1 dans ce cas.

9. i ne peut pas être le dernier sommet d'un diamètre à l'étape l puisque nous considérons $x_{a,ij}^l$.

Pour notre solution, (4.3) est respectée grâce à (4.13) et s'écrit

$$\sum_i \sum_{v \in V} q_{vi}^k = \sum_{a \in \bar{V}} q_a^k \overset{(4.13)}{=} 1.$$

(4.4) s'écrit pour notre solution

$$\sum_{v \in V} q_{vi}^k + \sum_{j \in \delta_G^+(i)} (\sum_{l=1}^{k} \sum_{a \in \bar{V}} x_{a,ij}^l) \leqslant \sum_{j \in \delta_G^+(i)} (\sum_{l=1}^{k+1} \sum_{a \in \bar{V}} x_{a,ij}^l)$$

ce que nous pouvons simplifier en

$$\sum_{v \in V} q_{vi}^k \leqslant \sum_{j \in \delta_G^+(i)} \sum_{a \in \bar{V}} x_{a,ij}^{k+1}$$

qui est respectée et n'est rien d'autre qu'une sommation sur (4.17).

(4.5) se traduit pour notre solution par

$$\sum_{l=1}^{k} q_{ij}^l + \sum_{l=1}^{k} q_{ji}^l \leq \sum_{l=1}^{k} \sum_{a \in \bar{V}} x_{a,ij}^l + \sum_{l=1}^{k} \sum_{a \in \bar{V}} x_{a,ji}^l.$$

Par (4.16) nous avons $q_{ij}^l \leqslant \sum_{a \in \bar{V}} x_{a,ij}^l$ et $q_{ji}^l \leqslant \sum_{a \in \bar{V}} x_{a,ji}^l$. En sommant, nous obtenons bien (4.5).

(4.6) se traduit par

$$\sum_{l=1}^{k} \sum_{a \in \bar{V}} x_{a,ij}^l + \sum_{l=1}^{k} \sum_{a \in \bar{V}} x_{a,ji}^l \leqslant m \cdot \sum_{l=1}^{k} (q_{ij}^l + q_{ji}^l). \tag{4.47}$$

Soit k fixé. Considérons deux cas. Soit $\sum_{l=1}^{k}(q_{ij}^l + q_{ji}^l) = 1$ et le membre de gauche de (4.47) est m et (4.47) est automatiquement vérifiée. Soit $\sum_{l=1}^{k}(q_{ij}^l + q_{ji}^l) = \alpha < 1$ mais dans ce cas, dès qu'un flot $x_{a,ij}^{k+p} = \beta$ entre par exemple en (ij), par le lemme 2 page 6 qui stipule que dans toute solution optimale on dessert une arête avant de la parcourir, ce flot desservira la portion de l'arc (i,j) pour une valeur de β et $q_{ij}^{k+p} = \beta$ et donc (4.47) est vérifiée.

Finalement, (4.7) s'obtient grâce à (4.13) :

$$\begin{aligned}\sum_{\substack{i<j\\(i,j)\in \mathrm{E}}} y_{ij}^k &= \sum_{\substack{i<j\\(i,j)\in \mathrm{E}}} (\sum_{l=1}^{k} q_{ij}^l + \sum_{l=1}^{k} q_{ji}^l)\\ &= \sum_{l=1}^{k}(\sum_{\substack{i<j\\(i,j)\in \mathrm{E}}} (q_{ij}^l + q_{ji}^l)) = \sum_{l=1}^{k}\sum_{a\in\bar{V}} q_a^k\\ &\overset{(4.13)}{=} \sum_{l=1}^{k} 1 = k.\end{aligned}$$

□

4.3.6 $z_{\mathbf{rel}}(\mathbf{L6}) \leq z_{\mathbf{rel}}(\mathbf{L8})$

Lemme 16 $z_{rel}(L6) \leq z_{rel}(L8)$

Démonstration

Pour pouvoir différencier facilement les variables du modèle L6 de celles du modèle L8, nous surmontons les variables du modèle L6 par un tiret. Ainsi, $\bar{y}_{ij}^k$ représente une variable du modèle L6 alors que y_{ij}^k représente une variable du modèle L8.

Soit $\{y_{ij}^k\}$ une solution relaxée du modèle L8. Nous fabriquons alors la solution suivante pour le modèle L6 :

$$\begin{aligned}\bar{x}_{ij}^k &= \frac{1}{2}\left[\sum_p y_{pi}^k + \sum_q y_{iq}^{k+1}\right] + \frac{1}{2}\left[\sum_p y_{pj}^k + \sum_q y_{jq}^{k+1}\right]\\ \bar{y}_{ij}^k &= y_{ij}^k + y_{ji}^k\end{aligned}$$

Montrons maintenant qu'une telle solution est réalisable pour le modèle L6.

(4.25) est bien égal à 1 grâce à (4.41) :

$$\frac{1}{2}\left[\sum_{k=1}^{m}\left(\sum_p y_{pi}^k + \sum_q y_{iq}^{k+1}\right)\right] + \frac{1}{2}\left[\sum_{k=1}^{m}\left(\sum_p y_{pj}^k + \sum_q y_{jq}^{k+1}\right)\right] \overset{(4.41)}{=} \frac{1}{2}[1] + \frac{1}{2}[1] = 1.$$

(4.26) est bien égal à 1 aussi mais cette fois grâce à (4.42) :

$$\frac{1}{2}\left[\sum_{(i,j)\in E}\left(\sum_p y_{pi}^k + \sum_p y_{pj}^k\right)\right] + \frac{1}{2}\left[\sum_{i,j)\in E}\left(\sum_q y_{iq}^{k+1} + \sum_q y_{jq}^{k+1}\right)\right]$$
$$= \frac{1}{2}\left(\sum_{(i,j)\in \overline{A}_{sp}} y_{ij}^k\right) + \frac{1}{2}\left(\sum_{(i,j)\in \overline{A}_{sp}} y_{ij}^{k+1}\right) \overset{(4.42)}{=} 1.$$

(4.27) se traduit par

$$\sum_{k=1}^{m}(y_{ij}^k + y_{ji}^k) \overset{(4.41)}{\leqslant} 1$$

puisque la somme des flots en i ou en j doit être égale à 1 par (4.41).

(4.28) devient

$$\sum_{(i,j)\in E}(y_{ij}^k + y_{ji}^k) = \sum_{(i,j)\in \overline{A}_{sp}} y_{ij}^k \overset{(4.42)}{=} 1$$

et est donc bien respectée.

(4.29) et (4.30) sont respectées par construction.

(4.31) nous donne

$$\sum_{k=1}^{m}\sum_{f\in \overline{\mathrm{E}};S\in f} \bar{y}_f^k = \sum_{k=1}^{m}\left(\sum_p y_{pS}^k + \sum_q y_{Sq}^k\right) \overset{(4.41)}{=} 1.$$

Finalement, (4.32) est obtenue par une application directe de (4.42) dans le cas $k = 1$ et la structure du graphe auxiliaire utilisé pour le modèle L8.

□

4.3.7 $z_{\mathbf{rel}}(\mathbf{L7}) \leq z_{\mathbf{rel}}(\mathbf{L8})$

Lemme 17 $z_{rel}(L7) \leq z_{rel}(L8)$

Démonstration

Comme pour la démonstration précédente, nous surmontons les variables du modèle L7 par un tiret pour les distinguer des variables du modèle L8.

Soit $\{y_{ij}^k\}$ une solution relaxée du modèle L8. Construisons une solution réalisable pour le modèle L7 comme suit :

$$\begin{aligned}
\bar{y}_{ij}^k &= y_{ij}^k + y_{ji}^k \\
\bar{x}_{ij}^k &= \frac{1}{2}\left[\sum_p y_{pi}^k + \sum_q y_{iq}^k + \sum_p y_{pj}^k + \sum_q y_{jq}^k\right] \\
\bar{q}_v &= \frac{1}{2} \text{ et } q_s = 1,\ q_t = 1.
\end{aligned}$$

(4.33) est vérifiée grâce à (4.41) :

$$\sum_{k=1}^{m+1} \sum_{f\in\overline{\mathrm{E}};S\in f} \bar{y}_f^k = \sum_{k=1}^{m+1}(\sum_p y_{pS}^k + \sum_q y_{Sq}^k) \overset{(4.41)}{=} 1.$$

(4.34) est aussi vérifiée grâce à (4.41) :

$$\sum_{k=1}^{m} \bar{x}_{ij}^k = \frac{1}{2}\left[\sum_{k=1}^{m}\left(\sum_p y_{pi}^k + \sum_q y_{iq}^k\right) + \sum_{k=1}^{m}\left(\sum_p y_{pj}^k + \sum_q y_{jq}^k\right)\right] \overset{(4.41)}{=} \frac{1}{2}[1+1] = 1.$$

(4.35) est vérifiée grâce à (4.42) :

$$\sum_{f\in\overline{E}} \bar{y}_f^k = \sum_{(i,j)\in\overline{A}_{sp}} y_{ij}^k \overset{(4.42)}{=} 1.$$

(4.36) est aussi vérifiée grâce à (4.42) :

$$\begin{aligned}
\sum_{(i,j)\in E} \bar{x}_{ij}^k &= \frac{1}{2}\left[\sum_{(i,j)\in E}\left(\sum_p y_{pi}^k + \sum_p y_{pj}^k\right) + \sum_{(i,j)\in E}\left(\sum_q y_{iq}^k + \sum_q y_{jq}^k\right)\right] \\
&= \frac{1}{2}\left[\sum_{(i,j)\in\overline{A}_{sp}} y_{ij}^k + \sum_{(i,j)\in\overline{A}_{sp}} y_{ij}^k\right] \overset{(4.42)}{=} 1.
\end{aligned}$$

(4.37) est évidemment vérifiée.

(4.38) et (4.39) sont respectées par construction. Par exemple :

$$\sum_{f\in\overline{\mathrm{E}};S\in f} \bar{y}_f^k - \bar{x}_e^k = \sum_p y_{pi}^k + \sum_q y_{iq}^k - \frac{1}{2}\left(\sum_p y_{pi}^k + \sum_q y_{iq}^k + \sum_p y_{pj}^k + \sum_q y_{jq}^k\right)$$
$$= \frac{1}{2}\left(\sum_p y_{pi}^k + \sum_q y_{iq}^k\right) - \frac{1}{2}\left(\sum_p y_{pj}^k + \sum_q y_{jq}^k\right).$$

Comme d'une part

$$0 \leqslant \sum_p y_{pi}^k + \sum_q y_{iq}^k \leqslant 1$$

et d'autre part

$$0 \leqslant \sum_p y_{pi}^k + \sum_q y_{iq}^k + \sum_p y_{pj}^k + \sum_q y_{jq}^k \leqslant 1$$

nous avons que

$$-\frac{1}{2} \leqslant \frac{1}{2}\left(\sum_p y_{pi}^k + \sum_q y_{iq}^k\right) - \frac{1}{2}\left(\sum_p y_{pj}^k + \sum_q y_{jq}^k\right) \leqslant \frac{1}{2}.$$

et donc (4.38) est respectée. Par un raisonnement similaire, nous pouvons montrer que (4.39) est respectée aussi.

Finalement, (4.40) est l'équivalent de $\bar{q}_s = 1$, $\bar{q}_t = 0$ et (4.41) dans les cas où $k = 1$ et $k + m + 1$.

□

4.3.8 $z_{\mathbf{rel}}(\mathbf{L4}) \leq z_{\mathbf{rel}}(\mathbf{L8})$

Cette dernière démonstration est très technique et demanderait un nombre de pages considérable si nous écrivions explicitement tous les détails. Nous esquissons ici les grandes lignes, la rendant suffisamment intelligible pour que la lectrice puisse suivre notre raisonnement et vérifier par elle-même la justesse de nos propos.

Lemme 18 $z_{rel}(L4) \leqslant z_{rel}(L8)$

Démonstration

À partir d'une solution optimale du modèle L8, nous construisons une solution réalisable correspondante pour le modèle L4. Voici tout d'abord une explication du déroulement de la démonstration. Soit $\{y_{ij}^k\}$ une solution optimale du modèle L8 relaxé. Grâce au lemme 2 (toute arête doit être desservie avant d'être parcourue dans toute solution optimale), nous pouvons déduire qu'une telle solution sera composée de circuits partant du dépôt et revenant à ce dépôt, desservant une portion des arêtes avant de les emprunter et utilisant m arêtes pas forcément distinctes. Ce sont exactement les solutions partielles que le modèle L4 accepte.

Plus exactement, soit une solution relaxée optimale $\{y_{ij}^k\}$ du modèle L8. Les trois contraintes (4.41)-(4.43) du modèle L8 font que nécessairement cette solution est constituée de p circuits partant et retournant au dépôt en passant par m arêtes pas nécessairement distinctes. Comme de plus, il s'agit d'une solution optimale, aucune portion d'arête n'est parcourue avant d'avoir été desservie grâce au lemme 2. Nous appelons ces circuits des *solutions partielles* pour le modèle L8. Si $\alpha = y_{ij}^k > 0$ alors il existe un plus court chemin $i = v_0 - v_1 - v_2 - \ldots - v_a = j$ dans le graphe original tel que chaque arête aura été desservie pour une proportion au moins aussi grande que α :

$$\forall\ v_i\ : \sum_{l=1}^{k_1}(\sum_c y_{c,v_i}^l + \sum_d y_{v_i,d}^l) \geqslant \alpha. \tag{4.48}$$

Ceci est une condition *sine qua non* pour pouvoir construire une solution correspondante du modèle L4. Revenons aux p circuits composés chacun de m arêtes. Nous pouvons leur assigner à chacun un flot f_l qui sera le flot minimum sur tout le circuit. Grâce aux contraintes (4.41)-(4.43) combinées nous pouvons démontrer que $\sum_{l=1}^p f_l = 1$.

Construisons maintenant une solution correspondante pour le modèle L4 comme suit. Pour chacun de ces p circuits, nous construisons une solution partielle correspondante pour le modèle L4. Soit $S = \{y_{s,v_1}^1, y_{v_2,v_3}^2, \ldots, y_{v_{2m-2},v_{2m-1}}^m, y_{v_{2m},t}^{m+1}\}$ une telle solution partielle de la solution optimale du modèle L8 et soit α le flot minimal sur ce chemin : $\alpha = \min\{y_{ij}^k \in S\}$. Pour chacune des variables $y_{ij}^k \in S$

construisons un plus court chemin correspondant dans le graphe original tel que (4.48) soit respectée. Soit $y_{u,v}^{k-1}$ la variable précédent la variable y_{ij}^{k}. Alors on prendra

$$x_{vi,ia}^{k} = x_{ia,ab}^{k} = \ldots = x_{cj,j\bar{j}}^{k} = q_{j\bar{j}}^{k} = \alpha.$$

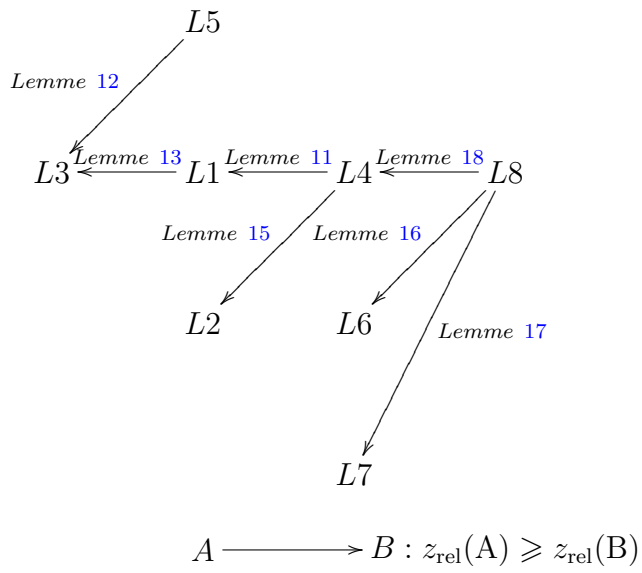
Lorsqu'un $x_{ia,ab}^{k}$ n'est pas nul, nous ajoutons la valeur α à sa valeur.

Par construction et suite à la discussion précédent le lemme 15 page 78, nous pouvons déduire que la solution ainsi construite respecte toutes les contraintes du modèles L4.

□

4.3.9 Relations de dominance entre les modèles

Proposition 19 (Relation de dominance entre les modèles)
Les seules relations de dominance entre les modèles L1, L2, L3, L4 L5, L6, L7 et L8 sont les suivantes :



Démonstration

Toutes les relations exprimées ont été démontrées dans des lemmes cités. Il faut aussi bien sûr inclure les relations obtenues par transitivité. Pour les non relation de dominance, il existe des graphes pour lesquels aucune relation ne

peut être respectée comme nous l'avons vu sur la figure 4.13 de la page 72. Par exemple, nous voyons que pour le graphe *g11* $z_{\text{rel}}(\text{L2}, \textit{g11}) \leqslant z_{\text{rel}}(\text{L1}, \textit{g11})$ alors que pour le graphe *g12* cette relation est inversée. Il ne peut donc pas y avoir de relation de dominance entre les modèles L1 et L2.

□

4.4 Conclusions

Modéliser un problème est un art difficile. Nous avons essayé plusieurs voies pour combiner les deux exigences du problème du postier chinois cumulatif : l'existence d'un chemin partant d'un dépôt et qui passe au moins une fois par toutes les arêtes du graphe et le calcul du coût cumulatif de ses arêtes.

Nos premières tentatives furent de modéliser le problème sur le graphe original en considérant des variables sur les arêtes comme pour le modèle L1 mais très vite nous avons été confronté à deux problèmes. D'une part, les bornes obtenues n'étaient pas très bonnes et d'autre part nous étions incapables de trouver des coupes effectives. Deux problèmes rédhibitoires dans une approche de *Branch and Cut* : non seulement nous partons loin de la valeur de la solution optimale mais en plus nous trainons en chemin.

Nous nous sommes donc tournés vers des modélisations qui exigent la construction d'un graphe auxiliaire. Par exemple en modélisant le problème sur des arcs représentés par des sommets ou bien en utilisant la propriété du lemme 3 qui permet de se concentrer sur l'utilisation de plus courts chemins entre les arêtes desservies. Encore faut-il savoir comment calculer les coûts cumulatifs sur les arêtes. Une tentative fut l'approche du *grand M*, par exemple dans les modèle L10 (voir la section D.1.1 dans l'annexe) ou L3. Si ces modèles sont relativement économes en nombre de variables, ils donnent des bornes médiocres et nous les avons délaissés pour cette raison. Nous nous sommes dès lors concentrés sur des modèles calculant explicitement les coûts cumulatifs comme le modèle L4 et nous nous sommes rendus compte que plus nous mettions de l'information, meilleures étaient les bornes obtenues. C'est ainsi que l'utilisation de plus courts chemins dans la solution comme pour le modèle L6 ou la contrainte de bâtir un chemin partiel à l'étape k à partir du chemin partiel de l'étape $k-1$ comme pour le modèle L5 nous donnent de meilleures bornes que celles obtenues pour les premiers modèles. En poussant plus loin cette logique, nous avons renforcé l'orientation du

service sur les arêtes comme pour le modèle L7 et finalement forcé l'orientation des plus courts chemins dans le modèle L8. A chaque fois, les bornes s'améliorent mais au détriment d'un nombre plus élevé de variables et de contraintes. Le modèle L8 possède ainsi un nombre astronomique de variables et un nombre conséquent de contraintes. Nous verrons dans la suite comment nous avons abordé ce problème d'abondance de variables et de contraintes.

Un autre aspect que nous avons totalement passé sous silence est celui des coupes. Nous avons aussi recherché une modélisation nous permettant d'obtenir facilement des coupes efficaces. Là aussi, nous avons construit le modèle L8 pour obtenir des coupes efficaces comme nous le verrons dans les chapitres suivants.

La proposition 19 montre toutes les relations de dominance entre les huit meilleurs modèles L1-L8 que nous avons développés. Le seul modèle qui ne soit pas dominé par le modèle L8 est le modèle L5. La force du modèle L5 est d'exiger qu'un chemin partiel soit construit à partir d'un chemin partiel de l'étape précédente mais cette exigence se modélise difficilement pour le modèle L8.

Chapitre 5

RÉSOLUTION DU MODÈLE L8 : PREMIÈRE APPROCHE

All that came afterward is just icing on the cake.

Vašek Chvátal (Applegate *et al.*, 1994)

Nous abordons une première approche basée essentiellement sur la parité de certains ensembles pour résoudre le modèle L8. L'idée de la parité est simple : certains sous-ensembles doivent être pair ou impair pour toute solution réalisable et nous pouvons transformer cette idée en des inégalités valides globalement (coupes) ou localement (règles de branchement).

Tout au long de notre recherche, nous avons trouvé plusieurs familles d'inégalités valides pour finalement nous rendre compte que certaines de ces familles n'étaient qu'une autre façon d'exprimer une même idée. De fait, la richesse du modèle L8 nous permet d'exprimer une même idée de différentes façons et c'est un thème récurrent dans ce chapitre et le suivant. Si ces différentes écritures sont équivalentes d'un point de vue théorique, il en va autrement lorsque nous les appliquons comme nos tests expérimentaux le montrent.

Notre premier algorithme pour résoudre le problème du postier chinois cumulatif est un algorithme de séparation et évaluation progressive (Branch and Cut) relativement dépouillé avec le modèle L8. Nous utilisons trois types de branchements, une seule façon de sélectionner les nœuds pendant la recherche peu importe le branchement utilisé, quelques pré-traitements sur les variables, quelques familles d'inégalités ainsi qu'une heuristique pour obtenir une solution réalisable de départ. Cette approche nous permet néanmoins de déjà battre un solveur comme CPLEX par un facteur de 3 à 58 sur nos graphes de référence lorsque nous utilisons notre algorithme avec ce même CPLEX pour résoudre les PL.

Nous commençons ce chapitre par une discussion sommaire sur la dimension du modèle ainsi que la difficulté d'obtenir rigoureusement la dimension de l'enveloppe convexe de ses

solutions réalisables dans la section 5.1. Pour discuter plus facilement de certains concepts, notamment les sommes utilisées, nous utilisons les conventions faites dans la section 5.2. Nous exposons notre heuristique brièvement dans la section 5.3. À la section 5.4, nous montrons comment nous débarrasser de certaines variables par pré-traitement pour après entrer dans le vif du sujet dans la section 5.5 : nos premières inégalités valides efficaces. Dans cette section 5.5, nous voyons également quelles sont les relations entre ces familles d'inégalités valides. Nous développons une certaine généralisation des inégalités de parité sous la forme des inégalités *generalized co-circuit* que nous exposons dans la section 5.6. Nous voyons trois branchements dans la section 5.7 et nous disséquons tous ces ingrédients de façon pratique dans la section 5.8 réservée aux résultats expérimentaux. Finalement, nous concluons dans la section 5.9.

La section 5.5 sur les inégalités valides est le résultat d'une collaboration avec les professeurs Ángel Corberán de l'Université de Valence et José María Sanchís de l'Université Polytechnique de Valence.

5.1 Discussion sommaire du modèle L8

5.1.1 Une surabondance de variables et de contraintes

Ce qui saute aux yeux, c'est le gigantisme du modèle. Le nombre de variables et de contraintes est résumé dans le tableau 5.1. La première colonne donne le nombre exacte de variables et de contraintes pour le modèle alors que la deuxième colonne donne le nombre de variables et de contraintes lorsqu'on considère la période de temps 1 comme une période comme les autres, *i.e.* lorsqu'on considère tous les sommets de $\overline{V}$ comme des dépôts potentiels.

	Modèle de base	Borne supérieure
#var	$4m^3 - 8m^2 + 6m + \deg(\text{dépôt})$	$4m^3 - 8m^2 + 7m$
#cons	$2m^2 + m + 1 + \deg(\text{dépôt})$	$2m^2 + 2m + 1$

Tableau 5.1 Nombres exacts et approchés de variables et de contraintes dans le modèle L8.

Dans le tableau 5.2 nous calculons le nombre de variables et de contraintes que comprend le modèle pour certains graphes complets. Approximativement, pour K_{10^k}, le modèle comporte $\frac{10^{6k}}{2}$ variables et $\frac{10^{4k}}{2}$ contraintes ! Ces nombres astronomiques sont impressionnants et

suggèrent fortement une approche de décomposition.

Graphes	#var	#cons
K_{10}	348 614	4 105
K_{10^2}	484 953 509 799	49 010 050
K_{10^3}	498 499 503 500 997 999	499 001 000 500
K_{10^4}	499 849 995 003 500 099 979 999	4 999 000 100 005 000
K_{10^6}	499 998 499 999 500 003 500 000 999 997 999 999	499 999 000 001 000 000 500 000
K_{10^k}	$\sim \frac{10^{6k}}{2}$	$\sim \frac{10^{4k}}{2}$

TABLEAU 5.2 Nombre de variables et de contraintes pour des graphes complets pour le modèle L8.

Ce sont les contraintes (4.43) qui sont les plus nombreuses comme le montre le tableau 5.3.

Graphes	#Contrainte (4.41) $2m$	#Contrainte (4.42) $m+1$	#Contrainte (4.43) $2m^2-2m+\deg(\text{dépôt})$	Total $2m^2+m+1+\deg(\text{dépôt})$
K_{10}	90	46	3 969	4 105
K_{10^2}	9 900	4 951	48 990 250	49 010 050
K_{10^3}	999 000	499 501	498 999 501 999	499 001 000 500
K_{10^4}	99 990 000	49 995 001	4 998 999 950 019 999	4 999 000 100 005 000
K_{10^k}	$\sim 10^{2k}$	$\sim \frac{10^{2k}}{2}$	$\sim \frac{10^{4k}}{2}$	$\sim \frac{10^{4k}}{2}$

TABLEAU 5.3 Nombre de contraintes détaillé par type de contraintes pour des graphes complets.

Nous nous occupons de réduire le nombre de variables soit par pré-traitement (section 5.4 page 97) soit par génération de colonnes (section 6.1.3 page 149).

5.1.2 La dimension du modèle L8

Disons toute de suite que nos tentatives pour trouver la dimension de l'enveloppe convexe du polyèdre associé au modèle n'ont pas abouti. Nous ne démontrons pas les résultats préliminaires obtenus car nous voulons juste attirer l'attention sur la difficulté de calculer rigoureusement cette dimension et une analyse trop détaillée nous emporterait trop loin. D'une part, on peut démontrer que le polyèdre n'est pas de pleine dimension. En fait, la dimension réelle est sans doute bien plus petite voire un ordre de grandeur plus petite

que $4m^3 - 8m^2 + 6m + \deg(\text{dépôt})$. D'autre part, nos premières recherches nous poussent à croire que la dimension dépend aussi du type de graphes envisagés. Enfin, mentionnons que la recherche de points linéairement ou affinement indépendants n'est pas aisée. Par exemple, des points correspondants à des solutions réalisables ne sont pas forcément linéairement indépendants comme le montre l'exemple sur les figures 5.1 et 5.2.

La figure 5.1 représente un graphe et les arêtes originales du graphe auxiliaire éclaté. L'arête (E,B) a été déplacée pour obtenir des figures plus simples.

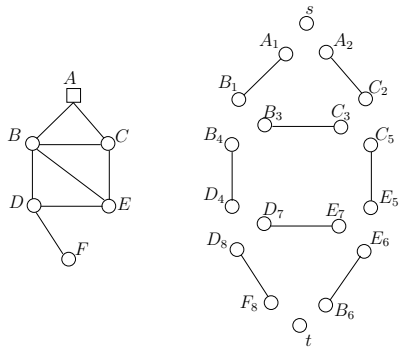


FIGURE 5.1 Un graphe et les arêtes originales de son graphe auxiliaire éclaté. L'arête (E,B) a été déplacée pour obtenir une représentation plus simple sur la figure 5.2.

La figure 5.2 montre comment obtenir la solution réalisable (f) à partir d'une combinaison linéaire des solutions (a)-(e).

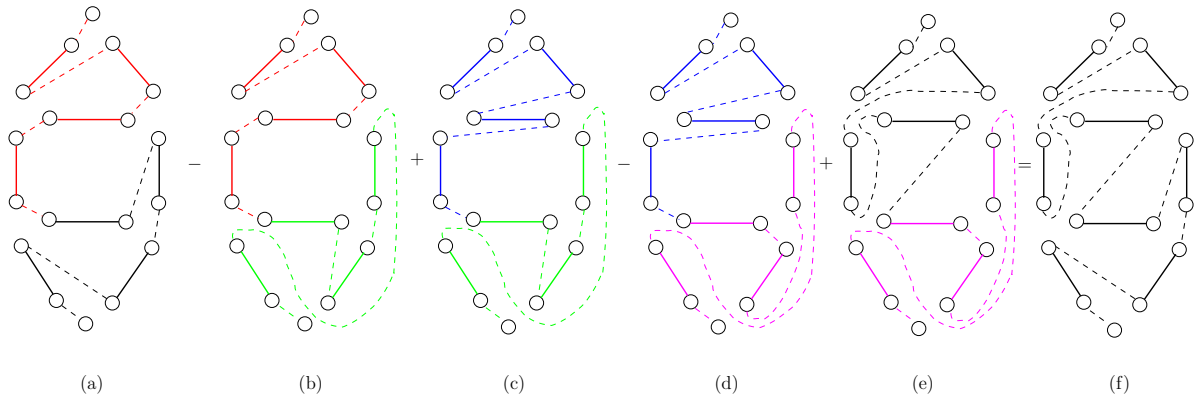


FIGURE 5.2 La solution (f) est obtenue comme combinaison linéaire des solutions (a)-(e).

5.2 Conventions pour le modèle L8

Nous rassemblons ici les conventions que nous utilisons dans le reste de cette thèse. Rappelons que nous avons déjà vu quelques conventions à la section 4.1.7 page 64.

5.2.1 Les sommes

Comme le nombre de variables est important, nous utilisons plusieurs raccourcis d'écriture pour certaines sommes.

Pour tous sommets i et j, nous avons

$$\bar{y}_{ij} = y_{ij} + y_{ji}.$$

Soient $A,B \subset \overline{V} \cup \{s,t\}$ deux ensembles de sommets du graphe auxiliaire éclaté et soit $T \subseteq \{1,\ldots,m+1\}$ un ensemble d'indices de temps, nous avons alors

$$y^k_{A,B} = \sum_{i\in A; j\in B} y^k_{ij} = \sum_{\substack{(i,j)\in \overline{A}_{sp}:\\ i\in A, j\in B}} y^k_{ij} \text{ pour les variables } y^k_{ij} \text{ existantes.}$$

$$y_{A,B} = \sum_{k=1}^{m+1} y^k_{A,B} \text{ pour les variables } y^k_{ij} \text{ existantes.}$$

$$y^T_{A,B} = \sum_{k\in T} y^k_{A,B} \text{ pour les variables } y^k_{ij} \text{ existantes.}$$

Nous avons bien sûr les versions dédoublées :

$$\bar{y}^k_{A,B} = \sum_{i\in A; j\in B} \bar{y}^k_{ij} = \sum_{\substack{(i,j)\in \overline{A}_{sp}:\\ i\in A, j\in B}} \bar{y}^k_{ij} \text{ pour les variables } y^k_{ij} \text{ existantes.}$$

$$\bar{y}_{A,B} = \sum_{k=1}^{m+1} \bar{y}^k_{A,B} \text{ pour les variables } y^k_{ij} \text{ existantes.}$$

$$\bar{y}^T_{A,B} = \sum_{k\in T} \bar{y}^k_{A,B} \text{ pour les variables } y^k_{ij} \text{ existantes.}$$

Pour tout sous-ensemble d'arêtes dirigées $A \subseteq \overline{A}_{sp}$,

$$y^k(A) = \sum_{(i,j)\in A} y_{ij}^k \text{ pour les variables } y_{ij}^k \text{ existantes.}$$

Pour tout sous-ensemble d'arêtes dirigées $A \subseteq \overline{A}_{sp}$,

$$y(A) = \sum_{k=1}^{m+1} y^k(A) = \sum_{k=1}^{m+1} \sum_{(i,j)\in A} y_{ij}^k \text{ pour les variables } y_{ij}^k \text{ existantes.}$$

Pour tout sous-ensemble d'arêtes dirigées $A \subseteq \overline{A}_{sp}$ et tout ensemble d'indices $T \subseteq \{1,\ldots,m\}$,

$$y^T(A) = \sum_{k\in T} y^k(A) = \sum_{k\in T} \sum_{(i,j)\in A} y_{ij}^k \text{ pour les variables } y_{ij}^k \text{ existantes.}$$

5.2.2 Les valeurs spécifiques

La cardinalité d'un ensemble V est le nombre d'éléments de cet ensemble et est dénotée par $\#(V)$ (ou plus brièvement par $\#V$) ou $|V|$.

Pour tout ensemble de sommet $S \subset \overline{V} \cup \{s,t\}$ dénotons par $\sigma(S) \in \mathbb{N}$ le nombre d'arêtes originales sous-tendues par les sommets de S c'est-à-dire les arêtes de $\overline{E}_r$ dont les deux sommets sont dans S.

Pour deux ensembles T_1 et T_2 d'indices de temps, définissons les valeurs

$$\text{diff\_max}(T_1,T_2) = \max\{|t_1 - t_2| : t_1 \in T_1, t_2 \in T_2\}$$

$$\text{diff\_min}(T_1,T_2) = \min\{|t_1 - t_2| : t_1 \in T_1, t_2 \in T_2\}$$

5.2.3 Les ensembles

Pour tout sous-ensemble de sommets $A \subset B$ (avec $B = \mathrm{V}$ ou $B = \overline{V} \cup \{s,t\}$), $\delta(A)$ représente l'ensemble des arêtes (de E ou $\overline{A}_{sp}$ respectivement), communément appelé les arêtes de la coupe $(A - B \setminus A)$ (*edge cutset*), qui ont un sommet dans A et l'autre dans $B \setminus A$.

Le contexte permettra de comprendre par rapport à quel ensemble on prend le complément.

Pour tout sous-ensemble de sommets $A \subset \overline{V} \cup \{s,t\}$, $\delta^+(A)$ représente l'ensemble des arcs de $\overline{A}_{sp}$ qui ont un leur premier sommet dans A et le deuxième dans $\overline{V} \cup \{s,t\} \setminus A$.

Pour tout sous-ensemble de sommets $A \subset \overline{V} \cup \{s,t\}$, $\delta^-(A)$ représente l'ensemble des arcs de $\overline{A}_{sp}$ qui ont un leur premier sommet dans $\overline{V} \cup \{s,t\} \setminus A$ et le deuxième dans A.

Pour tout sous-ensemble de sommets $A \subset \overline{V} \cup \{s,t\}$, $\gamma(A)$ représente l'ensemble des arcs de $\overline{A}_{sp}$ qui ont leurs deux sommets dans A.

Soit $T \subset \{1,\ldots,m+1\}$ un ensemble d'indices de temps. Comme nous avons besoin des mêmes indices mais avec un certain décalage, définissons l'ensemble d'indices de temps :

$$T^{+t} = \{k \in \{1,\ldots,m+1\} : k - t \in T\}.$$

De même, définissons

$$T^{-t} = \{k \in \{1,\ldots,m+1\} : k + t \in T\}.$$

5.3 Recherche d'une solution initiale

Le bénéfice d'une heuristique qui nous donne des solutions réalisables dès le début de la recherche est double. Dans ce chapitre, nous l'utilisons pour nous donner une borne supérieure sur la valeur de la fonction objectif. Dans le chapitre suivant où nous générons nos variables, elle nous permet de récolter aussi un lot de variables de départ pour la génération de colonnes.

Nous utilisons une énumération simple des solutions réalisables. L'heuristique *enumeration* construit petit à petit une solution réalisable suivant deux paramètres α et β : α est le nombre d'arêtes que l'heuristique teste à partir d'un chemin partiel donné alors que β est le nombre d'arêtes effectivement prises dans la solution réalisable. Évidemment $\beta \leqslant \alpha$. La figure 5.3 illustre cette heuristique avec deux jeux de paramètres : le cas (a) utilise $(\alpha,\beta) = (2,1)$ et le cas (b) utilise $(\alpha,\beta) = (4,2)$. Alors que dans le cas (b) l'heuristique teste $\alpha = 4$ arêtes à l'avance au lieu de seulement $\beta = 2$ dans le cas (a), elle obtient un meilleur résultat dans le cas (a) : $z_{(\mathrm{a})} = 48 \leqslant z_{(\mathrm{b})} = 56$. En fait, elle obtient la solution optimale.

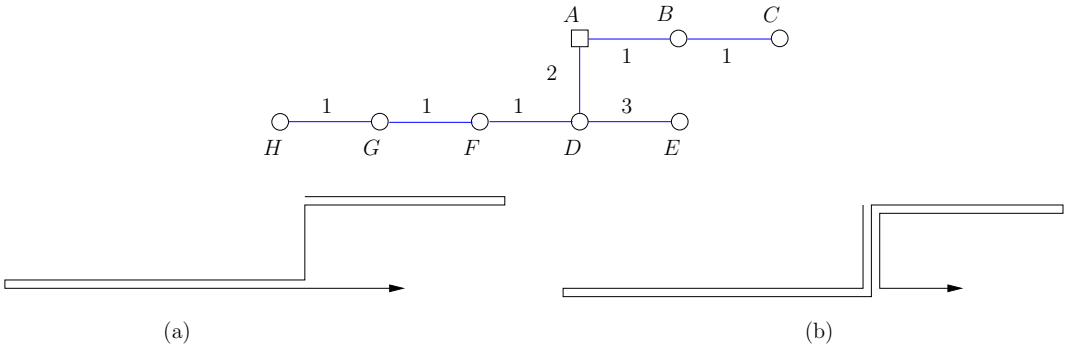


FIGURE 5.3 L'heuristique *enumeration* utilise les paramètres $(\alpha,\beta) = (2,1)$ dans le cas (a) et $(\alpha,\beta) = (4,2)$ dans le cas (b). La solution obtenue en (a) a pour coût 48 et est en fait une solution optimale. La solution obtenue en (b) a pour coût 56.

L'avantage de cette heuristique est sa rapidité de sorte que nous pouvons l'appeler plusieurs fois avec des paramètres différents pour obtenir un éventail de solutions réalisables qui nous seront précieuses dans les prochains chapitres.

5.4 Pré-traitements des variables

Nous discutons ici uniquement du pré-traitement des variables. Notons que seule une moitié des variables est véritablement nécessaire. En effet, si $y^l_{i,j} = 1$ et $y^{l+2}_{o,p} = 1$ alors nécessairement $y^{l+1}_{\bar{j},\bar{o}} = 1$. Nous utilisons cette propriété du modèle tout au long de la recherche d'une solution optimale. Par exemple, certaines règles de branchement ne considèrent que des variables pour les périodes de temps paires ou impaires. Nous n'en dirons pas plus concernant cette abondance de variables[1].

Nous présentons maintenant quelques règles relativement simples[2] à implémenter pour se débarrasser d'un certain nombre de variables. Si le pré-traitement permet des gains significatifs dans l'obtention d'une solution optimale dans un *Branch and Cut* ils doivent être utilisés

1. Il peut être tentant de croire qu'un modèle débarrassé de cette moitié de variables donnerait de meilleurs résultats mais nous n'avons pas trouvé comment faire cela intelligemment. Toutes nos tentatives dans cette voie ont été vaines. Ce que l'on gagne en réduisant le nombre de variables de moitié se paye avec un nombre considérable de contraintes supplémentaires.

2. Nous n'avons pas écrit de pré-traitement valable uniquement sur un certain type de graphes. Ces pré-traitements spécialisés permettraient d'éliminer des variables supplémentaires mais demanderaient plus de temps pour vérifier s'ils s'appliquent ou non sur un graphe donné.

à bon escient dans une approche de génération de colonnes. Nous verrons à la section 6.3.1 du chapitre 6 qu'éliminer un maximum de variables n'est pas la stratégie la plus efficace dans ce cas.

5.4.1 Le pré-traitement *mandatory edge*

Ce pré-traitement concerne l'existence de plus courts chemins $i \to j$ qui passent *tous* obligatoirement par l'arête originale $(\bar{j},j)$. Le lemme 2 interdit d'utiliser une arête avant de l'avoir desservie. Nous pouvons donc nous débarrasser de toutes les variables y^p_{ij} correspondantes pour tous les p. Sur la figure 5.4 nous voyons par exemple que nous pouvons nous débarrasser de toutes les variables y^p_{ax} car il n'est pas possible de desservir l'arc (x,w) sans passer préalablement par l'arête (w,x) à partir du sommet a. Par contre, nous ne pouvons exclure les variables y^p_{av} par exemple. Il est possible d'obtenir un plus court chemin $*a - *d - *v$ entre $*a$ et $*v$ dans le graphe original qui ne passe pas par l'arête $(*h, *v)$.

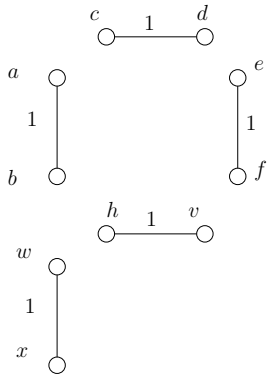


FIGURE 5.4 Nous pouvons exclure les variables y^p_{ax} mais pas les variables y^p_{av} avec le présolveur *mandatory edge*.

5.4.2 Le pré-traitement *shortest path minimum number of edges*

Si le plus petit[3] plus court chemin entre deux sommets $*i$ et $*j$ dans le graphe original comporte p arêtes, alors toutes les variables y^k_{ij} avec $k \leq p$ peuvent être fixées à 0. En effet, il n'est pas possible d'utiliser p arêtes alors que nous sommes dans une période $k \leq p$. Sur la figure 5.5, nous voyons que $y^2_{bd} = 0$ car il faut deux arêtes au minimum pour aller de b à d en passant par un plus court chemin.

3. En termes du nombre d'arêtes.

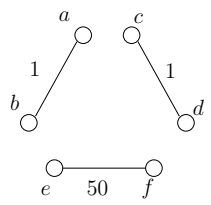


FIGURE 5.5 Le pésolveur *shortest path minimum number of edges* fixe y^2_{bd} à zéro car il est impossible à la période 2 d'utiliser un plus court chemin comportant 2 arêtes ou plus.

5.4.3 Le pré-traitement *reachable edge*

Définissons le *temps minimum de passage* $T(e)$ sur une arête originale $e \in \overline{E}_r$. Il s'agit du nombre minimal d'arêtes dans le graphe original pour rejoindre cette arête à partir du dépôt plus un. La figure 5.6 montre les temps de passage pour toutes les arêtes originales.

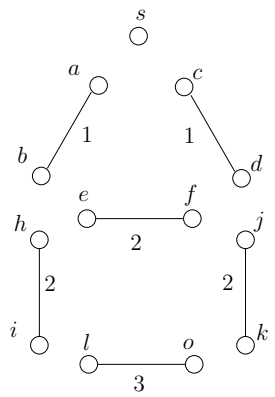


FIGURE 5.6 Les nombres affichés sont les *temps minimums de passage* sur les arêtes. Le pésolveur *reachable edge* fixe par exemple y^2_{he} et y^2_{dl} à zéro.

Soit une variable y^k_{ij}. Si

$$T((*\bar{\imath}, *i)) > k - 1 \text{ ou } T((*j, *\bar{\jmath})) > k$$

alors $y^k_{ij} = 0$. Dans le premier cas, nous ne pouvons pas partir à l'étape k de l'arête $(\bar{\imath},i)$ car on ne peut pas atteindre cette arête en $k-1$. Dans le deuxième cas, on ne peut pas desservir l'arête $(j,\bar{\jmath})$ à l'étape k car elle est inatteignable à cette période. Sur la figure 5.6, nous trouvons par exemple que $y^2_{he} = 0$ (premier cas) et que $y^2_{dl} = 0$ (deuxième cas).

5.4.4 Le pré-traitement *reachable shortest path*

Soit P_{vw} l'ensemble de tous les plus courts chemins entre les sommets v et w du graphe original. Définissons $T_{\max}(p)$ pour un plus court chemin $p \in P_{vw}$ comme le temps de passage maximum parmi les temps de passage des arêtes composant ce chemin p. Sur la figure 5.7 (a), nous avons 2 plus courts chemins entre les deux sommets $*i$ et $*j$: d'une part le chemin p_1 : $*i - v_1 - v_2 - *j$ et d'autre part le chemin p_2 : $*i - w_1 - w_2 - w_3 - w_4 - *j$ pour un coût de 5. Nous avons $T_{\max}(p_1) = 3$ et $T_{\max}(p_2) = 4$.

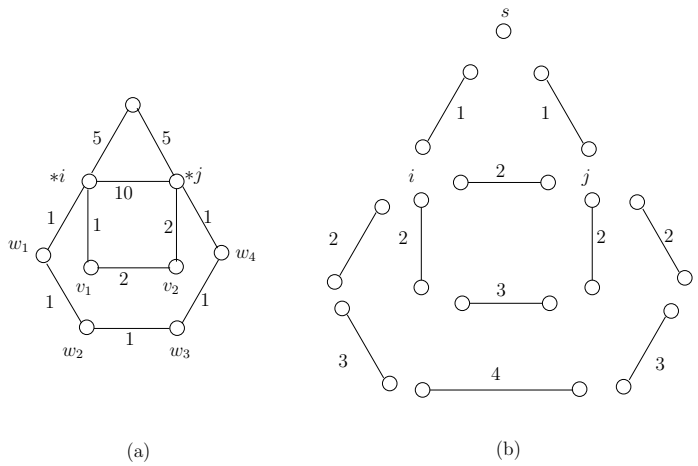


FIGURE 5.7 Les nombres affichés en (a) sont les coûts sur les arêtes et ceux affichés en (b) sont les temps minimum de passage sur les arêtes. Le pésolveur *reachable shortest path* fixe par exemple y_{ij}^2 et y_{ij}^3 à zéro.

Soit une variable y_{ij}^k et soit

$$t_{ij} = \min_{p \in P_{*i,*j}} \{T_{\max}(p)\}$$

alors $y_{ij}^k = 0 \ \forall \ 2 \leq k \leq t_{ij}$. Sur la figure 5.7 (b), on a $t_{ij} = \min\{3,4\} = 3$ et donc $y_{ij}^2 = y_{ij}^3 = 0$.

5.4.5 Le pré-traitement *depot*

Pour toute arête originale $(i,j) \in \overline{E}_r$ telle que $*i$ correspond au dépôt, nous avons $\forall\, v \in \overline{V}$: $(i,v) \in \overline{A}_{sp}$: $y_{iv}^2 = 0$. En effet, soit nous empruntons (i,j) à la période 1, soit nous empruntons une autre arête (a,b) à la période 1 et donc $\sum_{\substack{w \in \overline{V}: \\ (b,w) \in \overline{A}_{sp}}} y_{bw}^2 = 1$. Dans les deux cas, nous ne pouvons partir du sommet i à la période 2 comme illustré sur la figure 5.8.

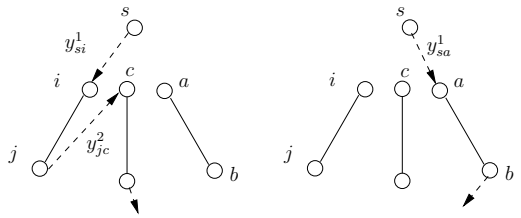


FIGURE 5.8 Le pésolveur *depot* fixe y^2_{iv} à zéro pour tout sommet $v \in \overline{V}$ car il est impossible de pouvoir partir de i à la période 2. Toute les variables y affichées sont à 1.

Pour la même raison invoquée et le fait que nous n'acceptons pas d'arête parallèle, nous pouvons aussi fixer y^3_{iv} à 0 $\forall\ v \in \overline{V}\ :(i,v) \in \overline{A}_{sp}$ et $*i$ est le dépôt.

5.4.6 Le pré-traitement *backward compatibility*

Soit une arête $e \in E$ du graphe original comme sur la figure 5.9. Regardons toutes les façons d'aboutir à cette arête à partir du sommet s en prenant des plus courts chemins pour passer d'une arête à une autre comme le requiert le lemme 3. Si nous voulons desservir l'arête e à la période $k = 2$, nous voyons sur la figure 5.9 (b) qu'il n'y a qu'une seule façon de faire : il faut desservir l'arête e_1 avant de desservir l'arête e. Nous sommes donc assurés de ne pas pouvoir desservir l'arête e_1 à partir de l'arête e à la période $k+1 = 3$. Donc $y^3_{e,e_1} = 0$.

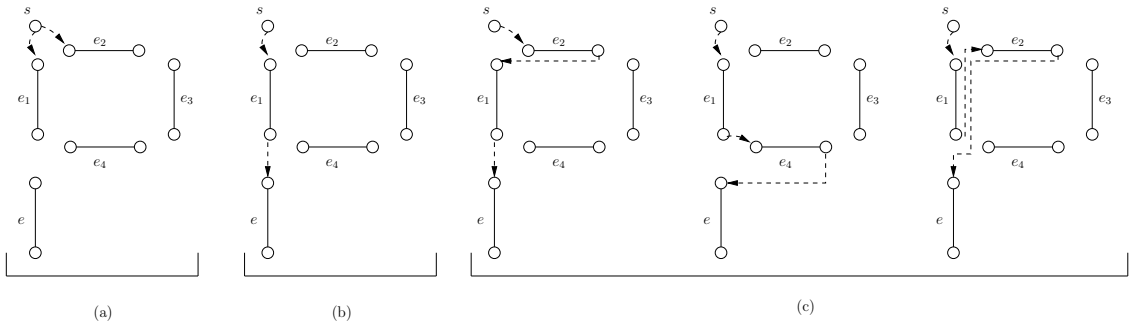


FIGURE 5.9 Le pésolveur *backward compatibility* permet de fixer pratiquement les variables y^3_{e,e_1} (voir (b)) et théoriquement les variables y^4_{e,e_1} (voir (c)) à 0.

Regardons sur la figure 5.9 (c) ce qui se passe si nous voulons desservir l'arête e à la période $k = 3$. Il y a seulement trois façons de faire. A chaque fois, il faut passer par l'arête e_1. Nous sommes donc assurés de ne pouvoir desservir l'arête e_1 à partir de l'arête e à la période $k+1 = 4$. Donc, $y^4_{e,e_1} = 0$. En procédant de la même manière pour la période $k = 4$,

nous fixerions y^5_{e,e_4} à 0.

Cette régle nous a paru trop compliquée[4] à implémenter pour ne fixer que très peu de variables. Il y a cependant un cas relativement simple qui a retenu notre attention. Celui où il n'existe qu'un seul plus court chemin ou plus exactement quand le plus court chemin composé d'un nombre minimum d'arêtes est unique. C'est le cas (b) de la figure 5.9 pour l'arête e.

Soit d le dépôt dans le graphe original et soit $v \in \mathrm{V} \setminus \{d\}$ un sommet tel qu'il existe un plus court chemin unique avec un minimum d'arêtes : $d = v_0 - v_1 - v_2 - v_3 - \ldots - v_{p-1} - v_p = v$ qui comprend p arêtes. Soit $s - (d,v_1) - (v_1,v_2) - (v_2,v_3) - \ldots - (v_{p-1},v)$ l'équivalent de ce plus court chemin unique dans le graphe auxiliaire. Alors nous avons

$$y^{p+1}_{\{v,v_{p-1}\},\{d,v_1,v_2,\ldots,v_{p-2}\}} = 0.$$

5.4.7 Le pré-traitement *forward compatibility*

Ce pré-traitement est le petit frère du pré-traitement *back compatibility*. Prenons la figure 5.10. Par un raisonnement analogue à celui utilisé précedemment pour le pré-traitement *back compatibility*, nous trouvons que les arêtes inatteignables à partir de l'arête e à la période $k = 3$ sont e_3 et e_5. Donc $y^3_{e,e_3} = y^3_{e,e_5} = 0$. De façon similaire, pour $k = 4$, nous fixons y^4_{e,e_6} à 0. Comme dans le cas *back compatibility* nous n'avons implémenté que le cas des plus courts chemins uniques composés d'un nombre minimum d'arêtes.

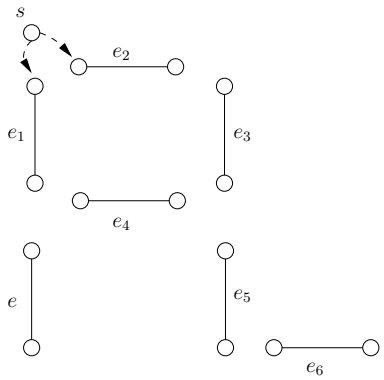


FIGURE 5.10 Le pésolveur *forward compatibility* permet de fixer pratiquement les variables $y^3_{\{e\},\{e_3,e_5\}}$ et théoriquement les variables y^4_{e,e_6} à 0.

4. Ce n'est pas démontré dans cette thèse mais nos tests nous ont montré que ce pré-traitement n'est pas très efficace.

Soit d le dépôt dans le graphe original et soit $v \in \mathrm{V}\backslash\{d\}$ un sommet tel qu'il existe un plus court chemin unique avec un minimum d'arêtes : $d = v_0 - v_1 - v_2 - v_3 - \ldots - v_{p-1} - v_p = v$ qui comprend p d'arêtes. Soit $\{e_1, e_2, \ldots, e_k\}$ l'ensemble des arêtes restantes du graphe original qui ne sont pas adjacentes au chemin, *i.e.* $e_i \cap v_j = \emptyset \ \forall \ i \forall \ j$. Alors nous avons

$$y^{p+1}_{\{v_{p-1},v_p\},\{e_1,\ldots,e_k\}} = 0.$$

5.5 Les inégalités de parité

Le travail présenté ici est issu d'une collaboration avec le professeur Ángel Corberán de l'Université de Valence et le professeur José María Sanchís de l'Université Polytechnique de Valence. Nous présentons 9 familles d'inégalités valides[5] : 6 familles dont 3 ont été généralisées.

Les inégalités de parité traitent de la parité de certains ensembles. Par exemple, toute tournée qui passe par un sommet donné de degré impair, devra nécessairement repasser au moins une fois de plus par ce sommet (inégalités *vertex*). Ceci n'est possible que parce que nous avons ajouté les sommets fictifs s et t. Sans ceux-ci, nous serions bien en peine de pouvoir exiger une quelconque parité car rappelons que le problème du postier chinois cumulatif n'exige pas de retour au dépôt et qu'une solution optimale pourrait très bien s'arrêter en n'importe quel sommet du graphe.

L'introduction d'un second sommet fictif nous permet de considérer l'arête (s,t) comme une arête originale. Cet artifice n'est pas indispensable mais permet de ne pas à avoir à considérer le sommet s comme un sommet particulier et facilite donc l'implémentation.

Petit à petit, nous nous sommes aperçus que la richesse du modèle L8 nous permettait d'écrire différemment les mêmes familles d'inégalités. Certaines de ces familles trouvées indépendamment se sont révélées être l'expression de la même idée mais vue sous différents angles[6]. Finalement, nous nous sommes rendus compte que toutes les familles d'inégalités de parité peuvent s'exprimer par seulement deux d'entre-elles.

5. Par manque de place, nous ne démontrons pas la validité des ces inégalités ni ne présentons les algorithmes de séparation excepté pour les inégalités *generalized co-circuit* à la section 5.6.2.

6. La lectrice est renvoyée à la section 5.5.9 page 118 pour un résumé de ces équivalences.

Nous commençons cette section par deux lemmes qui nous permettent de démontrer les équivalences entres les différentes familles d'inégalités valides.

5.5.1 Deux lemmes importants

La conservation de flot assurée par les égalités (4.41) et (4.43) du modèle nous permet, pour tout ensemble $S \subset \overline{V} \cup \{s,t\}$ de sommets, d'exprimer de manière équivalente ce qui se passe à l'intérieur de cet ensemble par ce qui se passe sur la frontière de cet ensemble et vice-versa. Ainsi, nous pouvons exprimer les inégalités de parité de deux manières différentes. Le lemme 20 suivant exprime mathématiquement cette relation entre l'intérieur et la frontière d'un ensemble de sommets. La figure 5.11 illustre le propos.

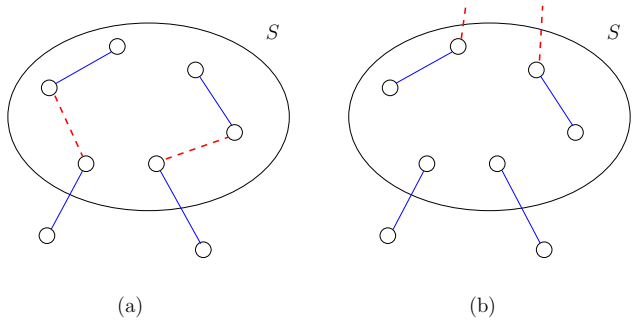


FIGURE 5.11 Ce qui se passe à l'intérieur *(a)* peut s'exprimer par ce qui se passe sur la frontière *(b)* et vice-versa.

Lemme 20 (Équivalence entre l'intérieur et la frontière)

$$\forall\, S \subset \overline{V} \cup \{s,t\} \quad : \quad y(\delta(S)) + 2\; y(\gamma(S)) = |S| \tag{5.1}$$

Démonstration

Soit un ensemble $S \subset \overline{V} \cup \{s,t\}$ de sommets. Pour chacun des sommets de $\overline{V}$, l'équation (4.41) du modèle impose un flot de 1. Pour les sommets fictifs s et t c'est l'équation (4.43) qui s'en charge. Donc, nous avons au total $|S|$ flots pour l'ensemble S.

□

Ce lemme nous permet d'écrire très simplement des relations équivalentes d'inégalités entre $y(\gamma(S))$ et $y(\delta(S))$:

$$y(\gamma(S)) \leq \alpha \Leftrightarrow y(\delta(S)) \geq |S| - 2 \cdot \alpha$$
$$y(\delta(S)) \geq \alpha \Leftrightarrow y(\gamma(S)) \leq \frac{|S| - \alpha}{2}.$$

Le lemme suivant exprime la conservation de flots pour tout coupe (edge-cutset) entre deux ensembles A,$B \subset \overline{V}$ de sommets. La figure 5.17 page 112 décrit la situation. Sur la figure, la coupe est d'ordre impair mais le lemme est valide aussi pour les coupes d'ordre pair. En reprenant les mêmes notations que celles de la figure 5.17, voici le lemme :

Lemme 21

$$y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{A,I} + y^k_{B,I}\right) = \sum_{k=2}^{m} \left(y^k_{J,A} + y^k_{J,B}\right) + y^{m+1}_{J,t} \tag{5.2}$$

Démonstration

Partons de l'observation que

$$y^k_{A,I} + y^k_{B,I} = y^{k+1}_{J,A} + y^{k+1}_{J,B} \quad \forall\ k \in \{2, \ldots, m-1\}$$

et que

$$y^1_{s,I} = y^2_{J,B} + y^2_{J,A} \text{ et } y^m_{A,I} + y^m_{B,I} = y^{m+1}_{J,t}.$$

Sommons le tout $\forall\ k \in \{1, \ldots, m+1\}$ pour obtenir (5.2).

□

Nous utiliserons aussi ce lemme dans la situation un peu plus simple de la figure 5.16 page 109. Nous avons alors :

Corollaire 22

$$y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{I,I} + y^k_{B,I}\right) = \sum_{k=2}^{m} \left(y^k_{J,B} + y^k_{J,I}\right) + y^{m+1}_{J,t} \tag{5.3}$$

Démonstration

Partons du lemme 5.2 et posons $A = I$. Ou bien partons de l'observation que

$$y^k_{I,I} + y^k_{B,I} = y^{k+1}_{J,B} + y^{k+1}_{J,I} \ \forall\ k \in \{2, \ldots, m-1\}$$

et que

$$y^1_{s,I} = y^2_{J,B} + y^2_{J,I} \text{ et } y^m_{I,I} + y^m_{B,I} = y^{m+1}_{J,t}.$$

Sommons le tout $\forall\ k \in \{1,\ldots,m+1\}$ pour obtenir (5.3).

$\square$

5.5.2 Les inégalités *vertex*

Soit un sommet $v \in \mathrm{V}$ et soit $I \subset \overline{V}$ l'ensemble des sommets correspondants dans $\overline{V}$, *i.e.* $I = \{i \in \overline{V} : *i = v\}$. Pour tout sommet $v \in \mathrm{V}$ de degré supérieur ou égal à deux, nous avons

$$y(\gamma(I)) = \sum_{k=2}^{m} \sum_{\substack{(i,j)\in\overline{A}_{sp}:\\ *i=v=*j}} y_{ij}^k \leq \lfloor \frac{\deg(v)}{2} \rfloor \tag{5.4}$$

La figure 5.12 illustre ces inégalités dans le cas d'un sommet de degré 3. (5.4) donne une limitation sur le nombre de plus courts chemins possibles entre des sommets de $\overline{V}$ qui représentent le même sommet $v \in \mathrm{V}$. Nous ne pouvons utiliser plus d'arêtes que le nombre d'arêtes dans un appariement maximal (maximal edge matching).

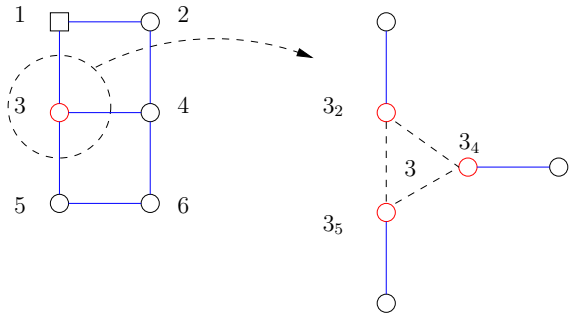


FIGURE 5.12 Les inégalités (5.4) sont illustrées sur le sommet 3. Dans le graphe auxiliaire éclaté, le sommet 3 est représenté par les trois sommets 3_2, 3_4 et 3_5 et seul un plus court chemin peut être utilisé entre ces trois sommets : $\sum_{k=2}^{m}(y_{3_2 3_4}^k + y_{3_4 3_2}^k + y_{3_2 3_5}^k + y_{3_5 3_2}^k + y_{3_4 3_5}^k + y_{3_5 3_4}^k) \leqslant 1$.

5.5.3 Les inégalités *clique*

Les inégalité *vertex* (5.4) se généralisent dans deux directions : tout d'abord, on étend la période aux cas $k = 1$ et $k = m + 1$ et donc aux sommets fictifs s et t, ensuite on considère n'importe quel ensemble de sommets de $\overline{V} \cup \{s,t\}$ et non plus seulement un ensemble de sommets de $\overline{V} \cup \{s,t\}$ représentants un même sommet $v \in V$. La seule restriction - que nous levons dans la généralisation des ces inégalités (voir les inégalités *generalized clique* (5.6)

page 108) - est que l'ensemble de sommets ne comporte pas d'arête de $\overline{E}_r$.

Pour toute clique $C \subset \overline{V} \cup \{s,t\}$ ne comprenant aucune arête de $\overline{E}_r$ avec $|C| \geq 2$, nous avons :

$$y(\gamma(C)) = \sum_{k=1}^{m+1} \sum_{\substack{(i,j)\in\overline{A}_{sp}: \\ i,j\in \text{ clique } C}} y_{ij}^k \leq \lfloor \frac{|C|}{2} \rfloor \tag{5.5}$$

La figure 5.13 illustre les inégalités *clique* dans le cas d'une clique à 5 sommets.

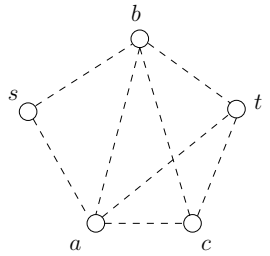


FIGURE 5.13 Les inégalités (5.5) sont illustrées sur 5 sommets dont les deux sommets fictifs s et t dans le graphe auxiliaire éclaté et seuls deux plus courts chemins peuvent être utilisés entre ces cinq sommets : $y(\gamma(\{a,b,c,s,t\})) = y^1_{s,\{a,b\}} + \sum_{k=2}^{m} y^k_{\{a,b,c\},\{a,b,c\}} + y^{m+1}_{\{a,b,c\},t} \leq \lfloor \frac{5}{2} \rfloor = 2$.

5.5.4 Les inégalités *generalized clique*

Considérons maintenant la possibilité d'avoir des arêtes de $\overline{E}_r$ dans un sous-ensemble C de sommets. Il est pratique de considérer que les sommets fictifs s et t forment une arête fictive (s,t). Définissons pour n'importe quel sous-ensemble $C \subset \overline{V} \cup \{s,t\}$:

$$\tilde{\sigma}(C) = \begin{cases} \sigma(C) + 1 & \text{si } s,t \in C \\ \sigma(C) & \text{sinon} \end{cases}$$

et $\tilde{E}_r = \overline{E}_r \cup \{(s,t)\}$.

Il est remarquable de constater que lorsqu'on considère un ensemble quelconque C de sommets de $\overline{V} \cup \{s,t\}$, il y ait peu de différence que l'on considère que les sommets soient reliés entre eux par des arêtes de $\tilde{E}_r$ ou pas. Le seul cas où la distinction est importante est

celui où tous les sommets de l'ensemble C sont reliés deux à deux par des arêtes de $\tilde{E}_r$.

Pour tout sous-ensemble $C \subset \overline{V} \cup \{s,t\}$ de sommets avec $|C| \geq 2$, nous avons

$$y(\gamma(C)) \leq \begin{cases} \lfloor \frac{|C|}{2} \rfloor & \text{si } \tilde{\sigma}(C) < \frac{|C|}{2} \\ \tilde{\sigma}(C) - 1 & \text{si } \tilde{\sigma}(C) = \frac{|C|}{2} \end{cases} \tag{5.6}$$

Il y a deux cas à distinguer. Le premier, lorsque $\tilde{\sigma}(C) < \frac{|C|}{2}$ (i.e. lorsque $|C|$ est impair ou bien lorsque $|C|$ est pair mais que tous les sommets ne sont pas reliés deux à deux par des arêtes de $\tilde{E}_r$), est similaire aux inégalités *clique* (5.5). La figure 5.14 montre toutes les possibilités à une permutation des arêtes près lorsque $|C| = 5$.

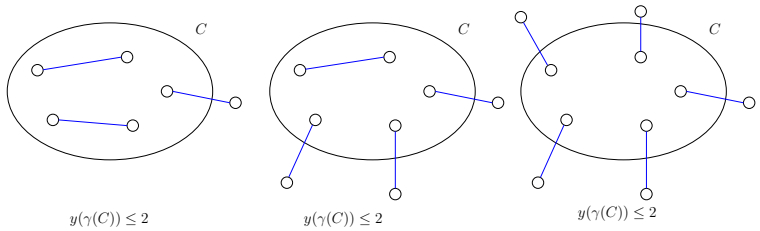


FIGURE 5.14 Les inégalités (5.6) sont illustrées sur 5 sommets. $y(\gamma(C)) \leq \lfloor \frac{5}{2} \rfloor = 2$.

Le deuxième cas, lorsque $\tilde{\sigma}(C) = \frac{|C|}{2}$ (i.e. lorsque $|C|$ est pair et que tous les sommets sont reliés deux à deux par des arêtes de $\tilde{E}_r$), se distingue des inégalités *clique* (5.5). C'est le cas le plus intéressant. La figure 5.15 montre toutes les possibilités à une permutation des arêtes près lorsque $|C| = 6$. Le cas qui nous intéresse est représenté sur l'ensemble le plus à gauche sur la figure : il y a exactement 3 arêtes de $\tilde{E}_r$ pour 6 sommets et $y(\gamma(C)) \leq 2$.

5.5.5 Les inégalités *odd-cut*

Ces inégalités indiquent que pour chaque sommet de degré impair dans le graphe original, il faut passer au moins une fois de plus que ce degré dans le graphe auxiliaire. Ceci n'est vrai que parce que nous avons introduit les sommets fictifs s et t ce qui impose de devoir passer un nombre pair de fois par chaque sommet non fictif.

La figure 5.16 illustre ces inégalités pour le sommet 3 de degré 3 de la figure 5.12 page 106. Soit $I \subset \overline{V}$ l'ensemble des sommets qui représentent le sommet $v \in$ V et $J \subset \overline{V}$ l'ensemble

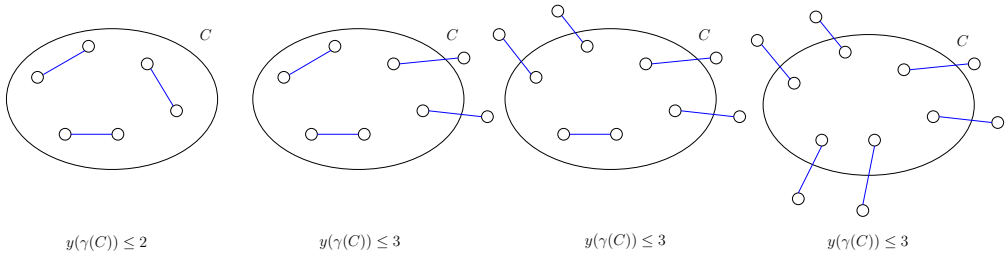


FIGURE 5.15 Les inégalités (5.6) sont illustrées sur 6 sommets. Le cas le plus intéressant (extrême gauche) est celui de l'ensemble avec 3 arêtes de $\tilde{E}_r$. Dans ce cas $y(\gamma(C)) \leq 3-1 = 2$ sinon $y(\gamma(C)) \leq \dfrac{6}{2} = 3$

des sommets qui sont adjacents aux sommets de I par une arête dans $\overline{E}_r$.

Pour tout sommet impair $v \in V$, lorsqu'on compte le nombre de passages entre I et J, on obtient :

$$2 \cdot y^1_{s,I} + y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{I,I} + y^k_{I,B\setminus J} + 2 \cdot y^k_{I,J} + y^k_{B,J} + 2 \cdot y^k_{B,I} \right) + y^{m+1}_{I,t} \geq \deg(v) + 1 \qquad (5.7)$$

(5.7) décrit le fait qu'une solution doit passer par un sommet de degré impair au moins une fois de plus que son degré. Dans l'inégalité (5.7), $I = I(v) = \{i \in \overline{V} : *i = v\}$, $J = J(v) = \{j \in \overline{V} : *(\bar{j}) = v\}$ et $B = \overline{V} \setminus I(v)$.

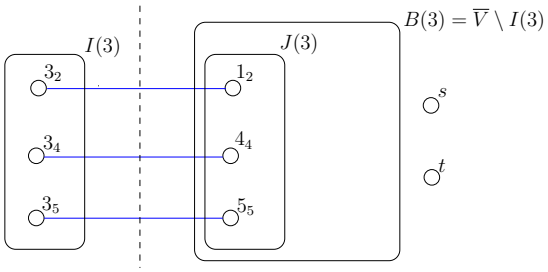


FIGURE 5.16 Les inégalités *odd-cut* (5.7) sont illustrées sur le sommet 3 du graphe gauche de la figure 5.12 page 106. Dans le graphe auxiliaire éclaté, le sommet 3 est représenté par les trois sommets 3_2, 3_4 et 3_5 et toute solution doit passer au moins 4 fois par ce sommet : $y^1_{s,J} + \sum_{k=2}^{7} \left(y^k_{I,I} + y^k_{I,B\setminus J} + 2 \cdot y^k_{I,J} + y^k_{B,J} + 2 \cdot y^k_{B,I} \right) + y^8_{I,t} \geq 4$. Il est à noter que dans ce cas précis, les termes $2 \cdot y^k_{s,I}$ sont absents.

En gardant la même notation, les inégalités (5.7) peuvent s'écrire de façon plus simple :

$$y(\delta(I)) = y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{I,B} + y^k_{B,I}\right) + y^{m+1}_{I,t} \geq 1 \tag{5.8}$$

En effet, puisqu'on sait qu'on va desservir les arêtes de $\overline{E}_r$, il faut juste imposer de repasser au moins une fois de plus par la coupe $I - B$.

Proposition 23 (Inégalités *odd-cut* : équivalence entre (5.7) et (5.8))
Pour tout sommet impair $v \in V$

$$2 \cdot y^1_{s,I} + y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{I,I} + y^k_{I,B\setminus J} + 2 \cdot y^k_{I,J} + y^k_{B,J} + 2 \cdot y^k_{B,I}\right) + y^{m+1}_{I,t} \geq \deg(v) + 1 \tag{5.7}$$

$$\Leftrightarrow$$

$$y(\delta(I)) = y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{I,B} + y^k_{B,I}\right) + y^{m+1}_{I,t} \geq 1 \tag{5.8}$$

Démonstration

Démontrons comment obtenir (5.8) à partir de (5.7). Le cheminement inverse permet d'obtenir (5.7) à partir de (5.8).

Soit une coupe $I - J$ comme sur la figure 5.16 page 109 où I est l'ensemble de tous les sommets $i \in \overline{V}$ qui représentent un sommet $v \in V$: $I = \{i \in \overline{V} : *i = v\}$.

Sommons pour tous les sommets $j \in J$ l'équation (4.41) du modèle. Nous obtenons alors :

$$y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{J,I} + y^k_{J,B} + y^k_{I,J} + y^k_{B,J}\right) + y^{m+1}_{J,t} = \deg v \tag{5.9}$$

Grâce au lemme 22, nous pouvons remplacer

$$\sum_{k=2}^{m} \left(y^k_{J,B} + y^k_{J,I}\right) + y^{m+1}_{J,t}$$

par

$$y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{I,I} + y^k_{B,I}\right)$$

dans (5.9), ce qui donne :

$$y^1_{s,I} + y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{I,I} + y^k_{B,I} + y^k_{I,J} + y^k_{B,J} \right) = \deg v \tag{5.10}$$

Prenons les inégalités *odd-cut* (5.7) :

$$2 \cdot y^1_{s,I} + y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{I,I} + y^k_{I,B\setminus J} + 2 \cdot y^k_{I,J} + y^k_{B,J} + 2 \cdot y^k_{B,I} \right) + y^{m+1}_{I,t} \geq \deg(v) + 1 \tag{5.7}$$

et soustrayons (5.10) de (5.7) membre à membre :

$$y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{I,B\setminus J} + y^k_{I,J} + y^k_{B,I} \right) + y^{m+1}_{I,t} \geq 1$$

ou encore

$$\lambda(I) = y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{I,B} + y^k_{B,I} \right) + y^{m+1}_{I,t} \geq 1 \tag{5.8}$$

□

En résumé, on peut écrire l'inégalité de deux façons différentes :

$$2 \cdot y^1_{s,I} + y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{I,I} + y^k_{I,B\setminus J} + 2 \cdot y^k_{I,J} + y^k_{B,J} + 2 \cdot y^k_{B,I} \right) + y^{m+1}_{I,t} \geq \deg(v) + 1 \tag{5.7}$$

$$y(\delta(I)) = y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{I,B} + y^k_{B,I} \right) + y^{m+1}_{I,t} \geq 1 \tag{5.8}$$

Dans la section 5.5.9 page 118 nous verrons que nous pouvons écrire ces inégalités de manière encore plus simple.

5.5.6 Les inégalités *generalized odd-cut*

Soit une coupe $A - B$ du graphe auxiliaire éclaté comme sur la figure 5.17. L'ensemble des sommets de $\overline{V}$ du graphe auxiliaire éclaté est partitionné en deux ensembles A et B. Soit $I \subset \overline{V}$ l'ensemble des sommets de $\overline{V}$ des arêtes de la coupe inclus dans l'ensemble A et soit $J \subset \overline{V}$ l'ensemble des sommets de $\overline{V}$ des arêtes de la coupe inclus dans l'ensemble B.

Alors pour toute coupe $A-B$ d'ordre impair qui possède la configuration[7] de la figure 5.17, il faut passer un nombre pair de fois et lorsque nous comptons ces passages, nous avons :

$$y^1_{s,A\setminus I}+2\cdot y^1_{s,I}+y^1_{s,J}+\sum_{k=2}^{m}\left(y^k_{A,I}+y^k_{A,B\setminus J}+2\cdot y^k_{A,J}+y^k_{B,J}+y^k_{B,A\setminus I}+2\cdot y^k_{B,I}\right)+y^{m+1}_{A,t}\geq |A-B|+1 \tag{5.11}$$

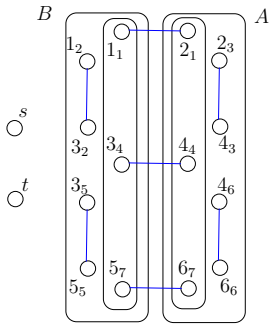


FIGURE 5.17 Les inégalités *generalized odd-cut* (5.11) sont illustrées sur le graphe gauche de la figure 5.12 page 106 dont nous représentons ici le graphe auxiliaire éclaté. Toute solution doit passer au moins 4 fois par les trois arêtes de la coupe $A-B$, ce qu'exprime l'inégalité 5.11 : $y^1_{s,A\setminus I}+y^1_{s,J}+\sum_{k=2}^{7}\left(y^k_{A,I}+y^k_{A,B\setminus J}+2\cdot y^k_{A,J}+y^k_{B,J}+y^k_{B,A\setminus I}+2\cdot y^k_{B,I}\right)+y^8_{A,t}\geq 4$. Il est à noter que dans ce cas-ci, il n'y a pas de terme $2\cdot y^1_{s,I}$.

Avec les mêmes notations et la même configuration, nous pouvons réécrire ces inégalités par

$$2\cdot\sum_{k=2}^{m}y^k_{I,I}+\sum_{k=2}^{m}\left(y^k_{I,A\setminus I}+y^k_{A\setminus I,I}\right)\leq |A-B|-1+y^1_{s,A\setminus I}+\sum_{k=2}^{m}\left(y^k_{B,A\setminus I}+y^k_{A\setminus I,B}\right)+y^{m+1}_{A\setminus I,t} \tag{5.12}$$

Remarquons que lorsque $A=I$, nous avons $2\cdot\sum_{k=2}^{m}y^k_{I,I}\leq |A-B|-1$, c'est-à-dire les inégalités *clique* (5.5).

Proposition 24 (Inégalités generalized odd-cut : équivalence entre (5.11) et (5.12))

$$y^1_{s,A\setminus I}+2\cdot y^1_{s,I}+y^1_{s,J}+\sum_{k=2}^{m}\left(y^k_{A,I}+y^k_{A,B\setminus J}+2\cdot y^k_{A,J}+y^k_{B,J}+y^k_{B,A\setminus I}+2\cdot y^k_{B,I}\right)+y^{m+1}_{A,t}\geq |A-B|+1 \tag{5.11}$$

7. Il faut s'assurer que les sommets fictifs s et t se trouvent bien du même côté.

$$\Leftrightarrow$$

$$2\cdot\sum_{k=2}^{m} y^k_{I,I}+\sum_{k=2}^{m}\left(y^k_{I,A\setminus I}+y^k_{A\setminus I,I}\right)\leq |A-B|-1+y^1_{s,A\setminus I}+\sum_{k=2}^{m}\left(y^k_{B,A\setminus I}+y^k_{A\setminus I,B}\right)+y^{m+1}_{A\setminus I,t} \quad (5.12)$$

Démonstration

Démontrons comment obtenir (5.12) à partir de (5.11). Le cheminement inverse permet d'obtenir (5.11) à partir de (5.12).

Soit une coupe $I-J$ comme sur la figure 5.17 page 112.

Sommons pour tous les sommets $i\in I$ et $j\in J$ l'équation (4.41) du modèle. Nous obtenons alors :

$$\begin{aligned} y^1_{s,I}+y^1_{s,J}+\sum_{k=2}^{m}\left(y^k_{I,A}+y^k_{I,B}\right)+ \\ \sum_{k=2}^{m}\left(y^k_{A,I}+y^k_{B,I}\right)+ \\ \sum_{k=2}^{m}\left(y^k_{J,A}+y^k_{J,B}\right)+ \\ \sum_{k=2}^{m}\left(y^k_{A,J}+y^k_{B,J}\right)+ \\ y^{m+1}_{I,t}+y^{m+1}_{J,t}=2\cdot|I| \end{aligned} \quad (5.13)$$

Prenons les inégalités *generalized odd-cut* (5.11) :

$$y^1_{s,A\setminus I}+2\cdot y^1_{s,I}+y^1_{s,J}+\sum_{k=2}^{m}\left(y^k_{A,I}+y^k_{A,B\setminus J}+2\cdot y^k_{A,J}+y^k_{B,J}+y^k_{B,A\setminus I}+2\cdot y^k_{B,I}\right)+y^{m+1}_{A,t}\geq |A-B|+1 \quad (5.11)$$

et soustrayons (5.13) de (5.11) membre à membre. Après simplifications, nous obtenons :

$$y^1_{s,A\setminus I}+y^1_{s,I}+\sum_{k=2}^{m}\left(y^k_{A\setminus I,B}+y^k_{B,A\setminus I}-y^k_{I,A}+y^k_{B,I}-y^k_{J,A}-y^k_{J,B}\right)-y^{m+1}_{J,t}+y^{m+1}_{A\setminus I,t}\geq 1-|A-B|.$$

Grâce au lemme 5.2, nous pouvons remplacer

$$y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{B,I} - y^k_{J,A} - y^k_{J,B}\right) - y^{m+1}_{J,t}$$

par

$$-\sum_{k=2}^{m} y^k_{A,I}.$$

Nous obtenons donc

$$y^1_{s,A\setminus I} + \sum_{k=2}^{m} \left(y^k_{A\setminus I,B} + y^k_{B,A\setminus I} - y^k_{I,A} - y^k_{A,I}\right) + y^{m+1}_{A\setminus I,t} \geq 1 - |A-B|$$

ou

$$|A-B| - 1 + y^1_{s,A\setminus I} + \sum_{k=2}^{m} \left(y^k_{A\setminus I,B} + y^k_{B,A\setminus I}\right) + y^{m+1}_{A\setminus I,t} \geq \sum_{k=2}^{m} \left(y^k_{I,A} + y^k_{A,I}\right)$$

c'est-à-dire (5.12).

□

Il existe cependant une écriture encore plus simple de ces inégalités :

$$y(\delta(A)) = y^1_{s,A} + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A}\right) + y^{m+1}_{A,t} \geq 1 \tag{5.14}$$

Remarquons que si $A = I$, nous obtenons les inégalités *odd-cut* (5.8).

Démontrons cette équivalence entre (5.11) et (5.14) :

Proposition 25 (Inégalités generalized odd-cut : équivalence entre (5.11) et (5.14))

$$y^1_{s,A\setminus I} + 2\cdot y^1_{s,I} + y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{A,I} + y^k_{A,B\setminus J} + 2\cdot y^k_{A,J} + y^k_{B,J} + y^k_{B,A\setminus I} + 2\cdot y^k_{B,I}\right) + y^{m+1}_{A,t} \geq |A-B| + 1 \tag{5.11}$$

$$\Leftrightarrow$$

$$y(\delta(A)) = y^1_{s,A} + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A}\right) + y^{m+1}_{A,t} \geq 1 \tag{5.14}$$

Démonstration

Démontrons comment obtenir (5.14) à partir de (5.11). Le cheminement inverse permet d'obtenir (5.11) à partir de (5.14).

Soit une coupe $I - J$ comme sur la figure 5.17 page 112.

Sommons pour tous les sommets $j \in J$ l'équation (4.41) du modèle. Nous obtenons alors :

$$y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{J,A} + y^k_{J,B} + y^k_{B,J} + y^k_{A,J}\right) + y^{m+1}_{J,t} = |A - B| \tag{5.15}$$

Prenons l'inégalité *generalized odd-cut* (5.11) :

$$y^1_{s,A\setminus I} + 2 \cdot y^1_{s,I} + y^1_{s,J} + \sum_{k=2}^{m} \left(y^k_{A,I} + y^k_{A,B\setminus J} + 2 \cdot y^k_{A,J} + y^k_{B,J} + y^k_{B,A\setminus I} + 2 \cdot y^k_{B,I}\right) + y^{m+1}_{A,t} \geq |A - B| + 1 \tag{5.11}$$

et soustrayons (5.15) de (5.11) membre à membre. Après simplifications, nous obtenons :

$$\begin{aligned} y^1_{s,A\setminus I} + 2 \cdot y^1_{s,I} + \sum_{k=2}^{m} & \left(y^k_{A,I} + y^k_{A,B\setminus J} + 2 \cdot y^k_{A,J} + y^k_{B,J} + y^k_{B,A\setminus I} + 2 \cdot y^k_{B,I}\right) \\ & - \sum_{k=2}^{m} \left(y^k_{I,A} + y^k_{J,B} + y^k_{B,J} + y^k_{A,J}\right) + y^{m+1}_{A,t} - y^{m+1}_{J,t} \geq 1. \end{aligned}$$

Grâce au lemme 5.2, nous pouvons remplacer

$$y^1_{s,I} + \sum_{k=2}^{m} \left(y^k_{A,I} + y^k_{B,I}\right) - y^{m+1}_{J,t}$$

par

$$\sum_{k=2}^{m} \left(y^k_{J,A} + y^k_{J,B}\right).$$

Nous obtenons donc

$$y^1_{s,A} + \sum_{k=2}^{m} \left(y^k_{A,B\setminus J} + y^k_{A,J} + y^k_{B,A\setminus I} + y^k_{B,I} \right) + y^{m+1}_{A,t} \geq 1$$

ou

$$y^1_{s,A} + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A} \right) + y^{m+1}_{A,t} \geq 1$$

c'est-à-dire (5.14).

□

Ces inégalités peuvent aussi se formuler ainsi :

$$\begin{aligned} y^1_{s,A} + y^{m+1}_{A,t} &\leq 1 + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A} \right) \\ y^1_{s,B} + y^{m+1}_{B,t} &\leq 1 + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A} \right) \end{aligned} \tag{5.16}$$

Proposition 26 (Inégalités generalized odd-cut : équivalence entre (5.16) et (5.14))

$$\begin{aligned} y^1_{s,A} + y^{m+1}_{A,t} &\leq 1 + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A} \right) \\ y^1_{s,B} + y^{m+1}_{B,t} &\leq 1 + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A} \right) \end{aligned} \tag{5.16}$$

$$\Leftrightarrow$$

$$y^1_{s,A} + \sum_{k=2}^{m} \left(y^k_{A,B} + y^k_{B,A} \right) + y^{m+1}_{A,t} \geq 1 \tag{5.14}$$

Démonstration

Remplaçons $y^1_{s,A}$ par $1 - y^1_{s,B}$ et $y^{m+1}_{A,t}$ par $1 - y^{m+1}_{B,t}$ dans 5.14, nous obtenons alors la deuxième des inégalités 5.16. Une démarche analogue permet de trouver la première des inégalités 5.16 en constatant que les inégalités 5.14 sont aussi valides lorsqu'on remplace l'ensemble A par l'ensemble B.

□

En résumé, on peut écrire les inégalités *generalized odd-cut* de plusieurs façons différentes :

$$y^1_{s,A\setminus I}+2\cdot y^1_{s,I}+y^1_{s,J}+\sum_{k=2}^{m}\left(y^k_{A,I}+y^k_{A,B\setminus J}+2\cdot y^k_{A,J}+y^k_{B,J}+y^k_{B,A\setminus I}+2\cdot y^k_{B,I}\right)+y^{m+1}_{A,t}\geq |A-B|+1 \tag{5.11}$$

$$2\cdot\sum_{k=2}^{m}y^k_{I,I}+\sum_{k=2}^{m}\left(y^k_{I,A\setminus I}+y^k_{A\setminus I,I}\right)\leq |A-B|-1+y^1_{s,A\setminus I}+\sum_{k=2}^{m}\left(y^k_{B,A\setminus I}+y^k_{A\setminus I,B}\right)+y^{m+1}_{A\setminus I,t} \tag{5.12}$$

$$y(\delta(A))=y^1_{s,A}+\sum_{k=2}^{m}\left(y^k_{A,B}+y^k_{B,A}\right)+y^{m+1}_{A,t}\geq 1 \tag{5.14}$$

$$\begin{aligned} y^1_{s,A}+y^{m+1}_{A,t}&\leq 1+\sum_{k=2}^{m}\left(y^k_{A,B}+y^k_{B,A}\right)\\ y^1_{s,B}+y^{m+1}_{B,t}&\leq 1+\sum_{k=2}^{m}\left(y^k_{A,B}+y^k_{B,A}\right)\end{aligned} \tag{5.16}$$

5.5.7 Les inégalités *odd-set*

Les inégalités *generalized odd-cut* peuvent se généraliser de la façon suivante. Pour tout ensemble impair $I\subset\overline{V}\cup\{s,t\}$ de sommets, il faut un plus court chemin vers les autres sommets :

$$y(\delta(I))=y^1_{s,I}+\sum_{k=2}^{m}\left(y^k_{I,V\setminus I}+y^k_{V\setminus I,I}\right)+y^{m+1}_{I,t}\geq 1 \tag{5.17}$$

(5.17) impose que tout ensemble impair de sommets soit relié à l'extérieur. La figure 5.18 illustre ces inégalités.

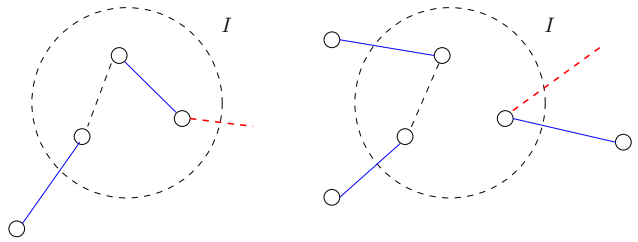


FIGURE 5.18 *Les inégalités* odd-set (5.17) *sont illustrées : il faut un lien avec l'extérieur.*

Remarquons que lorsque l'ensemble $I=I_v=\{i\in\overline{V}:*i=v\}$ comprenant les

sommets de $\overline{V}$ représentant un seul sommet $v \in$ V du graphe original, nous retrouvons les inégalités *odd-cut* (5.17). Lorsque nous sommes dans la configuration de la figure 5.17 page 112, les inégalités *odd-set* 5.17 ne sont rien d'autre que les inégalités *generalized odd-cut* (5.14).

5.5.8 Les inégalités *even-set*

En considérant les inégalités *odd-set* (5.17), on peut se demander s'il n'existe pas des inégalités similaires pour les ensembles pairs de sommets. Prenons un ensemble pair de sommets $S \subsetneqq \overline{V}$ d'arêtes originales. Alors nous avons

$$y(\delta(S)) \geq 2 \tag{5.18}$$

(5.18) impose que tout ensemble pair de sommets de S représentant un ensemble d'arêtes originales soit relié à l'extérieur. La figure 5.19 illustre ces inégalités.

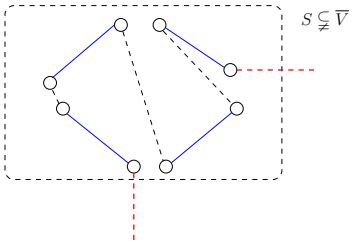


FIGURE 5.19 Les inégalités (5.18) sont illustrées : il faut un lien avec l'extérieur.

5.5.9 Équivalences entre inégalités de parité

Le lemme 5.1 page 104 permet de démontrer relativement simplement l'équivalence entre plusieurs inégalités de parité dans certaines configuration. Nous détaillons seulement la démonstration de la proposition 27. Des démonstrations analogues et simples permettent de démontrer les propositions 28-30.

Les inégalités *vertex* (5.4) et les inégalités *odd-cut* (5.7) sont en fait strictement équivalentes comme le démontre la proposition suivante.

Proposition 27 (Équivalence entre inégalités *vertex* (5.4) et *odd-cut* (5.7))
Pour tout sommet impair $v \in \mathrm{V}$, dénotons par $I_v = \{i \in \overline{V} : *i = v\}$. Nous avons alors

$$y(\gamma(I_v)) = \sum_{k=2}^{m} \sum_{\substack{(i,j)\in\overline{A}_{sp}:\\ *i=v=*j}} y_{ij}^k \leq \lfloor \frac{\deg(v)}{2} \rfloor \tag{5.4}$$

$$\Leftrightarrow$$

$$y(\delta(I_v)) = y_{s,I}^1 + \sum_{k=2}^{m} \left(y_{I,B}^k + y_{B,I}^k\right) + y_{I,t}^{m+1} \geq 1 \tag{5.8}$$

Démonstration

Il s'agit d'une conséquence directe du lemme 5.1 page 104 et du fait que l'on traite de sommets $v \in \mathrm{V}$ de degré impair. Plus précisément :

$$y(\gamma(I_v)) \leq \lfloor \frac{\deg(v)}{2} \rfloor = \frac{\deg(v)-1}{2} \Leftrightarrow y(\delta(I_v)) \geq |I_v| - 2(\frac{\deg(v)-1}{2}) = 1$$

□

Les inégalités *generalized clique* (5.6) sont bien plus générales que les inégalités *generalized odd-cut* (5.11) mais si on se trouve dans la configuration de la figure 5.17 page 112, alors elles sont équivalentes.

Proposition 28 (Équivalence entre les inégalités *gen. clique* (5.6) et *gen. odd-cut* (5.11))
Soit la configuration de la figure 5.17 page 112. A,$B \subset \overline{V}$, $|A|$,$|B|$ impairs, $A - B$ coupe impaire de G(V,E). Alors

$$y(\gamma(S)) \leq \lfloor \frac{|A|}{2} \rfloor \tag{5.6}$$

$$\Leftrightarrow$$

$$y(\delta(A)) \geq 1 \tag{5.11}$$

Nous avons le même type d'équivalence entre les inégalités *generalized clique* et les inégalités *odd-set*.

Proposition 29 (Équivalence entre les inégalités *generalized clique* (5.6) et *odd-set* (5.17))
Pour tout ensemble impair $I \subset \overline{V} \cup \{s,t\}$ de sommets, nous avons

$$y(\gamma(I)) \leq \frac{|I|-1}{2} \tag{5.5}$$

$$\Leftrightarrow$$

$$y(\delta(I)) \geq 1 \tag{5.17}$$

Et finalement, nous avons aussi l'équivalence entre les inégalités *generalized clique* et *even-set*.

Proposition 30 (Équivalence entre les inégalités *generalized clique* (5.6) et *even-set* (5.18)) *Pour tout ensemble pair $I \subset \overline{V}$ de sommets ne comprenant que des arêtes originales, nous avons*

$$y(\gamma(I)) \leq \sigma(I) - 1 \tag{5.6}$$

$$\Leftrightarrow$$

$$y(\delta(I)) \geq 2 \tag{5.17}$$

Résumons les équivalences dans le tableau 5.4.

Tableau 5.4 Équivalences entre les inégalités de parité.

IMPAIR	Intérieur		Extérieur
$v \in \mathrm{V} : v$ impair $I = \{i \in \overline{V} : *i = v\}$	inégalités *vertex* $y(\gamma(I)) \leq \lfloor \frac{\deg(v)}{2} \rfloor$	$\Leftrightarrow$	inégalités *odd-cut* $y(\delta(I)) \geq 1$
$A,B \subset \overline{V}$ $\lvert A\rvert,\lvert B\rvert$ impair $A - B$ coupe impair	inégalités *generalized clique* $y(\gamma(I)) \leq \lfloor \frac{\lvert A\rvert}{2} \rfloor$	$\Leftrightarrow$	inégalités *generalized odd-cut* $y(\delta(A)) \geq 1$
$I \subset \overline{V} \cup \{s,t\}$ $\lvert I\rvert$ impair	inégalités *generalized clique* $y(\gamma(I)) \leq \lfloor \frac{\lvert I\rvert}{2} \rfloor$	$\Leftrightarrow$	inégalités *odd-set* $y(\delta(I)) \geq 1$
PAIR	Intérieur		Extérieur
$I \subset \overline{V}$ $\lvert I\rvert$ pair et $\sigma(I) = \frac{\lvert I\rvert}{2}$	inégalités *generalized clique* $y(\gamma(I)) \leq \sigma(I) - 1$	$\Leftrightarrow$	inégalités *even-set* $y(\delta(I)) \geq 2$

Étudions maintenant l'inclusion des différentes familles d'inégalités de parité. Nous dirons qu'une famille X d'inégalités de parité est inclue dans une famille Y d'inégalités de parité si

toute inégalité de X peut s'écrire comme une inégalité de Y et nous noterons $X \subset Y$. Si de plus, certaines inégalités de Y ne peuvent pas s'obtenir comme inégalités de X, l'inclusion est stricte et est notée par $X \subsetneqq Y$.

La propriété 31 suivante nous donne les relations d'inclusion pour les inégalités *odd-cut*, *generalized odd-cut* et *odd-set*.

Proposition 31

$$\text{odd-cut inequalities} \subsetneqq \text{generalized odd-cut inequalities} \subsetneqq \text{odd-set inequalities} \tag{5.19}$$

Démonstration

odd-cut inequalities $\subset$ *generalized odd-cut inequalities* :
Soit $I_v = \{i \in \overline{V} : *i = v\}$ pour un sommet $v \in$ V de degré impair. Il suffit de prendre $A = I_v$ dans (5.14) pour obtenir (5.8).
generalized odd-cut inequalities $\subset$ *odd-set inequalities* :
Soient A et B comme sur la figure 5.17 page 112. Il suffit de prendre $I = A$ dans (5.17) pour obtenir (5.14).
La stricte inégalité s'obtient en prenant des ensembles qui ne sont pas considérés dans une famille mais bien dans l'autre.

□

La figure 5.20 permet de mieux saisir l'inclusion des diverses familles d'inégalités de parité.

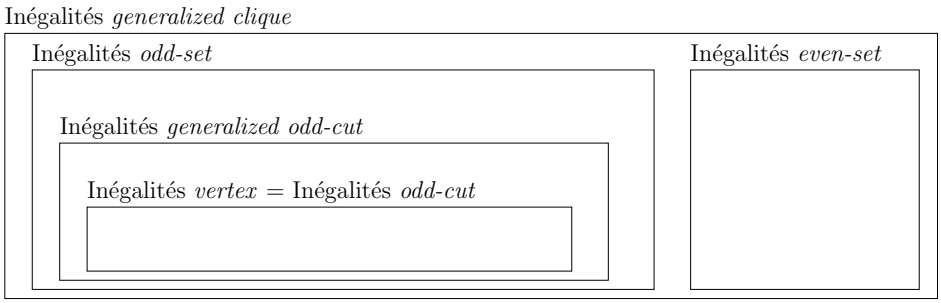


FIGURE 5.20 Les inclusions des familles d'inégalités de parité.

Cette section clôture les inégalités de parité. On peut se demander s'il n'y a pas moyen de faire mieux et la réponse est oui comme nous le verrons dans la prochaine section.

5.6 Les inégalités *co-circuit*

Les inégalités *co-circuit* ont été introduites dans le paradigme du problème du postier chinois par G. Ghiani et G. Laporte dans leur article *A branch-and-cut algorithm for the Undirected Rural Postman Problem* (voir Ghiani et Laporte, 2000) mais ces inégalités étaient connues avant. Ghiani et Laporte citent d'ailleurs un article de Barahona et Grötschel (voir Barahona et Grötschel, 1986) qui introduit le nom de ces inégalités et citent d'autres références concernant ces inégalités (voir Ghiani et Laporte, 2000, page 473). Nous pouvons utiliser ces inégalités avec le modèle L8. Cette adaptation est due aux professeurs Ángel Corberán et José María Sanchís.

5.6.1 Les inégalités *co-circuit*

Soit un ensemble de sommets $S \subsetneq \overline{V} \cup \{s,t\}$ pair. Comme l'ensemble est pair, il faut que $y(\delta(S))$ soit pair aussi. Nous savons donc que si un flot de 1 entre ou sort de cet ensemble S, il en faut automatiquement un autre qui sort ou entre pour garder la parité. Comme toutes les variables sont binaires, nous avons les inégalités valides suivantes :

$$y\left(\delta(S) \setminus \{e\}\right) \geqslant y_e \qquad \forall\ e \in \delta(S),\ S \text{ pair.} \tag{5.20}$$

La figure 5.21 illustre le cas où $y_e = 1$.

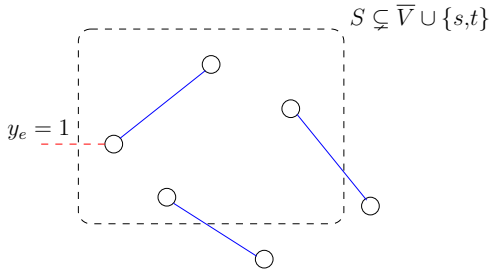


FIGURE 5.21 Les inégalités *co-circuit* (5.20) sont illustrées : comme S est pair et qu'une arête e entre ou sort de S, il en faut une autre qui sort ou entre de S.

Nous pouvons généraliser ces inégalités de la façon suivante. Soit un ensemble impair d'arêtes $F \subset \delta(S)$, nous avons alors les inégalités valides suivantes :

$$y\left(\delta(S) \setminus F\right) \geqslant y(F) - |F| + 1 \qquad \forall\ F \subset \delta(S) \text{ et } F \text{ impair, } S \text{ pair.} \tag{5.21}$$

Illustration des inégalités

La figure 5.22 permet de mieux comprendre les inégalités (5.21). Cette figure représente une solution relaxée dont nous avons agrégé les flots entre deux sommets pour toutes les périodes de temps : le flot agrégé entre deux sommets i et j est $\bar{y}_{ij} = \sum_{k=1}^{m+1}(y_{ij}^k + y_{ji}^k)$. Nous gardons les arêtes originales pour maintenir la connexité du graphe de support (*support graph*). Les arêtes originales ont toutes un coût nul.

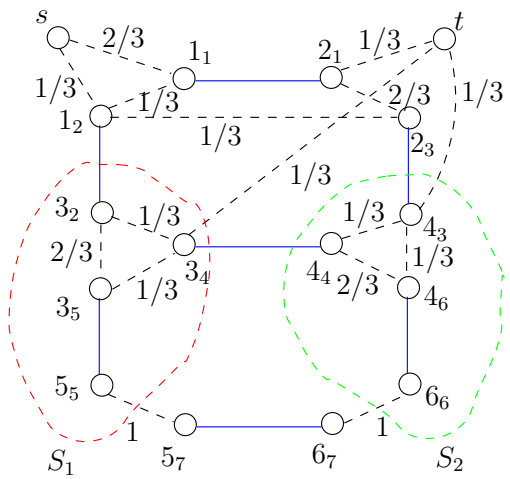


FIGURE 5.22 Les inégalités *co-circuit* (5.21) sont illustrées. S_1 etS_2 représentent des inégalités violées.

Si nous prenons $S_1 = \{3_2,3_4,3_5,5_5\}$ et $F_1 = \{(5_5,5_7)\}$ nous avons bien une inégalité *co-circuit* violée :

$$y\left(\delta(S_1) \setminus F\right) = 1/3 \geqslant y(F_1) - |F_1| + 1 = 1.$$

Un autre exemple est obtenu en prenant $S_2 = \{4_3,4_4,4_6,6_6\}$ et $F_2 = \{(6_6,6_7)\}$.

5.6.2 Les inégalités *generalized co-circuit*

Nous pouvons généraliser les inégalités *co-circuit* lorsque l'ensemble de sommets $S \subsetneq \overline{V} \cup \{s,t\}$ est impair : il suffit de considérer un nombre pair d'arêtes comme sur la figure 5.23. Prenons $F = \{e_1,e_2\}$. Si $y_{e_1} = y_{e_2} = 1$ alors $y\left(\delta(S) \setminus F\right) \geqslant 1$.

Soient $S \subset \overline{V} \cup \{s,t\}$ et $F \subset \delta(S)$ tels que $|S| + |F|$ est impair. Nous avons alors :

$$y\left(\delta(S) \setminus F\right) \geqslant y(F) - |F| + 1 \qquad \forall\ F \subset \delta(S) \text{ et } |F| + |S| \text{ impair.} \tag{5.22}$$

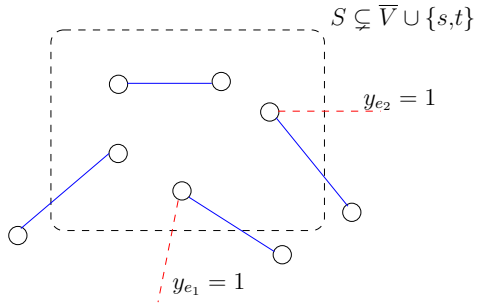


FIGURE 5.23 Les inégalités *generalized co-circuit* (5.22) sont illustrées : le nouveau cas où S est impair et F pair.

Reconsidérons les deux cas :

Cas 1. S est pair et F est impair et nous avons les inégalités *co-circuit* (5.21).

Cas 2. S est impair et F est pair. Si $F = \emptyset$ alors nous obtenons $y(\delta(S)) \geqslant 1$ c'est-à-dire les inégalités *generalized odd-cut* (5.14).

Nous pouvons donc considérer les inégalités *generalized co-circuit* comme une généralisation des inégalités de parités sauf dans le cas où l'ensemble S est composé uniquement d'arêtes originales au sens de la section 5.5.8 sur les inégalités *even-set*. Dans ce cas, nous avons $y(\delta(S)) \geq 2$. qui n'est pas repris par les inégalités *generalized co-circuit*.

Nous présentons maintenant deux procédures pour la séparation. Tout d'abord une procédure heuristique. Si celle-ci ne trouve aucune nouvelle coupe, alors nous utilisons la procédure exacte présentée après.

Procédure heuristique pour la séparation

Soit $\{\bar{y}_e\}$ une solution relaxée agrégée pour les périodes de temps : $\bar{y}_e = \sum_{k=1}^{m+1}(y_{ij}^k + y_{ij}^k)$ avec $e = (ij)$. Nous construisons alors un graphe induit par les arêtes e telles que $\bar{y}_e > 0$ et nous attribuons un flot $h_e = \min\{\bar{y}_e, 1 - \bar{y}_e\}$ sur ces arêtes.

Nous détectons les composantes connexes $C_1, \ldots, C_p$ et travaillons sur chacune de ces composantes suivant la parité de celles-ci :

Si C_i est impair Nous trouvons $F \subseteq \delta(C_i)$ avec F pair comme suit :

Soit $e \in \delta(C_i)$. Si $\bar{y}_e \geqslant 0{,}5$ nous prenons e dans F sinon $e \in \delta(C_i) \setminus F$.
Si F est pair ou nul nous avons terminé. Sinon, nous ajoutons ou enlevons une arête de F comme suit :
Soit $f \in F$ une arête de flot minimal et $g \in \delta(C_i) \setminus F$ une arête de flot maximal :

$$\bar{y}_f = \min\{\bar{y}_e : e \in F\} \text{ et } \bar{y}_g = \max\{\bar{y}_e : e \in \delta(C_i) \setminus F\}$$

Si $|0{,}5 - \bar{y}_f| < |0{,}5 - \bar{y}_g|$ nous enlevons l'arête f de F sinon nous ajoutons l'arête g dans F. Maintenant soit $F = \emptyset$ soit F est pair.

Si C_i est pair Nous trouvons $F \subseteq \delta(C_i)$ avec F impair comme suit en utilisant la même démarche que ci-dessus. À la fin, $|F| \geqslant 1$ et impair.

Dans les deux cas, une fois la composante connexe C_i et l'ensemble F calculés, nous testons si l'inégalité équivalente suivante est violée. Si

$$\sum_{e \in \delta(C_i) \setminus F} \bar{y}_e + \sum_{e \in F} (1 - \bar{y}_e) < 1$$

alors l'inégalité est violée.

La condition $h_e > 0$ peut être remplacée par la condition $h_e > \varepsilon$. Dans notre implémentation, nous essayons d'abord la procédure avec $\varepsilon = 0$. Si aucune coupe n'est trouvée, nous recommençons avec $\varepsilon = 0{,}2$ puis $\varepsilon = 0{,}4$ si nécessaire. Si l'heuristique ne produit aucune coupe, alors nous utilisons la procédure exacte suivante.

Procédure exacte pour la séparation

Cet algorithme est basé sur celui présenté par Padberg et Rao (voir Padberg et Rao, 1982). Comme eux, nous calculons une coupe impaire de poids minimum dans un graphe G'. Soit $\{\bar{y}_e\}$ une solution relaxée aggrégée pour les périodes de temps : $\bar{y}_e = \sum_{k=1}^{m+1}(y_{ij}^k + y_{ij}^k)$ avec $e = (ij)$. Nous construisons alors le graphe G' comme suit. La figure 5.24 reprend la construction d'un tel graphe pour la solution relaxée présentée sur la figure 5.22.

Initialement, G' contient tous les sommets $v \in \overline{V} \cup \{s,t\}$ du graphe auxiliaire éclaté et les arêtes e lorsque $\bar{y}_e > 0$. Tous les sommets sont étiquetés « impair ». Puis, pour chaque arête $e = (i,j)$ un nouveau sommet i_e et deux nouvelles arêtes sont créées comme sur la

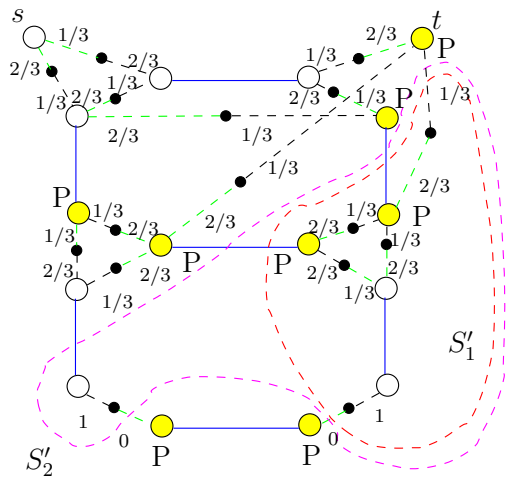


FIGURE 5.24 Le graphe G' utilisé pour séparer les inégalités *generalized co-circuit* (5.22). Les sommets marqués par un P sont étiquetés « pair », tous les autres sont étiquetés « impair ». L'ensemble S'_1 indique une coupe violée car il s'agit bien d'une coupe impaire avec 7 sommets étiquetés « impair » et de coût $1/3 < 1$. L'ensemble S'_2 indique une coupe violée du type du troisième cas.

figure 5.25 :

(i,i_e) avec un flot de $1 - \bar{y}_e$ et

(i_e,j) avec un flot de $\bar{y}_e$.

et l'arête $e = (i,j)$ est enlevée. Le nouveau sommet reçoit le l'étiquette « impair » et l'étiquette du sommet i est changé de « pair » à « impair » ou vice-versa.

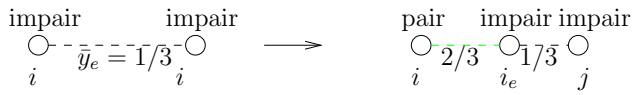


FIGURE 5.25 Comment transformer une arête dans le graphe G'.

Nous ajoutons maintenant toutes les arêtes $e \in \overline{E}_r$ avec un flot de 0 pour obtenir un graphe G' connexe. Soit une coupe (*edge-cut*) impaire (i.e. contenant un nombre impair de sommets étiquetés « impair ») minimum. Si le coût de cette coupe est $\geqslant 1$, il n'y a pas d'inégalité violée. Par contre, si la coupe a un coût < 1, nous avons une inégalité violée. Soit S' l'ensemble de sommets d'un côté de la coupe impaire. S' possède donc un nombre impair de sommets étiquetés « impair ». Soit $S = S' \cap (\mathrm{V} \cup \{s,t\})$, c'est-à-dire les sommets originaux du graphes auxiliaire éclaté et soit F le sous-ensemble des arêtes $e \in \delta(S)$ telles que les arêtes correspondantes avec un flot de $1 - \bar{y}_e$ sont dans la coupe $\delta(S')$. On peut

démontrer que l'inégalité correspondante est violée.

Pour calculer une coupe impaire de coût minimum, nous utilisons la procédure de Padberg et Rao. Leur algorithme calcule un arbre de Gomory-Hu de coupes minimales. Chaque arête de l'arbre associée à une coupe impaire minimale de coût < 1 produit une inégalité violée.

Sur la figure 5.24, l'ensemble S_1' représente une coupe impaire car S_1' contient 7 sommets étiquetés « impair » et de coût $= 1/3 < 1$. L'inégalité violée correspond dans le graphe auxiliaire éclaté de la figure 5.22 aux ensembles :

$$\begin{aligned} S &= \{4_3,4_4,4_6,6_6\} \qquad |S| \text{ pair} \\ F &= \{(6_6,6_7)\} \qquad |F| \text{ impair} \end{aligned}$$

que nous avions déjà obtenue à la section précédente.

L'ensemble S_2' contient 9 sommets étiquetés « impair » et représente une coupe de coût $= 1/3 < 1$. L'inégalité violée correspond aux ensembles :

$$\begin{aligned} S &= \{5_5,4_3,4_4,4_6,6_6\} \qquad |S| \text{ impair} \\ F &= \{(5_5,5_7),(6_6,6_7)\} \qquad |F| \text{ pair.} \end{aligned}$$

Remarques

1. Les inégalités *odd set* sont respectées sur la figure 5.23 puisque ces inégalités disent tout simplement que $y(\delta(S)) \geqslant 1$. Les inégalités *generalized co-circuit* sont donc une généralisation de celles-ci. Il suffit de prendre $F = \emptyset$.
2. Pour l'implémentation de la méthode de Padberg et Rao, nous utilisons explicitement un calcul des arbres de Gomory-Hu. Il existe une implémentation plus rapide proposée par Letchford, Reinelt et Theis (voir Letchford *et al.*, 2008) mais vu la taille relativement modeste des graphes que nous solutionnons ce n'est pas vraiment nécessaire. Les arbres de Gomory-Hu sont d'ailleurs calculés par l'élégante procédure proposée par Gusfield (voir Gusfield, 1990) qui ne nécessite pas de contraction des sommets.

5.7 Les règles de branchement

Nous en présentons trois dans cette section. La première, *fix max edge orientation*, semble *a priori* une excellente idée mais nous verrons dans la section 5.8.2 concernant les résultats expérimentaux que finalement cette règle n'est pas très bonne. La deuxième, *fix max flow two vertices*, peut paraître trop simple à première vue mais va se révéler être la meilleure des trois. Dans les remarques qui suivent la formalisation de la règle, nous tenterons d'expliquer pourquoi elle se comporte si bien. Enfin, la troisième, *parity*, nous est suggéré par des situations que les inégalités *generalized co-circuit* ne peuvent séparer.

Une règle de branchement ne vaut rien sans une règle adéquate pour sélectionner les nœuds. SCIP met à notre disposition cinq règles (voir Achterberg, 2008, chapitre 6) de sélection des nœuds dans un arbre de recherche :

`nodesel_bfs` sélectionne le meilleur nœud;

`nodesel_dfs` sélectionne le nœud suivant une recherche en profondeur d'abord;

`nodesel_estimate` sélectionne le nœud avec la meilleure estimation fournie par SCIP;

`nodesel_hybridestim` un mixte entre `nodesel_bfs` et `nodesel_estimate`;

`nodesel_restartdfs` sélectionne le nœud suivant une recherche en profondeur d'abord et de temps en temps sélectionne le meilleur nœud.

Si très vite, nous avons pu éliminer les règles `nodesel_dfs` et `nodesel_restartdfs`, nous n'avons pu, malgré des tests intensifs, différencier véritablement et de manière probante les trois autres règles. Ceci est sans doute dû à la taille relativement modeste des graphes sur lesquels nous sommes capables de résoudre le problème du postier chinois cumulatif. De manière quelque peu arbitraire, nous avons opté pour la règle `nodesel_bfs` qui sélectionne le nœud dont la relaxation linéaire est la plus grande.

5.7.1 Le branchement *fix max edge flow direction*

Ce branchement est peut-être le premier branchement auquel on pense en voyant des solutions relaxées fractionnaires. La figure 5.26 montre typiquement ce qui se passe localement pour les flots qui transitent par une arête originale : les flots peuvent aller dans les deux sens sur l'arête.

Ce branchement permet d'empêcher localement cette situation.

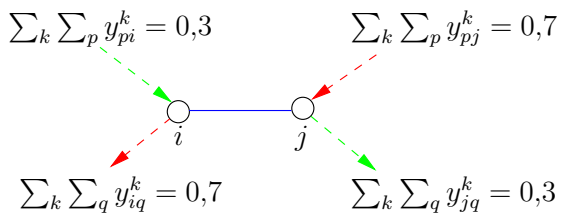


FIGURE 5.26 Dans une solution fractionnaire relaxée, les flots peuvent aller dans les deux sens sur une arête originale.

Idée

L'idée est de fixer le sens du flot sur une arête $e = (i,j) \in \overline{E}_r$ du graphe original. Pour nous donner une certaine latitude, nous utilisons un paramètre $\varepsilon \in]0{,}1/2]$ qui nous permet de décider si nous choisissons une arête pour laquelle le flot est équitablement réparti dans les deux sens ($\varepsilon \approx 0{,}5$) ou un peu plus réparti dans un sens que dans l'autre ($\varepsilon \approx 0{,}3$) ou encore au contraire très fort réparti dans un sens et pas trop dans l'autre ($\varepsilon \approx 0{,}1$). Nous choisissons l'arête pour laquelle

$$|\varepsilon - \min\,(\sum_k \sum_p y^k_{pi}, \sum_k \sum_q y^k_{iq})|$$

est minimal.

Formalisation

Pour une telle arête $e = (i,j) \in \overline{E}_r$, nous rajoutons deux nœuds à l'arbre de recherche : un premier où le flot est dirigé de i vers j et un second où le flot est dirigé dans le sens inverse de j vers i. Dans les deux cas, nous rajoutons une contrainte locale pour forcer le flot dans un sens ou l'autre :

$$\sum_k \sum_p y^k_{pi} = 1 \qquad \text{pour forcer le flot de } i \text{ à } j \text{ et}$$

$$\sum_k \sum_p y^k_{pj} = 1 \qquad \text{pour forcer le flot de } j \text{ à } i.$$

Remarques

Cette règle de branchement ne suffit pas pour la recherche complète d'un optimum. Il se peut très bien que nous aboutissions à une branche de l'arbre de recherche où tous les sens des flots sur les arêtes originales ont été fixées sans pour autant pouvoir en déduire quoi que ce soit. La figure 5.27 illustre un tel exemple. Dans ce cas, nous utilisons un branchement complet supplémentaire qui vient suppléer notre branchement *fix max edge flow direction* lorsque celui-ci est incapable de subdiviser une branche de l'arbre de recherche. Nous développerons une telle règle de branchement dans la section suivante.

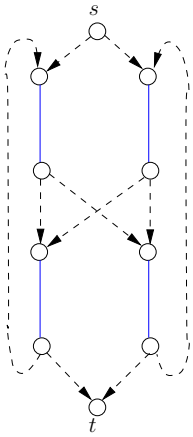


FIGURE 5.27 Un exemple qui ne peut pas être séparé par le branchement *fix max edge flow direction*. Toutes les arêtes de plus courts chemins ont un flot de 1/2.

5.7.2 Le branchement *fix max flow two vertices*

Entre deux sommets non reliés par une arête originale, soit il y a du flot, soit il n'y en pas. Ce constat tout simple va nous permettre pourtant de développer une règle de branchement relativement efficace.

Idée

L'idée est de fixer un flot agrégé fractionnaire entre deux sommets : dans toute solution réalisable ce flot est à 1 ou à 0. Comment choisir les deux sommets ? L'idée est de donner un petit coup de pouce à toute solution fractionnaire relaxée, donc nous choisissons deux sommets qui ont un flot agrégé non nul entre eux. Pour nous donner une certaine latitude, nous utilisons un paramètre $\varepsilon \in]0,1[$ qui nous permet de décider si nous prenons deux

sommets qui ont un flot agrégé important ($\varepsilon \approx 1$) ou au contraire un flot agrégé moyen ($\varepsilon \approx 1/2$) ou encore un flot agrégé insignifiant mais réel ($\varepsilon \approx 0$). Nous prenons les deux sommets i et j pour lesquels

$$|\varepsilon - \sum_k (y_{ij}^k + y_{ji}^k)|$$

est minimal.

Formalisation

Soient i et j deux tels sommets. Nous rajoutons deux nœuds à l'arbre de recherche : un premier où le flot agrégé entre i et j est nul et un second où le flot agrégé est égal à 1. Dans les deux cas, nous rajoutons une contrainte locale :

$$\sum_k (y_{ij}^k + y_{ji}^k) = 1 \qquad \text{pour forcer le flot entre } i \text{ et } j \text{ et}$$
$$\sum_k (y_{ij}^k + y_{ji}^k) = 0 \qquad \text{pour empêcher tout flot entre } i \text{ et } j.$$

Remarques

Cette règle de branchement est complète dans le sens où elle permet la recherche de tout l'arbre de recherche. Nous l'utiliserons fréquemment pour poursuivre la recherche lorsque d'autres règles de branchement ne peuvent plus séparer une solution fractionnaire.

Cette règle de branchement est plus élaborée qu'il n'y paraît de prime abord. La situation décrite sur la figure 5.28 ne peut pas être séparée par une inégalité *generalized co-circuit* (5.22).

L'ensemble S est pair et pourtant $y(\delta(S)) = 3$. Nous savons que dans ce cas, $y(\delta(S)) \leqslant 2$ ou $y(\delta(S)) \geqslant 4$. En regardant de plus près, nous nous rendons compte que le problème se situe au niveau des sommets i et j : $y(\delta(\{i,j\})) = 1$ or nous devrions plutôt avoir $y(\delta(\{i,j\})) = 0$ ou $y(\delta(\{i,j\})) \geqslant 2$. C'est exactement le branchement qu'opère notre règle lorsque $\varepsilon = 1/2$.

5.7.3 Le branchement *parity*

En regardant localement des solutions relaxées fractionnaires qui ne peuvent être séparées par les inégalités *generalized co-cicrcuit*, on peut se demander s'il n'y a pas moyen de brancher de façon efficace. Voici une tentative.

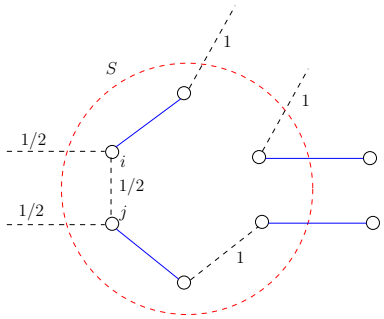


FIGURE 5.28 Voici une situation que les inégalités *generalized co-circuit* ne peuvent séparer. En exigeant qu'un flot passe ou non entre les deux sommets i et j, nous pouvons créer deux nœuds dans l'arbre de recherche qui sont débarrassés de cette situation.

Idée

Les inégalités *generalized co-circuit* ne peuvent pas trancher dans certains cas simples comme ceux de la figure 5.29.

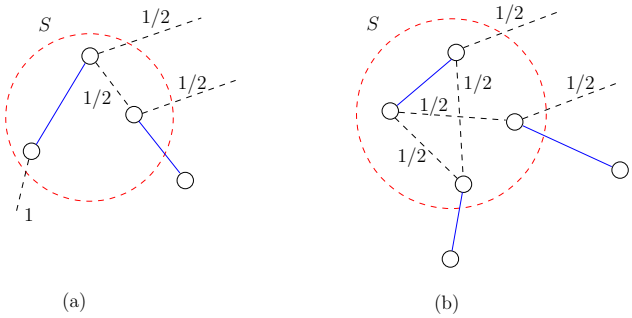


FIGURE 5.29 Situations où la règle de branchement *parity* peut s'appliquer. Les flots sur les plus courts chemins sont indiqués. Dans le cas (a), l'ensemble S est impair et $y(\delta(S)) = 2$ alors que dans le cas (b), l'ensemble S est pair et $y(\delta(S)) = 1$.

Dans le cas (a) de la figure 5.29, l'ensemble S est impair et $y(\delta(S)) = 2$. Clairement, dans toute solution réalisable soit $y(\delta(S)) = 1$, soit $y(\delta(S)) = 3$. Le cas (b) est similaire. L'ensemble S est pair et $y(\delta(S)) = 1$ alors que dans toute solution réalisable soit $y(\delta(S)) = 0$, soit $y(\delta(S)) \geqslant 2$.

Formalisation

Soit une solution relaxée et soit S un ensemble de sommets tel que $y(\delta(S)) \notin \mathbb{N}$ ou bien tel que cet ensemble soir pair alors que $y(\delta(S))$ est impair ou vice-versa, alors la règle de branchement *parity* créé deux nœuds dans l'arbre de recherche. Utilisons une nouvelle notation pour nous simplifier la vie. Soient $\lfloor \alpha \rfloor_p$ et $\lceil \alpha \rceil_p$ le plus grand entier pair $\leqslant \alpha$ et le plus petit entier pair $\geqslant \alpha$ respectivement. Par exemple, $\lfloor 3{,}51 \rfloor_p = 2$ et $\lceil 3{,}51 \rceil_p = 4$. De la même manière, définissons $\lfloor \alpha \rfloor_i$ et $\lceil \alpha \rceil_i$ mais avec comme résultats des nombres impairs. Par exemple, $\lfloor 3{,}51 \rfloor_i = 3$ et $\lceil 3{,}51 \rceil_i = 5$.

Pour tout ensemble S pair (impair), la règle créé deux nœuds pour lesquels

$$\sum_k \sum_{\substack{i \in S \\ j \notin S}} (y_{ij}^k + y_{ji}^k) \leqslant \lfloor y(\delta(S)) \rfloor_p \qquad (\lfloor y(\delta(S)) \rfloor_i) \tag{5.23}$$

$$\sum_k \sum_{\substack{i \in S \\ j \notin S}} (y_{ij}^k + y_{ji}^k) \geqslant \lceil y(\delta(S)) \rceil_p \qquad (\lceil y(\delta(S)) \rceil_i) \tag{5.24}$$

sont localement valides respectivement.

Remarques

Nous choisissons un ensemble S (pair ou impair) pour lequel la plus petite des deux différences en valeur absolue entre les membres de droite et de gauche de (5.23) et (5.24) est maximale. Par exemple, entre un ensemble S_1 avec $|S_1| = 5$ et $y(\delta(S_1)) = 3{,}1$ et un ensemble S_2 avec $|S_2| = 6$ et $y(\delta(S_2)) = 3{,}1$, nous choisissons S_2 car la plus petite différence en valeur absolue pour S_1 est $\min\{|3 - 3{,}1|, |5 - 3{,}1|\} = 0{,}1$ alors que nous obtenons $\min\{|2 - 3{,}1|, |4 - 3{,}1|\} = 0{,}9$ pour S_2.

Ce choix est fait à l'aide d'une heuristique qui nous permet de sélectionner des ensembles pairs, impairs ou les deux à la fois et de privilégier de petits ou grands ensembles de sommets. Nous expliquerons les différents paramètres de cette heuristique dans la section 5.8.2 page 137 lorsque nous testerons les différentes règles de branchement.

Si nous nous limitons à des ensembles S de deux sommets, cette règle n'est rien d'autre que la règle *fix max flow two vertices*. Si nous prenons seulement des ensembles S de trois sommets ou plus, la règle n'est plus complète et doit être suppléée par un branchement

complet comme *fix max flow two vertices*. La figure 5.30 montre une solution relaxée qui ne peut être séparée dans ce cas.

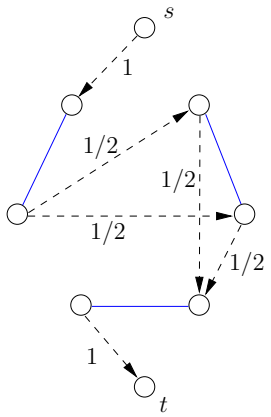


FIGURE 5.30 Voici une solution que la règle de branchement *parity* ne peut séparer si nous considérons seulement des ensembles S avec $|S| \geqslant 3$.

5.8 Résultats expérimentaux

Dans cette section, nous testons empiriquement les outils que nous avons développés tout au long de ce chapitre. Les résultats obtenus nous serviront de référence pour le prochain chapitre.

Dans le cadre de cette thèse, nous considérons les solveurs comme des boîtes noires auxquelles nous ne touchons pas. Les deux solveurs retenus pour résoudre les programmes linéaires (PL) sont SoPlex et CPLEX. Le premier est libre et l'autre non. Bien que SoPlex ne soit pas le solveur libre le plus rapide[8] qui s'offre à nous, c'est le solveur fourni par défaut avec SCIP et c'est celui qui bénéficie de la meilleure intégration avec ce dernier. Bien que relativement lent, il nous paraît extrêmement stable. En fait, c'est le seul solveur dont la stabilité à toute épreuve nous a permis d'utiliser sans anicroche les approches présentées dans le reste de cette thèse. Nous utilisons aussi le solveur CPLEX pour résoudre les PL, ce qui nous permet de comparer de manière plus équitable nos résultats avec le solveur CPLEX.

Remarquons aussi que si CPLEX a gagné en stabilité, nos premiers essais avec CPLEX 9

8. Voir les études comparatives de Hans Mittelmann sur le site www.neos-server.org/neos/

ne furent guère concluants[9]. Nous reviendrons sur ces problèmes d'instabilité dans le prochain chapitre.

5.8.1 Les pré-traitements sur les variables

Pour juger de l'utilité de nos algorithmes de pré-traitement sur les variables nous faisons deux types de comparaisons. Tout d'abord, une comparaisons du nombre de variables éliminées par nos pré-traitements avec le pré-traitement utilisé par différents solveurs bien connus. Ensuite, pour juger de la qualité de ceux-ci, nous comparons les temps nécessaires à la résolution du problème du postier chinois cumulatif sur quelques graphes de référence avec et sans pré-traitement par notre algorithme de séparation et évaluation progressive.

Comparaison du nombre de variables éliminées

Nous avons comparé le nombre de variables éliminées par pré-traitement par les solveurs SCIP et CPLEX avec nos propres pré-traitements. La figure 5.31 résume ces comparaisons sur les quinze graphes de référence. Nous utilisons les versions de base des pré-traitements offerts par ces solveurs.

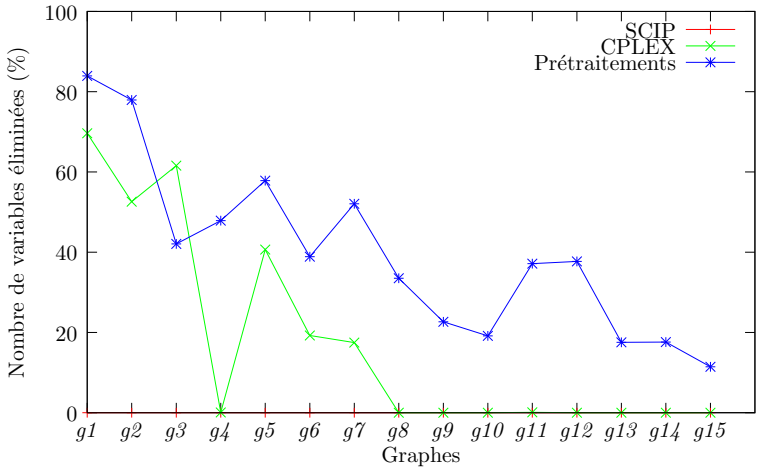


FIGURE 5.31 Une comparaison du nombre de variables éliminées par pré-traitement.

Remarquons plusieurs tendances sur la figure 5.31. Tout d'abord, plus les graphes sont

9. Par exemple pour le graphe *g4*, CPLEX 9 nous donnait 12 591 504 comme valeur optimale alors que la vraie valeur optimale est 12 591 503.

imposants (graphes *g9-g10* et *g13-g15*), moins le nombre de variables éliminées est grand proportionnellement quel que soit le pré-traitement considéré. Plus inquiétant, pour les graphes les plus imposants, CPLEX n'est pas capable d'éliminer beaucoup de variables. SCIP quant à lui est incapable d'éliminer quoi que ce soit et ce peu importe la taille des graphes. Nous éliminons donc plus de variables (sauf pour le graphe *g3*), mais c'est plutôt normal puisque nous avons une meilleure connaissance du problème. Dans la section suivante, nous verrons aussi que ces pré-traitements nous permettent de faire des gains appréciables en temps pour la résolution du problème.

Nous n'avons pas jugé pertinent de faire une comparaison pour les temps nécessaires aux pré-traitements : pour nos plus grands graphes de référence ces temps n'excédaient pas les 0,17 secondes pour l'ensemble de nos algorithmes de pré-traitement.

Comparaison des temps de résolution avec et sans pré-traitement sur les variables

Pour des graphes de taille raisonnable[10], nous obtenons des gains en temps. Le solveur utilisé est SoPlex. Ceux-ci sont reproduits sur la figure 5.32.

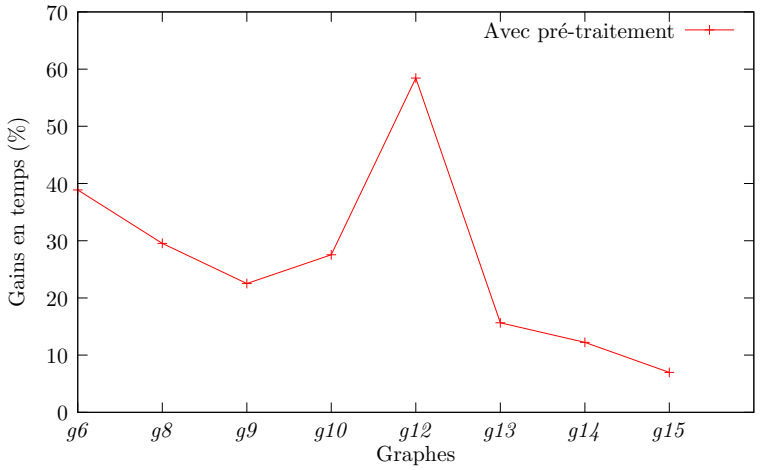


FIGURE 5.32 Gains en temps avec les pré-traitements.

Le cas le plus impressionnant est celui du graphe *g12* où nous obtenons un gain en temps de résolution de 58,43 %, mais même pour des graphes plus conséquents comme le graphe *g15*, nous obtenons un gain appréciable de 6,97 %. Nous reviendrons sur le pré-traitement des variables à la section 6.1 lorsque nous aborderons la génération de variables.

10. Nous avons omis les graphes *g1-g5*, *g7* et *g11* car leurs temps de résolution sont trop courts ($\leqslant$ 0,10 secondes) pour être significatifs.

5.8.2 Les règles de branchement

Nous commençons d'abord l'étude des branchements *fix max flow two vertices* et *fix max edge flow direction* séparément pour déterminer quelles valeurs donner aux paramètres respectifs. Pour ce faire, nous les utiliserons sans ajouter de coupe.

Le cas du branchement *parity* est différent. Comme il a été conçu pour palier les manques des inégalités *generalized co-circuit*, il ne sert à rien de le tester sans la présence de celles-ci. Plusieurs tentatives d'implémentation de l'idée sous-jacente au branchement *parity* ont été testées. Malheureusement, aucune d'entre-elles ne s'est révélée véritablement efficace. Ce branchement se révèle donc être le moins performant des trois.

Comme annoncé dans la section 5.7, nous choisissons systématiquement le nœud dont la relaxation linéaire prend la plus grande valeur.

Nous commençons d'abord par traiter la règle de branchement *fix max flow two vertices* car nous l'utiliserons en parallèle lorsque nous traiterons des deux autres branchements.

fix max flow two vertices

La table 5.5 nous donne le nombre de nœuds dans l'arbre de recherche ainsi que les temps de résolution en secondes pour trois valeurs du paramètre ε. Rappelons que celui-ci correspond à un flot cible entre deux sommets. Nous privilégions la paire de sommets dont le flot agrégé entre les deux se rapproche le plus de ε.

Tableau 5.5 Nombre de nœuds et temps pour *fix max flow two vertices*.

Paramètres	Graphes			
	g6	*g8*	*g12*	*g13*
	Nombre de nœuds			
$\varepsilon = 0{,}5$	17	5	3	137
$\varepsilon = 0{,}9$	17	9	5	207
$\varepsilon = 0{,}1$	31	17	17	579
	Temps en secondes			
$\varepsilon = 0{,}5$	0,97	2,93	2,69	286,59
$\varepsilon = 0{,}9$	1,12	3,54	4,01	359,49
$\varepsilon = 0{,}1$	1,46	4,31	3,66	741,68

Sans surprise, la valeur 1/2 se révèle être le meilleur choix tant pour le nombre de nœuds obtenus que par rapport aux temps nécessaires pour la résolution du problème. C'est pour cette valeur que nous perturbons le plus les solutions fractionnaires comme nous l'avons mentionné dans les remarques qui suivent l'introduction de cette règle page 131. Nous fixerons dorénavant ε à 1/2.

fix max edge flow direction

Le tableau 5.6 présente les résultats avec la règle *fix max edge flow direction*. Comme cette règle de branchement n'est pas complète, nous la suppléons avec notre règle *fix max flow two vertices* lorsque la première est dans l'impossibilité de trouver de nouveaux nœuds. Dans le tableau 5.6, les lignes de la première partie correspondent respectivement au nombre de nœuds et au nombre d'appels à chacune des deux règles de branchement (tout d'abord *fix max edge flow direction*, puis *fix max flow two vertices*). Par exemple, pour le paramètre $\varepsilon = 0{,}5$, nous voyons que pour le graphe *g6* nous avons dû appeler notre branchement 18 fois (et donc créer 36 nœuds) ainsi que le suppléer une fois (avec création de deux nœuds supplémentaires). Rappelons que le paramètre $\varepsilon \in]0{,}1/2]$ est un flot cible pour le plus petit flot agrégé qui entre ou sort d'un sommet d'une arête originale. Dans la seconde partie, les lignes correspondent aux temps d'exécution pour les diverses valeurs d'ε considérées.

Tableau 5.6 Nombre de nœuds et temps pour *fix max edge flow direction*.

Paramètres	Graphes			
	g6	*g8*	*g12*	*g13*
	Nombre de nœuds			
$\varepsilon = 0{,}5$	31	13	3	371
	18	8	1	183
	1	0	0	4
$\varepsilon = 0{,}3$	33	13	5	505
	17	6	3	250
	1	0	0	4
$\varepsilon = 0{,}1$	75	87	43	2511
	29	39	20	1147
	8	4	1	109
	Temps en secondes			
$\varepsilon = 0{,}5$	1,55	4,59	2,83	652,01
$\varepsilon = 0{,}3$	1,56	3,84	3,58	822,32
$\varepsilon = 0{,}1$	1,49	5,28	3,62	1020,60

Ici encore, nous trouvons 1/2 comme valeur optimale du paramètre ε. C'est effectivement aussi pour cette valeur que nous modifions le plus les solutions fractionnaires. Remarquons que dans la plupart des cas, nous avons été obligés de faire entrer en action la règle *fix max flow two vertices*.

parity et comparaison des trois règles de branchement

Comme le branchement *parity* sert à séparer localement des situations que les inégalités *generalized co-circuit* ne peuvent pas traiter, il ne sert à rien de le tester sans l'utilisation de ces inégalités. Nous avons essayé plusieurs variantes de ce branchement, mais comme les tableaux 5.7 et 5.8 l'attestent, ce branchement est manifestement moins performant que les deux autres. Dans le tableau 5.7, nous comparons le nombre de nœuds obtenus avec 6 variantes du branchement *parity* à celui obtenu par les meilleures versions des branchements *fix max flow two vertices* et *fix max edge flow direction* sur cinq nouveau graphes. Ceux-ci ont été choisis de sorte qu'il n'y ait pas d'appel à la règle *fix max flow two vertices*. Le but ici est d'évaluer la performance des deux règles seules, sans interférence avec la règle *fix max flow two vertices*.

Explicitons les paramètres du tableau 5.7. La première colonne (parité) indique si nous nous concentrons sur des ensembles de sommets impairs (i), pairs (p) ou les deux (p+i). Les deuxième et troisième colonnes (min et max) indiquent respectivement le nombre minimal et maximal de sommets considérés. Notons que dans les deux dernières lignes, nous tolérons des ensembles à seulement deux sommets. La quatrième colonne (P/G) montre si - pour des situations identiques - nous privilégions les petits (P) ou les grands (G) ensembles de sommets. Nous retrouvons les temps de résolution correspondant dans le tableau 5.8.

Inutile de nous attarder sur une discussion des meilleurs paramètres pour la règle *parity*, car, comme le montrent les deux dernières lignes des tableaux 5.7 et 5.8, les deux autres règles de branchement sont plus efficaces.

Pour une comparaison utile entre les branchements *fix max flow two vertices* et *fix max edge flow direction*, nous nous tournons vers la figure 5.33. Celle-ci montre le gain relatif en temps de l'agorithme BC1 en utilisant *fix max flow two vertices* par rapport à l'utilisation de *fix max edge flow direction* (avec la règle *fix max flow two vertices* pour suppléer lorsque nécessaire) lorsque nous utilisons les inégalités valides vues dans ce chapitre. Ces gains ne sont pas très importants mais ils sont nets et nous avons tout intérêt à utiliser la règle de

Tableau 5.7 Nombre de nœuds pour les trois règles de branchement.

Paramètres				Graphes				
				b1	*b2*	*b3*	*b4*	*b5*
Branchement *parity*								
parité	min	max	P/G	Nombre de nœuds				
p	3	10	P	55	49	23	3	55
p	3	10	G	39	63	31	5	62
p+i	3	10	P	17	141	31	3	55
p+i	3	10	G	39	123	27	5	61
p	2	10	P	9	19	25	3	71
p+i	2	10	P	11	17	33	3	65
Branchement *fix max flow two vertices*								
$\varepsilon = 1/2$				9	21	15	5	45
Branchement *fix max edge flow direction*								
$\varepsilon = 1/2$				12	39	13	5	9

Tableau 5.8 Temps en secondes pour les trois règles de branchement. L'ordre des lignes correspond à celui du tableau 5.7.

b1	*b2*	*b3*	*b4*	*b5*
22,79	23,25	17,26	3,33	32,99
20,08	26,39	19,59	4,00	35,34
15,76	70,47	18,06	3,39	33,20
20,27	63,74	17,27	4,10	34,09
8,75	12,06	21,09	3,82	41,30
9,99	10,28	15,66	3,82	38,32
6,79	12,42	8,51	4,60	17,64
7,99	18,91	8,53	3,71	7,42

branchement *fix max flow two vertices*.

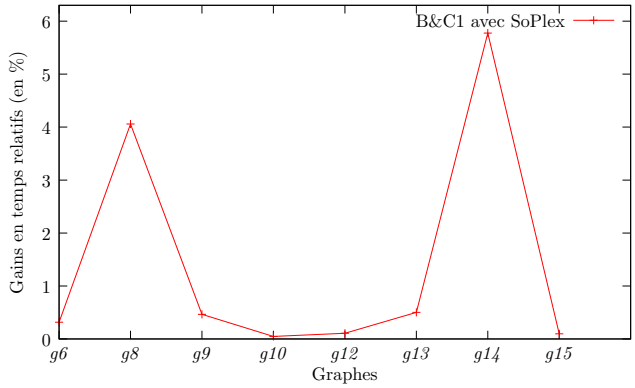


FIGURE 5.33 Gains en temps relatifs (en %) avec l'utilisation de la règle de branchement *fix max flow two vertices* par rapport à l'utilisation de la règle *fix max edge flow direction* lorsque nous utilisons toutes les inégalités vues dans ce chapitre avec l'algorithme BC1.

5.8.3 Les inégalités valides

Pour montrer l'efficacité des inégalités valides, nous comparons la résolution d'abord avec seulement les inégalités *odd set*, puis nous rajoutons les inégalités *even set* et enfin nous rajoutons les inégalités *generalized co-circuit*. Cet ordre est optimal et est le seul que nous présentons. Notre expérience nous a aussi montré qu'il valait mieux utiliser toutes les coupes détectées. Le branchement utilisé est *fix max flow two vertices*. Nous ne comparons pas nos résultats à la résolution sans coupe, car SCIP est incapable dans ce cas de résoudre les graphes *g9* et *g10-g15* dans des temps raisonnables.

Le tableau 5.9 présente le nombre de nœuds, les temps nécessaires ainsi que le nombre de coupes utilisées à chaque fois. Ainsi, pour le graphe *g6*, le nombre de coupes *odd set* utilisées est 15, ce que nous indiquons entre parenthèses (15). Dans la deuxième ligne, (15/1) représente l'utilisation de 15 coupes *odd set* et 1 coupe *even set*. Finalement, dans la troisième ligne, (15/1/1) indique l'utilisation de 15 coupes *odd set*, 1 coupe *even set* et 1 coupe *generalized co-circuit*.

Le tableau 5.9 nous montre clairement l'efficacité des inégalités *odd set* et *even set*. Par contre, l'ajout des inégalités *generalized co-circuit* semble contre-productive. En fait, il faut être plus nuancé et nous reviendrons sur leur efficacité au prochain chapitre lorsque nous

Tableau 5.9 Nombre de nœuds, temps nécessaires (en secondes) et nombres de coupes de parité utilisées pour 5 graphes de référence.

Graphes :	*g6*			*g8*			*g9*			*g10*			*g13*		
Inégalités	n	t	#	n	t	#	n	t	#	n	t	#	n	t	#
odd set	3	0,54	(15)	5	4,47	(40)	91	726,59	(79)	5	186,65	(44)	3	61,10	(27)
+ *even set*	1	0,56	(15/1)	5	4,31	(34/3)	15	278,81	(81/2)	1	169,47	(28/2)	3	60,44	(27/1)
+ *gen. co-circuit*	1	0,57	(15/1/1)	5	4,58	(33/3/7)	15	296,64	(75/2/11)	1	174,03	(28/2/3)	3	58,33	(27/2/1)

introduirons de nouvelles coupes.

5.8.4 Comparaisons des temps de résolution

Nous prenons tous les ingrédients vus jusqu'ici pour implémenter un algorithme de *Branch and Cut* BC1. La figure 5.34 nous montre la comparaison des temps de résolution pour les graphes[11] *g9-g10* et *g13-g15*. Les temps donnés sont relatifs par rapport au temps de résolution de CPLEX, *i.e.* le temps pris par CPLEX vaut 100 %. Le temps reproduit est le temps effectif utilisé par la machine et non le temps réel qui est bien plus long. Nous forçons CPLEX à n'utiliser qu'un seul processeur à la fois car SCIP n'a pas été conçu pour utiliser plusieurs processeurs en parallèle[12]. Notre algorithme, qu'il utilise SoPlex, CLP ou CPLEX pour résoudre les PL, est plus rapide que CPLEX, ce qui s'explique en partie par le fait que nous exploitons notre connaissance du problème, ce que CPLEX ne peut faire.

Le tableau 5.10 présente les temps en secondes de notre algorithme implémenté avec les trois solveurs ainsi que les temps du solveur CPLEX.

Tableau 5.10 Comparaisons des temps (en secondes) entre les *Branch and Cut* avec différents solveurs ainsi que le solveur CPLEX.

	g9	*g10*	*g13*	*g14*	*g15*
CPLEX avec L8	8235,26	782,47	717,74	576,56	4700,50
BC1 avec SoPlex	296,64	174,03	58,33	402,65	794,44
BC1 avec CLP	226,49	85,67	27,34	364,44	359,82
BC1 avec CPLEX	140,44	63,36	20,65	196,81	316,29

11. Les temps de résolution pour les graphes *g1-g8* et *g11-g12* sont trop petits pour être significatifs.
12. Il existe une version en parallèle de SCIP, ParaSCIP (voir Shinano *et al.*, 2010) mais nous ne l'utilisons pas ici.

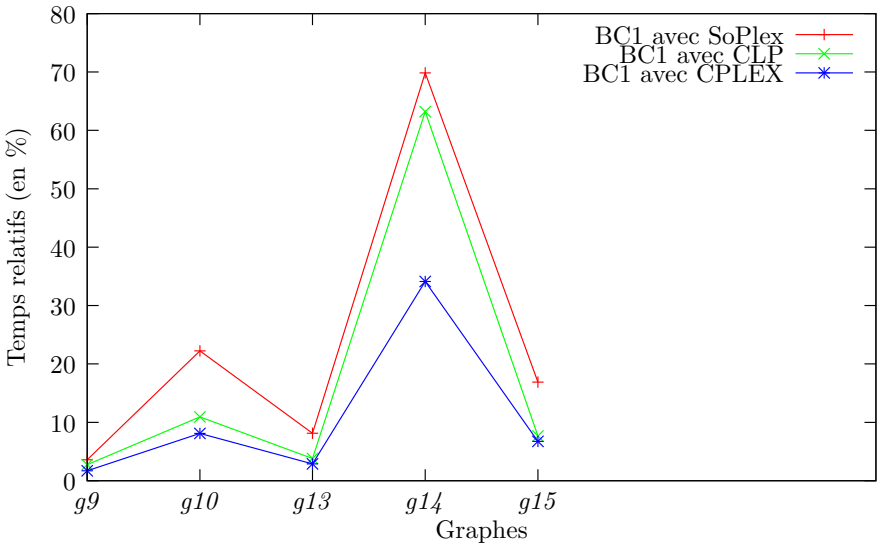


FIGURE 5.34 Comparaison des temps de résolution pour le BC1. Le temps de référence est le temps pris par CPLEX, *i.e.* pour tous les graphes ce temps vaut 100 %.

Nous voyons dans ce tableau 5.10 que notre algorithme avec CPLEX est de 3 (graphe *g14*) à 58 (graphe *g9*) fois plus rapide que CPLEX sur nos graphes de référence. En fait, le cas du graphe *g14* est atypique comme nous le verrons au chapitre suivant. En général, notre algorithme fonctionne 10 fois plus vite que CPLEX pour des graphes de la taille du graphe *g15*.

Comment est réparti le temps entre les différentes composantes de notre algorithme de séparation et évaluation progressive ? La figure 5.35 nous montre une répartition du temps typique pour nos plus gros graphes de référence. La grande majorité du temps (97,8 % - 783,61 secondes) est consacrée à la résolution des PL (par le simplex dual) et de ce temps, une part très importante (86,9 % - 690,00 secondes) est consacrée à la résolution du premier PL. Dans le chapitre suivant nous verrons comme réduire significativement ce temps.

Le reste de la machinerie - séparation et création des coupes (0,3 % - 2,70 secondes), prétraitements (0,06 % - 0,49 seconde[13]), l'heuristique (1,4 % - 10,94 secondes) ainsi que le reste (sélection des nœuds, maintien de l'arbre de recherche, échanges d'information entre SCIP et SoPlex, etc. : 0,44 % - 3,49 secondes) prend un temps relativement modeste. L'heuristique pour trouver une bonne solution de départ est invoquée cinq fois avec des paramètres différents.

13. Nos pré-traitements prennent 0,17 seconde mais il faut 0,32 seconde à SCIP pour les intégrer.

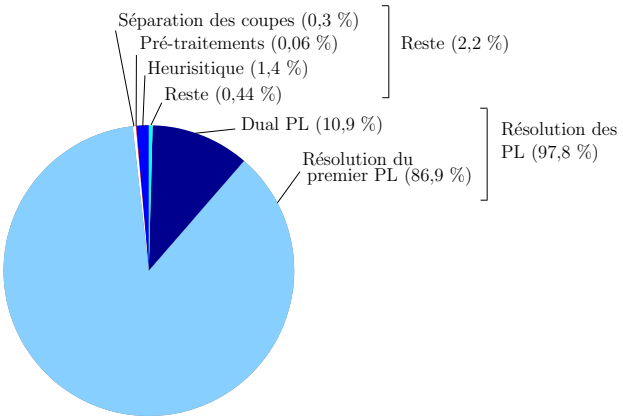


FIGURE 5.35 Répartition du temps pour le BC1 pour le graphe *g15*.

5.9 Conclusions

Le nombre astronomique de variables du modèle L8 suggère fortement une approche de génération de colonnes et ce d'autant plus que non seulement le nombre de variables requises dans toute solution optimale est minime (seulement $m + 1$ variables) mais aussi que le nombre de variables non nulles dans toute solution fractionnaire optimale est relativement restreint aussi. Par exemple, les 3 PL résolus pendant la recherche d'une solution optimale pour le graphe *g15* comprennent en moyenne 228 variables non nulles alors que le modèle contient 348 579 variables. De plus, en moyenne, il a fallu 14 997.33 itérations de la méthode du simplex dual par PL pour résoudre ceux-ci. Ces données sont encourageantes et nous permettent d'espérer que finalement peu de variables soient véritablement nécessaires dans la recherche d'un optimum avec la méthode du simplex.

Chapitre 6

RÉSOLUTION DU MODÈLE L8 : DEUXIÈME APPROCHE

> Moreover, these algorithmic advances have been incorporated in commercially inexpensive software that is readily available, easily portable, and supported by a variety of systems that make it possible for unsophisticated users to input and check their models and obtain understandable outputs
>
> George L. Nemhauser, (Nemhauser, 1994)

Les deux principales innovations de ce chapitre sont d'une part l'utilisation d'une approche de génération de colonnes et d'autre part l'introduction d'inégalités basées non plus sur les flots agrégés mais directement sur les flots dirigés eux-mêmes. L'algorithme utilisé ici est un *Branch, Price and Cut* où nous générons dynamiquement à chaque nœud de l'arbre de recherche aussi bien des variables que des contraintes supplémentaires.

SCIP est un merveilleux outil mais comme tout outil celui-ci a ses limitations. Nous n'avons pas trouvé comment implémenter de manière efficace un *Branch, Price and Cut* sans changer le code interne de SCIP. Si SCIP a été conçu pour fonctionner dans le cas d'un *Branch, Cut and Price*[1], nous pensons qu'il n'est pas vraiment adapté à notre approche, de sorte que nous avons aussi dû rajouter des surcouches notamment pour la mise à jour du dual à chaque nœud de l'arbre de recherche.

Mentionnons tout de suite que le solveur CLP doit être abandonné pour cette approche. Il n'arrive tout simplement pas à résoudre les PL qui sont trop instables numériquement pour lui. De plus, les PL obtenus sont dégénérés aussi bien au niveau primal qu'au niveau dual. Nous nous limitons donc à l'utilisation de SoPlex et CPLEX, le premier étant notre solveur

1. Nous faisons une distinction entre un *Branch, Price and Cut* et un *Branch, Cut and Price* où pour ce dernier toutes les coupes sont rajoutées une fois pour toute à la racine (voir par exemple Lübbecke et Desrosiers, 2005, page 1020).

de référence. Cet instabilité va, à partir de maintenant, devenir un autre thème récurrent dans cette thèse et constitue un problème majeur du modèle L8 et de ses variantes. Il faut constamment pouvoir contrôler les résultats numériques et les vérifier. Ceci est fait de façon automatisée par SCIP en temps normal mais dans notre cas, nous avons dû adapter le code car les instabilités numériques étaient trop grandes. Voici un exemple d'ajustement qui fut nécessaire. Les coûts réduits ne peuvent être calculés qu'à $\varepsilon = 0.01$ près, ce qui ne donne pas une précision très satisfaisante mais c'est la seule que semble garantir SoPlex. Par exemple, nous avons calculé que le coût réduit d'une variable est 110.94355147 alors que SoPlex la calcule à 110.94481367, ce qui fait une différence absolue plus grande que 10^{-3}. En fait, la précision est tellement mauvaise que nous avons dû implémenter nos propres fonctions de contrôle de précision. En effet, SCIP propose les fonctions `SCIPisEQ` et `SCIPisSumEQ` qui prennent en argument chacune un ε différent (par défaut, $\varepsilon = 10^{-9}$ et $\varepsilon = 10^{-6}$ respectivement). Il y a toutefois des bornes sur les valeurs possibles pour les ε : $\varepsilon \in [10^{-20}, 10^{-3}]$ pour `SCIPisEQ` et $\varepsilon \in [10^{-17}, 10^{-3}]$ pour `SCIPisSumEQ`, ce qui nous a obligé à utiliser nos propres versions.

Nous commençons ce chapitre par montrer comment nous générons nos colonnes dans la section 6.1, puis nous passons aux inégalités de flot dans la section 6.2. Nous discutons de la mise en pratique de toutes ces idées dans un algorithme BPC1 dans la section 6.3 consacrée aux résultats expérimentaux. Nous introduisons alors une petite astuce et créons l'algorithme BPC2 qui améliore un peu l'efficacité de l'algorithme BPC1 dans la section 6.4 et nous comparons les deux algorithmes dans la section 6.5. Finalement, la section 6.6 nous permet de conclure.

6.1 Génération de colonnes

6.1.1 Introduction

L'approche que nous avons adoptée est une version simplifiée de celle utilisée par Applegate, Bixby, Chvátal et Cook (Voir Applegate *et al.*, 2007, chapitre 12, pp. 345-372). Nous utilisons l'équivalence entre d'une part une variable du primal qui a un coût réduit négatif et d'autre part le fait que la contrainte duale correspondante est violée. Notre approche consiste simplement à générer les variables primales dont la contrainte duale correspondante est violée. Idéalement, le problème de trouver les contraintes duales violées, ou de manière équivalente les variables primales de coût réduit négatif, se fait en résolvant un problème

(*pricing problem*) plutôt qu'en énumérant toutes les variables. Malheureusement, nos tentatives dans ce sens n'ont pas abouti. Nous utiliserons donc une énumération explicite de toutes les variables. Cette approche donne déjà des résultats étonnants comme nous le verrons et montre tout le potentiel d'une approche de génération de colonnes.

Si l'idée de base est relativement simple, l'implémentation se révèle plus délicate comme l'atteste l'abondante littérature sur les problèmes inhérents à toute approche de génération de colonnes (Voir par exemple Lübbecke et Desrosiers, 2005; Applegate *et al.*, 2007). Comme l'idée est d'utiliser moins de variables, il serait contre-productif de générer toutes les variables qui ont un coût réduit négatif. D'un autre côté, nous avons vu que la résolution des PL est très coûteuse. Introduisons un nombre trop petit de telles variables et nous voilà obligés de solutionner un trop grand nombre de PL qui restent coûteux même avec un nombre réduit de variables. Introduisons un trop grand nombre de variables et nous nous retrouvons avec un nombre conséquent de variables dont nous pourrions nous passer.

A ce problème de dosage du nombre de variables de coût réduit négatif à introduire, se superpose l'utilisation dynamique de contraintes. En effet, à chaque nœud de l'arbre de recherche, nous rajoutons des coupes que nous ne pouvons connaître a priori. Donc, à chaque nœud, nous obtenons un PL dual différent. Il nous faudra donc faire cette mise à jour du PL dual de façon efficace.

Cette nouvelle approche nous oblige aussi à repenser les stratégies utilisées au chapitre précédent. Par exemple, les pré-traitements ne doivent plus être appliqués aveuglément. En effet, bien qu'ils éliminent des variables qui ne peuvent être utilisées dans une solution optimale, il se pourrait que l'introduction de ces variables permettent un nombre plus réduits de pivots dans la méthode du simplex. Nous verrons que tel est le cas.

Une autre problématique intéressante mais que nous passerons complètement sous silence ici, est la mise à jour des inégalités (pour les coupes ou les contraintes locales de branchement). Notre expérience nous a appris qu'en général il est plus intéressant de mettre à jour les inégalités avec toutes les variables nouvellement générées. Par contre, il est parfois avantageux de faire évoluer certaines de ces inégalités, par exemple les inégalités *generalized no joint flow* que nous verrons à la section 6.2.2, en retirant de ces inégalités certaines variables pour les remplacer par d'autres.

Nous avons vu que la richesse du modèle L8 nous permettait d'exprimer différemment une même idée et que cette écriture influençait dans la pratique l'efficacité de l'algorithme. Il faut donc retester nos écritures dans le cadre de l'utilisation d'un nombre restreint de variables.

Mentionnons que les inégalités valides obtenues dans la section 6.2 sont issues d'une collaboration avec les professeurs Ángel Corberán de l'Université de Valence et José María Sanchís de l'Université Polytechnique de Valence.

6.1.2 Le modèle L8$^+$

Les inégalités *vertex* sont parmi les inégalités de parité les plus efficaces. Comme elles ne sont pas nombreuses, nous les incorporons dorénavant dans le modèle L8. Voyons comment. Ces inégalités ont été définies page 106. Les revoici :

$$y(\gamma(I)) = \sum_{k=2}^{m} \sum_{\substack{(i,j)\in\overline{A}_{sp}:\\ *i=v=*j}} y_{ij}^k \leq \lfloor \frac{\deg(v)}{2} \rfloor \tag{5.4}$$

Les inégalités 5.4 sont valides que le sommet $v \in V$ soit de degré pair ou impair mais comme l'indique la proposition suivante, seul le cas où le degré est impair est intéressant.

Proposition 32 *Les* vertex inequalities *sont toujours respectées lorsque le sommet* $v \in$ V *est de degré pair.*

Démonstration

Il s'agit d'une conséquence directe du lemme 5.1 page 104 et du fait que l'on traite de sommets $v \in$ V de degré pair. Plus précisément [2] :

$$y(\gamma(I_v)) \leq \lfloor \frac{\deg(v)}{2} \rfloor = \frac{\deg(v)}{2} \Leftrightarrow y(\delta(I_v)) \geq |I_v| - 2(\frac{\deg(v)}{2}) = 0$$

Cette dernière inégalité est bien sûr respectée quel que soit le sommet pair $v \in$ V.

□

Puisqu'il n'est utile de rajouter l'inégalité que lorsque v est de degré impair, nous ajoutons le sommet fictif s ou le sommet fictif t lorsque nous rencontrons un sommet de degré pair.

2. Rappelons que pour tout sommet $v \in$ V, $I_v = \{i \in \overline{V} : *i = v\}$.

Après plusieurs essais, nous avons trouvé que la manière la plus efficace de le faire est la suivante. Soit un sommet $v \in V$ de degré plus grand ou égale à deux. Si le sommet est impair, nous rajoutons bien sûr $y(\gamma(I_v)) \leq \frac{\deg(v)-1}{2}$. Si le sommet est pair alors nous considérons deux cas. Soit ce sommet représente le dépôt et nous rajoutons $y(\gamma(I_v \cup \{s\})) \leq \frac{\deg(v)}{2}$. Soit il ne représente pas le dépôt et nous rajoutons $y(\gamma(I_v \cup \{t\})) \leq \frac{\deg(v)}{2}$. Nous appellerons le nouveau modèle ainsi obtenu par L8$^+$.

6.1.3 Le dual du modèle L8$^+$

Avant de décrire le problème dual, retranscrivons de manière plus détaillée le modèle L8$^+$ dans le tableau 6.1 avec les variables duales correspondantes pour chacune des contraintes du primal. Le modèle primal augmenté de ces contraintes comporte $4m^3-8m^2+6m+\deg(\text{dépôt})$ variables et $2m^2+m+1+\deg(\text{dépôt})+n = 2m^2+m+n+1+\deg(\text{dépôt})$ contraintes[3].

La fonction objectif est :

$$\min_{y_{ij}^k} \sum_{k=1}^{m} \sum_{(i,j)\in\overline{A}_{sp}} (m-k+1)\cdot \bar{c}(i,j)\cdot y_{ij}^k$$

Toutes les variables y_{ij}^k sont binaires.

Maintenant nous sommes en mesure d'écrire le dual dans le tableau 6.2. Il comporte $2m^2+m+n+1+\deg(\text{dépôt})$ variables et $4m^3-8m^2+6m+\deg(\text{dépôt})$ contraintes.

La fonction objectif est :

$$\max \sum_{i\in\overline{V}} \lambda_i + \sum_{k=1}^{m+1} \mu^k + \sum_{v\in V} \lfloor \frac{\deg(v)}{2} \rfloor \delta_v$$

Les variables λ_i, μ^k et γ_i^k sont libres alors que $\delta_v \leqslant 0$.

Le tableau 6.3 reprend une solution trouvée par SoPlex pour le graphe de l'exemple introductif de la section 1.3 page 8.

3. Il n'y a toutefois aucune contrainte *vertex* pour un sommet de degré 1, de sorte qu'il serait plus exact d'écrire que nous ajoutons autant de contraintes *vertex* qu'il y a de sommets de degré $\geqslant 2$ (et non n).

Nombres	Contraintes	duales
$2m$	$\forall\ i \in \overline{V}: \quad y^1_{si} + \sum_{k=2}^{m} \left(\sum_{(i,q)\in\overline{A}_{sp}} y^k_{iq} + \sum_{(p,i)\in\overline{A}_{sp}} y^k_{pi} \right) + y^{m+1}_{it} = 1$	λ_i
1	$\sum_{i\in\overline{V}:*i=\mathrm{A}} y^1_{si} = 1$	μ^1
$m-1$	$k = 2\ldots m: \quad \sum_{(i,j)\in\overline{A}_{sp}} y^k_{ij} = 1$	μ^k
1	$\sum_{i\in\overline{V}} y^{m+1}_{it} = 1$	μ^{m+1}
$\deg(\mathrm{A})$	$\forall\ i \in \overline{V} : *i = \mathrm{A}\ : \quad y^1_{si} = \sum_{(\bar{\imath},q)\in\overline{A}_{sp}} y^2_{\bar{\imath}q}$	γ^1_i
$2m(m-2)$	$\forall\ i \in \overline{V}, \forall\ k = 2\ldots m-1\ : \quad \sum_{(p,i)\in\overline{A}_{sp}} y^k_{pi} = \sum_{(\bar{\imath},q)\in\overline{A}_{sp}} y^{k+1}_{\bar{\imath}q}$	γ^k_i
$2m$	$\forall\ i \in \overline{V}\ : \sum_{(p,i)\in\overline{A}_{sp}} y^m_{pi} = y^{m+1}_{\bar{\imath}t}$	γ^m_i
n	$\forall\ v \in V$ avec $\deg v \geqslant 2$:	δ_v
	si $\deg(v)$ est impair :	
	$\sum_{k=2}^{m} \sum_{\substack{(i,j)\in\overline{A}_{sp}: \\ *i=v=*j}} y^k_{ij} \leqslant \lfloor \frac{\deg(v)}{2} \rfloor$	
	si $\deg(v)$ est pair :	
	si v est le dépôt : $\sum_{k=2}^{m} \sum_{\substack{(i,j)\in\overline{A}_{sp}: \\ *i=v=*j}} y^k_{ij} + \sum_{\substack{q\in\overline{V}: \\ *q=v}} y^1_{sq} \leqslant \lfloor \frac{\deg(v)}{2} \rfloor$	
	si v n'est pas le dépôt : $\sum_{k=2}^{m} \sum_{\substack{(i,j)\in\overline{A}_{sp}: \\ *i=v=*j}} y^k_{ij} + \sum_{\substack{q\in\overline{V}: \\ *q=v}} y^{m+1}_{qt} \leqslant \lfloor \frac{\deg(v)}{2} \rfloor$	

TABLEAU 6.1 Le modèle L8$^+$ détaillé. A représente le dépôt.

La valeur optimale est bien $25 + 25 + 30 - 23 = 57$.

6.1.4 Le problème du *pricing*

Nous venons de voir le dual du modèle L8$^+$. Il faut bien sûr l'adapter lorsque localement ou globalement nous rajoutons une ou plusieurs contraintes. Dans la section 6.4, nous verrons comment nous passer de cette mise à jour du dual. Pour l'instant, après avoir résolu un PL, nous passons en revue toutes les variables et vérifions pour chacune d'elle si la contrainte duale correspondante est violée ou non. Si cette contrainte est violée, nous sommes en présence d'une variable primale de coût réduit négatif. Nous ne l'introduisons pas forcément dans le PL. Nous collectons toutes ces variables[4] avec un coût réduit négatif et sélectionnons les x premières qui ont les coûts réduits les plus négatifs. D'autres stratégies sont bien sûr possibles et peut-être plus efficaces mais nous nous sommes limités à cette approche et nous verrons que les premiers résultats expérimentaux sont déjà très encourageants.

4. Évidemment, celles-ci ne sont pas créées.

Nombres	Contraintes	primales
$\deg(A)$	$k=1, \forall\ i \in \overline{V}\ : *i = A\ : \lambda_i + \mu^1 + \gamma_i^1 + \underbrace{\delta_A}_{\text{si } \deg(A) \text{ pair, rien sinon}} \leqslant m \cdot \bar{c}(s,i)$	y_{si}^1
	k = 2	
$\deg(A)$	$\forall\ i \in \overline{V}\ : *\bar{\imath} = A, \forall\ j \in \overline{V}\ : (i,j) \in \overline{A}_{sp}$	$y_{i,j}^2$
	$\lambda_i + \lambda_j + \mu^2 + \gamma_j^2 - \gamma_{\bar{\imath}}^1 + \underbrace{\delta_v}_{\text{si } *i = v = *j\text{, rien sinon}} \leqslant (m-1) \cdot \bar{c}(i,j)$	
$4m(m-1)$	$\forall\ i \in \overline{V}\ : *\bar{\imath} \neq A, \forall\ j \in \overline{V}\ : (i,j) \in \overline{A}_{sp}$	$y_{i,j}^2$
$-\deg(A)$	$\lambda_i + \lambda_j + \mu^2 + \gamma_j^2 + \underbrace{\delta_v}_{\text{si } *i = v = *j\text{, rien sinon}} \leqslant (m-1) \cdot \bar{c}(i,j)$	
$4m(m-1)$	$\forall\ k = 3 \ldots m, \forall\ (i,j) \in \overline{A}_{sp}$	$y_{i,j}^k$
$\cdot(m-2)$	$\lambda_i + \lambda_j + \mu^k + \gamma_j^k - \gamma_{\bar{\imath}}^{k-1} + \underbrace{\delta_v}_{\text{si } *i = v = *j\text{, rien sinon}} \leqslant (m-k+1) \cdot \bar{c}(i,j)$	
	$k = m+1$	
$\deg(A)$	$\forall\ i \in \overline{V}\ : *i = A\ : \lambda_i + \mu^{m+1} - \gamma_{\bar{\imath}}^m \leqslant 0$	$y_{i,t}^{m+1}$
$2m - \deg(A)$	$\forall\ i \in \overline{V}\ : *i \neq A\ : \lambda_i + \mu^{m+1} - \gamma_{\bar{\imath}}^m + \underbrace{\delta_{*i}}_{\text{si } \deg(*i) \text{ pair, rien sinon}} \leqslant 0$	$y_{i,t}^{m+1}$

TABLEAU 6.2 Le dual du modèle L8$^+$ détaillé. A représente le dépôt.

Variables	Valeurs	Obj
λ_{B_3}	24.999999999999	(obj:1)
λ_{C_3}	24.999999999998	(obj:1)
μ^1	30.000000000001	(obj:1)
μ^2	-22.999999999998	(obj:1)
$\gamma_{A_1}^1$	-27	(obj:0)
$\gamma_{A_2}^1$	-27.000000000001	(obj:0)
$\gamma_{B_3}^1$	23.999999999999	(obj:0)
$\gamma_{C_3}^2$	23.999999999998	(obj:0)
$\gamma_{B_3}^3$	25.000000000002	(obj:0)
$\gamma_{C_3}^3$	25.000000000002	(obj:0)

TABLEAU 6.3 Une solution optimale du dual du modèle L8$^+$ pour le graphe introductif de la section 1.3 page 8.

6.2 Les inégalités de flots

Nous présentons cinq familles d'inégalités valides dont quatre ont été généralisées. Mentionnons aussi qu'une famille d'inégalités (les inégalités *no same fruit salad*) ainsi que sa généralisation n'ont pas été implémentées faute d'avoir trouvé un algorithme de séparation

efficace. Nous les mentionnons néanmoins ici car nos premiers tests à la main nous ont démontré leur grande utilité.

6.2.1 Les inégalités *no joint flow*

Pour toute arête $(i,j) \in \overline{E}_r$ et pour tout sommet $v \in \overline{V}$ extérieur, s'il existe un flot entre chacun des sommets i et j de l'arête et le sommet v nous avons une impossibilité. Prenons le cas où un flot relie j et v à la période l et où un autre flot relie v et i à la période k comme sur la figure 6.1. Nous avons alors pour le sommet i[5] l'inégalité valide suivante :

$$y^k_{v,i} + y^l_{j,v} + \sum_{p=1}^{m} \sum_{(i,u)\in\overline{A}_{sp}} y^p_{iu} + \sum_{\substack{p=1 \\ p\neq l-1}}^{m} \sum_{(u,i)\in\overline{A}_{sp}:u\neq v} y^p_{u,i} \leq 1 \tag{6.1}$$

qui exprime que soit en k on entre en i à partir de v ($y^k_{v,i} = 1$), soit on sort en l de j pour aller en v ($y^l_{j,v} = 1$) soit on ne fait ni l'un ni l'autre. Une seule de ces trois possibilités est possible et donc la somme des possibilités est plus petite ou égale à 1.

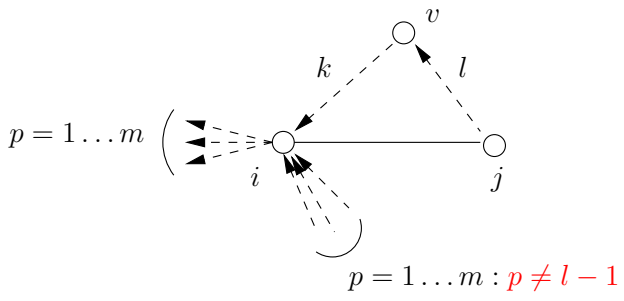


FIGURE 6.1 Les inégalités *no joint flow* (6.1) sont illustrées pour le sommet i.

6.2.2 Les inégalités *generalized no joint flow*

Nous généralisons les inégalités *no joint flow inequalities* (6.1) de la manière suivante. Soit une arête (i,j) de $\overline{E}_r$ et un sommet v de $\overline{V}$ différent de i et j. La figure 6.2 permet de suivre notre propos.

Introduisons de nouvelles notations. Soit $T \subset \{1,\ldots,m+1\}$ un ensemble d'indices de temps. Comme nous avons besoin des mêmes indices mais avec un certain décalage, définissons l'ensemble d'indices de temps :

5. Le cas du sommet j est similaire.

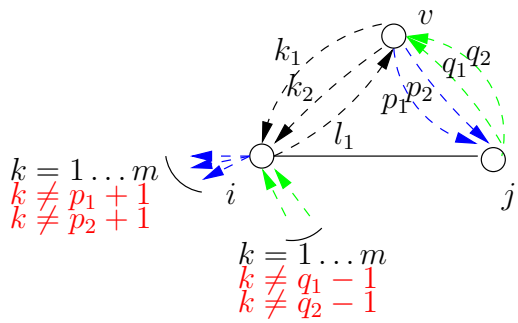


FIGURE 6.2 Les inégalités *generalized no joint flow* (6.2) sont illustrées pour le sommet i.

$$T^{+t} = \{k \in \{1, \dots, m+1\} : k - t \in T\}.$$

De même, définissons

$$T^{-t} = \{k \in \{1, \dots, m+1\} : k + t \in T\}.$$

Par exemple, si $T = \{k_1, k_2\}$ alors $T^{+1} = \{k_1 + 1, k_2 + 1\}$ et $T^{-1} = \{k_1 - 1, k_2 - 1\}$ pour autant que ces nombres sont compris entre 1 et $m+1$. Nous indiçons les différents ensembles d'indices de temps correspondants à différentes paires ordonnées pour les distinguer entre eux. Nous sommes maintenant en mesure de donner les inégalités *generalized no joint flow* pour le sommet i[6] :

$$\underbrace{\sum_{k \in T_{iv}} y_{iv}^k + \sum_{k \in T_{vi}} y_{vi}^k}_{(*)} + \underbrace{\sum_{k \in T_{vj}} y_{vj}^k + \sum_{k \in T_{jv}} y_{jv}^k}_{(\square)} + \underbrace{\sum_{k \in \{1,\dots,m\} \setminus T_{jv}^{-1}} \sum_{(u,i) \in \overline{A}_{sp} : u \neq v} y_{ui}^k + \sum_{k \in \{2,\dots,m+1\} \setminus T_{vj}^{+1}} \sum_{(i,u) \in \overline{A}_{sp} : u \neq v} y_{iu}^k}_{(\triangle)} \leq 1 \tag{6.2}$$

$(*)$ dénote tout ce qui se passe entre les sommets i et v. $(\square)$ est l'équivalent entre j et v et $(\triangle)$ représente tout ce qui entre et sort de i excepté pour le sommet v et qui est incompatible avec $(*)$ et $(\square)$. Comme $(*)$, $(\square)$ et $(\triangle)$ sont mutuellement incompatibles deux à deux, nous avons $(*) + (\square) + (\triangle) \leq 1$.

6. ici aussi, le cas du sommet j est similaire.

Nous avons exclus les cas où $v = s$ et $v = t$. Le premier cas est tout simplement impossible quant au deuxième il n'est d'aucune utilité.

6.2.3 Les inégalités *one-way*

Pour toute paire de sommets $(i,j) \in \overline{E}_r$ et toute paire de sommets $(p,q) \notin \overline{E}_r$ nous avons

$$\begin{aligned} \sum_{k=2}^{m} \left(y_{pi}^k + y_{pq}^k + y_{qp}^k + y_{qj}^k\right) \leq 1 \\ \sum_{k=2}^{m} \left(y_{ip}^k + y_{pq}^k + y_{qp}^k + y_{jq}^k\right) \leq 1 \end{aligned} \tag{6.3}$$

(6.3) renforce (mais ne suffit pas à imposer) l'exigence qu'une arête $(i,j) \in \overline{E}_r$ soit parcourue dans un seul sens. La figure 6.3 illustre cette inégalité.

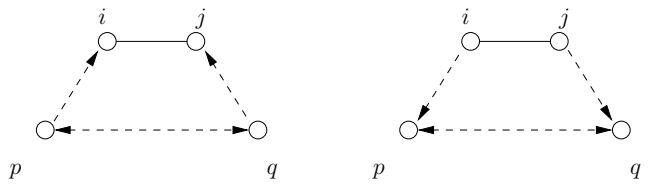


FIGURE 6.3 *Les inégalités* one way (6.3) *sont illustrées : il faut passer dans un sens ou dans l'autre.*

On peut adapter cette inégalité lorsqu'un des sommets p ou q est le sommet fictif s ou le sommet fictif t.

6.2.4 Les inégalités *generalized one-way*

Voici une façon de généraliser les inégalités *one-way inequalities* (6.3). La figure 6.4 permet de suivre notre raisonnement. Nous collectons toutes les variables entre d'une part les sommets i et p et d'autre part les sommets j et q qui soient mutuellement exclusives et imposons cette relation d'exclusivité sur les variables reliant les quatre sommets.

Soient $T'_{ip} \subseteq T_{ip}$ quelques indices de temps pour des variables de flot non nulles entre les sommets i et p, de même pour $T'_{pi} \subseteq T_{pi}$, $T'_{jq} \subseteq T_{jq}$, $T'_{qj} \subseteq T_{qj}$, $T'_{qi} \subseteq T_{qi}$, $T'_{pj} \subseteq T_{pj}$, $T'_{jp} \subseteq T_{iq}$

et $T'_{iq} \subseteq T_{pi}$ entre les sommets respectifs et tels qu'une seule variable de flot puisse être non nulle :

$$T'^{+1}_{pi} \cap T'_{jq} = \emptyset \text{ et } T'^{+1}_{qi} \cap T'_{jp} = \emptyset$$
$$T'^{+1}_{qj} \cap T'_{ip} = \emptyset \text{ et } T'^{+1}_{pj} \cap T'_{iq} = \emptyset$$

alors nous avons pour toute paire de sommets $(i,j) \in \overline{E}_r$ et toute paire de sommets $(p,q) \notin \overline{E}_r$:

$$\sum_{k \in T'_{ip}} y^k_{ip} + \sum_{k \in T'_{pi}} y^k_{pi} + \sum_{k \in T'_{jp}} y^k_{jp} + \sum_{k \in T'_{pj}} y^k_{pj} + \sum_{k \in T'_{jq}} y^k_{jq} + \sum_{k \in T'_{qj}} y^k_{qj} + \sum_{k \in T'_{iq}} y^k_{iq} + \sum_{k \in T'_{qi}} y^k_{qi} + \sum_{k=1}^{m} \bar{y}^k_{pq} \leq 1 \tag{6.4}$$

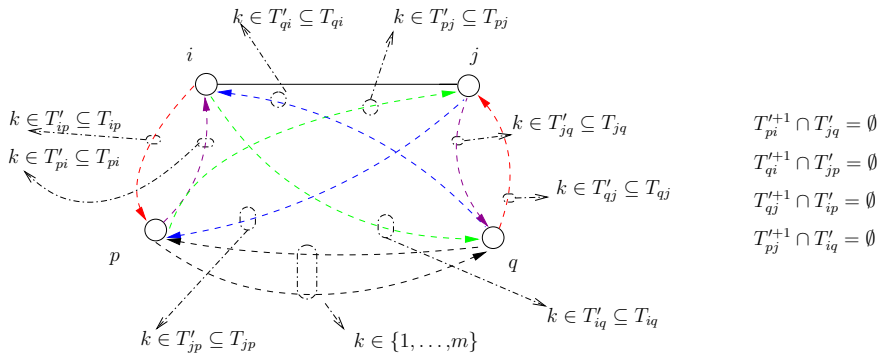


FIGURE 6.4 Les inégalités *generalized one way* (6.4) sont illustrées.

6.2.5 Les inégalités *no bridge flow*

Considérons une situation telle que sur la figure 6.5. Soit une solution relaxée telle que, comme sur illustrée sur la figure, nous ayons $y^p_{u_i,w_i} > 0$ et $y^p_{u_j,w_j} > 0$. Nous écrivons les inégalités pour l'arête $(u_i,\bar{u}_i) \in \overline{E}_r$ mais on pourrait les écrire pour les trois autres arêtes de $\overline{E}_r$.

Sommons toutes les possibilités contradictoires entre l'arête $(u_i,\bar{u}_i)$ et les deux arêtes $(w_j,\bar{w}_j)$ et $(u_j,\bar{u}_j)$ sachant que $y^p_{u_i,w_i} > 0$ et $y^p_{u_j,w_j} > 0$:

$$y^p_{u_i,w_i} + y^p_{u_j,w_j} + \sum_{k=2}^{m} y_{\{u_i,\bar{u}_i\},\{u_j,w_j,\bar{w}_j\}} + \sum_{\substack{k=2\\k\neq p-1}}^{m} y_{\{u_i,\bar{u}_i\},\{\bar{u}_j\}} \leq 1 \tag{6.5}$$

Comme toutes ces possibilités sont contradictoires deux à deux, il faut nécessairement que la somme soit ≤ 1.

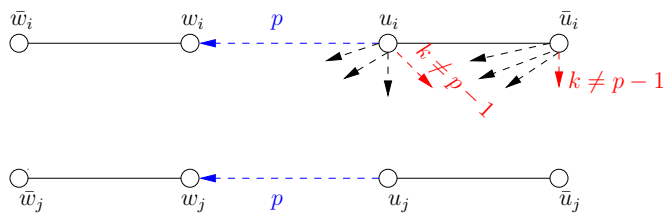


FIGURE 6.5 Les inégalités *no bridge flow inequalities* (6.5) sont illustrées pour l'arête $(u_i,\bar{u}_i)$.

6.2.6 Les inégalités *generalized no bridge flow*

Nous avons généralisé de plusieurs façons différentes les inégalités *no bridge flow inequalities* (6.5).

Une **première version** consiste à rajouter les variables de flot y^p_{ab} au temps p pour maintenir une somme plus petite ou égale à 1. Le figure 6.6 illustre notre propos. Soit $\Sigma = y^p_{u_i,w_i} + y^p_{u_j,w_j} + \sum_{k=2}^{m} y_{\{u_i,\bar{u}_i\},\{u_j,w_j,\bar{w}_j\}} + \sum_{\substack{k=2\\k\neq p-1}}^{m} y_{\{u_i,\bar{u}_i\},\{\bar{u}_j\}}$ le membre de gauche de l'inégalité (6.5) et soit $S \subseteq \overline{V}$ un ensemble de sommets ne comprenant pas les sommets $\{u_i,\bar{u}_i,u_j,\bar{u}_j,w_j,\bar{w}_j\}$, alors nous avons :

$$y^p_{u_j,\bar{w}_j} + y^p_{u_i,S\setminus\{w_i\}} + y^p_{\bar{u}_i,S} + \Sigma \leq 1 \tag{6.6}$$

Une **deuxième version** est obtenue en modifiant légèrement la première version : nous retirons les variables $y^{p-1}_{u_i,u_j}$ et $y^{p-1}_{\bar{u}_i,u_j}$ du membre de gauche de (6.6) pour pouvoir rajouter les variables $y^p_{\bar{u}_j,w_j}$ et $y^p_{\bar{u}_j,\bar{w}_j}$. La figure 6.7 illustre ici aussi notre propos. Soit $\Theta = y^p_{u_j,\bar{w}_j} + y^p_{u_i,S\setminus\{w_i\}} + y^p_{\bar{u}_i,S} + \Sigma$ le membre de gauche de (6.6), nous avons alors :

$$y^p_{\bar{u}_j,w_j} + y^p_{\bar{u}_j,\bar{w}_j} - y^{p-1}_{u_i,u_j} - y^{p-1}_{\bar{u}_i,u_j} + \Theta \leq 1 \tag{6.7}$$

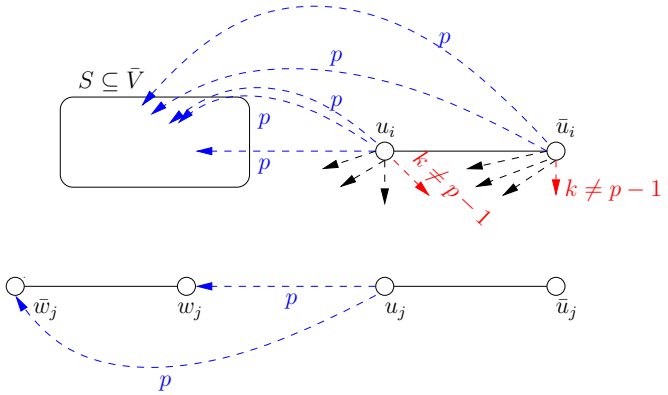


FIGURE 6.6 La première version des inégalités *generalized no bridge flow* (6.6) est illustrée pour l'arête $(u_i,\bar{u}_i)$.

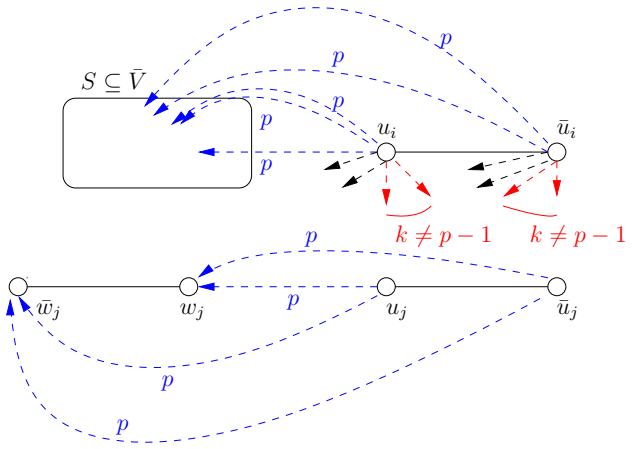


FIGURE 6.7 La deuxième version des inégalités *generalized no bridge flow* (6.7) est illustrée pour l'arête $(u_i,\bar{u}_i)$.

Si on se concentre sur le seul sommet u_i, on obtient une **troisième version**. La figure 6.8 permet de suivre notre propos. Nous gardons les variables $y^p_{u_i,S}$ et $y^p_{u_j,w_j}$ et rajoutons toutes les possibilités incompatibles entre le sommet u_i et les sommets $\{u_j,\bar{u}_j,w_j,\bar{w}_j\}$ et nous obtenons :

$$y^p_{u_i,S} + y^p_{u_j,w_j} + y^{2..m}_{u_i,\{u_j,\bar{u}_j,w_j,\bar{w}_j\}} + y^{2..m}_{\{u_j,\bar{u}_j,w_j,\bar{w}_j\},u_i} - y^{p-1}_{u_i,\bar{u}_j} - y^{p+1}_{\bar{w}_j,u_i} \leq 1 \tag{6.8}$$

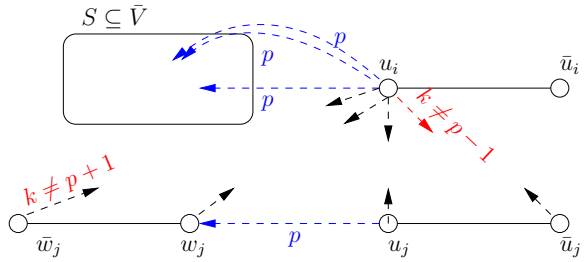


FIGURE 6.8 La troisième version des inégalités *generalized no bridge flow inequalities* (6.8) est illustrée pour le sommet u_i.

6.2.7 Les inégalités *no same fruit salad**

Ces inégalités ainsi que leur généralisation n'ont pas été implémentées parce que nous n'avons pas encore trouvé d'algorithme de séparation véritablement efficace. Nous les présentons néanmoins ici car nos premiers essais de séparation à la main montrent que ces inégalités sont importantes.

Prenons une situation telle que celle dépeinte sur la figure 6.9 et définissons par diff_max(T_1,T_2) la plus grande différence entre des indices de temps de deux ensembles T_1 et T_2:

$$\text{diff\_max}(T_1,T_2) = \max\{|t_1 - t_2| : t_1 \in T_1, t_2 \in T_2\}.$$

Concentrons-nous sur les variables de flot entre les quatre sommets u_i,u_j,w_i,w_j. Soient $T_1 \subseteq T_{u_i,w_i} \cup T_{w_i,u_i}$, $T_2 \subseteq T_{u_i,u_j} \cup T_{u_j,u_i}$, $T_3 \subseteq T_{u_j,w_j} \cup T_{w_j,u_j}$, $T_4 \subseteq T_{w_i,w_j} \cup T_{w_j,w_i}$, $T_5 \subseteq T_{u_i,w_j} \cup T_{w_j,u_i}$ et $T_6 \subseteq T_{w_i,u_j} \cup T_{u_j,w_i}$ tels que diff_max$(T_1,T_3) \leq 1$, diff_max$(T_2,T_4) \leq 1$ et diff_max$(T_5,T_6) \leq 1$, c'est-à-dire que les variables de flot correspondantes s'excluent deux à deux pour les ensembles T_1 et T_3 d'une part, les ensembles T_2 et T_4 d'autre part et les ensemble T_5 et T_6 d'encore une autre part, alors on a :

$$\sum_{k\in T_1} \bar{y}^k_{u_i,w_i} + \sum_{k\in T_3} \bar{y}^k_{u_j,w_j} + \sum_{k\in T_2} \bar{y}^k_{u_i,u_j} + \sum_{k\in T_4} \bar{y}^k_{w_i,w_j} + \sum_{k\in T_5} \bar{y}^k_{u_i,w_j} + \sum_{k\in T_6} \bar{y}^k_{u_j,w_i} \leq 1 \tag{6.9}$$

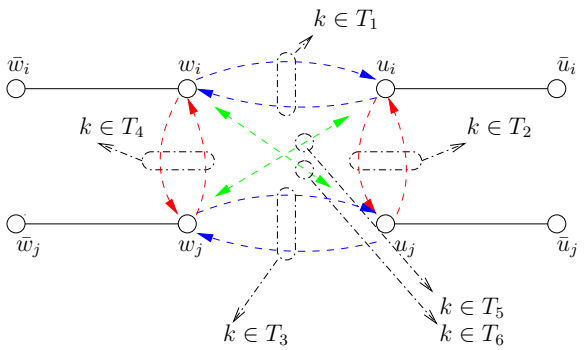


FIGURE 6.9 Les inégalités *no same fruit salad* (6.9) sont illustrées.

6.2.8 Les inégalités *generalized no same fruit salad**

Une façon de faire pour généraliser les inégalités *no same fruit salad* est d'exclure totalement les variables correspondantes à un des deux ensembles T_1 ou T_3, T_2 ou T_4 ou encore T_5 ou T_6, et de prendre toutes les variables correspondantes à l'ensemble conservé.

Par exemple, sur la figure 6.10, nous avons retiré les variables correspondantes à T_6 et T_4 et nous obtenons

$$\sum_{k\in T_1} \bar{y}^k_{u_i,w_i} + \sum_{k\in T_3} \bar{y}^k_{u_j,w_j} + \sum_{k=2}^{m} \left(\bar{y}^k_{u_j,w_j} + \bar{y}^k_{u_i,u_j} \right) \leq 1 \tag{6.10}$$

6.2.9 Les inégalités *no-loop*

Pour mieux comprendre cette inégalité, définissons un *p-circuit*. Soit une solution relaxée pour le modèle LAJM1. Un *p-circuit* est un ensemble alterné d'arcs (i,j) représentants des flots sur des plus courts chemins ($\exists\ k : y^k_{ij} > 0$) et d'arêtes du graphe original parcourus pour diverses périodes k se succédant et qui se recoupe en au moins un sommet des arêtes du graphe original et qui commence et se termine par un plus court chemin. La figure 6.11 montre le p-circuit 9 - 1 - 2 - 8 - 7 - 3 - 4 - 5 - 6 -8 - 7 - 10.

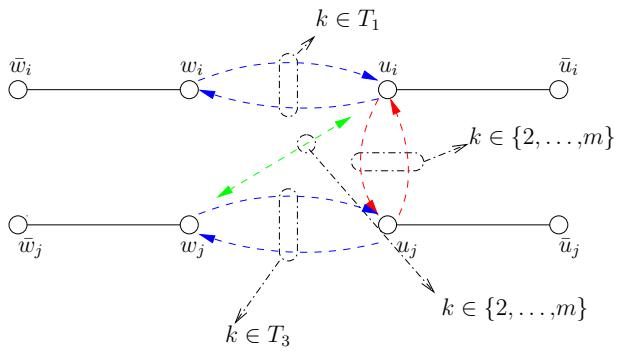


FIGURE 6.10 Les inégalités *generalized no same fruit salad* (6.10) sont illustrées.

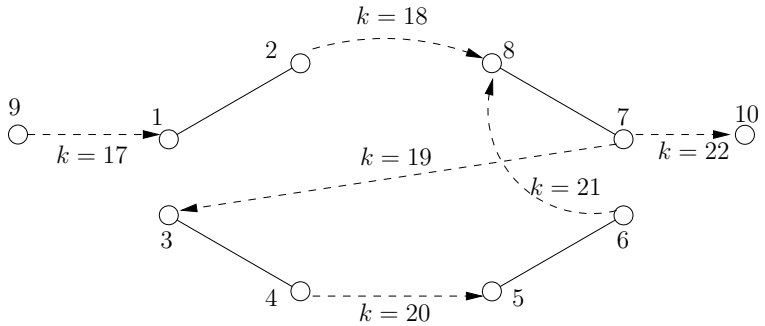


FIGURE 6.11 Un p-circuit composé de 6 arc de plus court chemin et de 4 arête originales.

Voici l'idée de ces inégalités. Si la solution relaxée comporte un p-circuit, il y a un décalage entre le flot entrant et le flot sortant. La figure 6.12 illustre cette idée. Le flot entrant (y^k) ne peut pas correspondre au flot sortant (y^{k+5}) si la solution parcours seulement les 4 arêtes sans sortir de l'ensemble S.

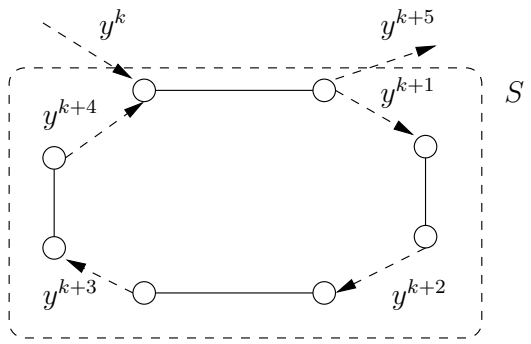


FIGURE 6.12 L'idée des inégalités *no-loop* (6.11) est illustrée : dans un p-circuit, il y a un décalage entre le flot entrant et le flot sortant.

Considérons un ensemble de sommets $S \subsetneqq \overline{V}$ d'arêtes originales. Nous avons donc[7] $\sigma(S) = \frac{|S|}{2}$. Dans la figure 6.12, $\sigma(S) = 4$. Soient T_1 et T_2 deux ensembles disjoints d'indices de temps ($T_1, T_2 \subset \{1, 2, \ldots, m+1\}$) tels que $\text{diff\_min}(T_1, T_2) > \sigma(S)$. Nous avons alors l'inégalité valide suivante :

$$\sum_{t \in T_1} \sum_{\substack{(i,p) \in \bar{E} \\ i \notin S, p \in S}} y^t_{ip} + \sum_{t \in T_2} \sum_{\substack{(q,j) \in \bar{E} \\ q \in S, j \notin S}} y^t_{qj} + y(\gamma(S)) \leq \sigma(S). \tag{6.11}$$

Illustration de la coupe

La figure 6.13 montre un p-circuit dans une solution relaxée. Toutes les variables y illustrées sont à 0,5. Le p-circuit est $10 - 3 - 4 - 5 - 6 - 7 - 8 - 2 - 1 - 3 - 4 - 11$. Le flot entrant l'est au temps 6 et le sortant au temps 11. Sans sortir de l'ensemble S, c'est impossible car celui-ci ne comprend que 4 arêtes.

Nous voyons que $\sigma(S) = 4$, $y(\delta(S)) = 2$ et $y(\gamma(S)) = 3$. La coupe

$$y^6_{10,3} + y^{11}_{4,11} + y^{13}_{5,12} + y(\gamma(S)) \leq 4$$

7. Rappelons que $\sigma(S)$ représente le nombre d'arêtes originelles sous-tendues par les sommets de l'ensemble S.

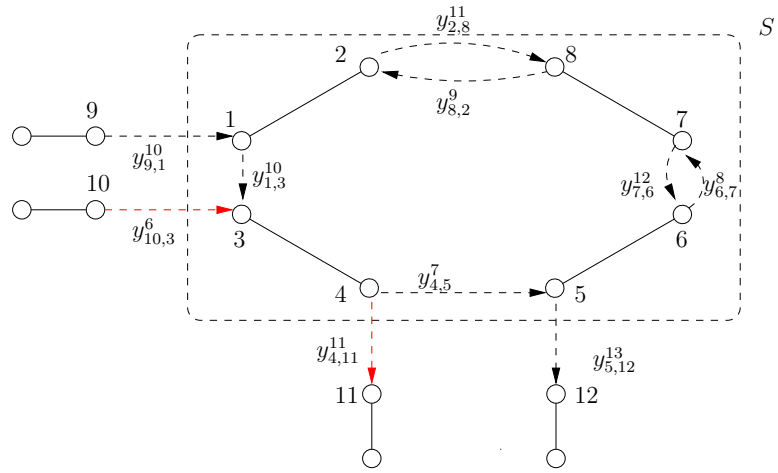


FIGURE 6.13 Une illustration des inégalités *no loop* (6.11).

avec $T_1 = \{6\}$ et $T_2 = \{11,13\}$ est donc violée.

Note sur la séparation

L'énoncé des inégalités (6.11) peut sembler plus général qu'il n'est en réalité. Même si nous considérons un ensemble quelconque de sommets $S \subsetneq \overline{V}$ d'arêtes originales, nous ne pouvons trouver une coupe violée que si nous sommes en présence d'un p-circuit. Observons d'une part que

$$\max_{T_1,T_2} \left(\sum_{t \in T_1} \sum_{\substack{(i,p) \in \bar{E} \\ i \notin S, p \in S}} y^t_{ip} + \sum_{t \in T_2} \sum_{\substack{(q,j) \in \bar{E} \\ q \in S, j \notin S}} y^t_{qj} \right) \geq \frac{y(\delta(S))}{2}.$$

Il suffit de pendre $T_1 = \{1,2,\ldots,m+1\}$ et $T_2 = \emptyset$. D'autre part, en reprenant le lemme (5.1), nous avons que $\sigma(S) - y(\gamma(S)) = \frac{|S|}{2}$. Pour que la coupe (6.11) soit violée, il suffit donc de trouver T_1 et T_2 tels que $\text{diff\_min}(T_1,T_2) > \sigma(S)$ et

$$\sum_{t \in T_1} \sum_{\substack{(i,p) \in \bar{E} \\ i \notin S, p \in S}} y^t_{ip} + \sum_{t \in T_2} \sum_{\substack{(q,j) \in \bar{E} \\ q \in S, j \notin S}} y^t_{qj} > \frac{y(\delta(S))}{2}. \tag{6.12}$$

Sans la présence d'un p-circuit, c'est impossible comme le suggère la figure 6.14. Si nous choisissons un indice rentrant, nous ne pouvons pas sélectionner l'indice sortant correspondant et vice-versa. Il suffit donc de collecter tous les p-circuits différents d'une solution relaxée et de tester si la contrainte (6.12) est respectée pour deux ensembles T_1 et T_2 tels

que diff_min(T_1,T_2) $> \sigma(S)$.

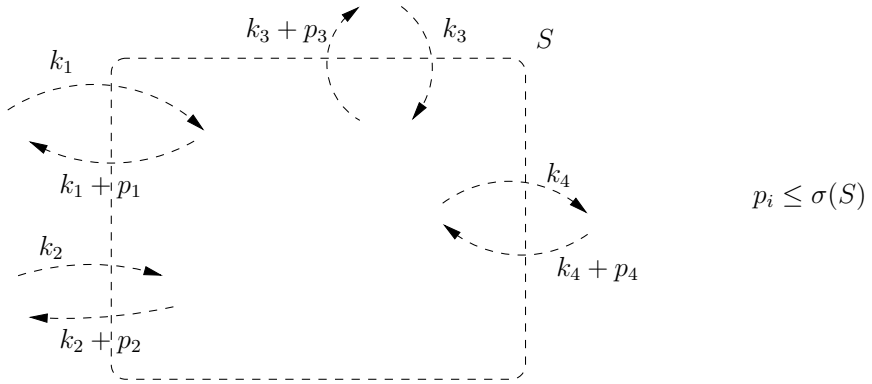


FIGURE 6.14 La présence d'un p-circuit est requise pour trouver une coupe *no loop* (6.11) violée.

6.3 Résultats expérimentaux : BPC1

Dans cette section, nous testons l'approche du *Branch, Price and Cut* avec le modèle L8$^+$. L'ajout des inégalités *vertex* au modèle L8 tel que spécifié à la section 6.1.2 permet d'obtenir une meilleure borne inférieure comme le montre le tableau 6.4 pour des temps de résolution relativement similaires.

TABLEAU 6.4 Comparaisons des écarts relatifs (en %) entre la valeur optimale du PPCC et la valeur optimale du premier PL obtenus avec les modèles L8 et L8$^+$.

Modèles	*g6*	*g7*	*g8*	*g9*	*g10*	*g11*	*g12*	*g13*	*g14*	*g15*
L8	4,09	3,02	3,52	4,09	3,16	0,97	1,55	1,36	1,04	0,43
L8$^+$	1.46	0	0,86	2,87	2,33	0,49	1,55	0,41	0,51	0,17

Nous testons tout ce que nous avons vu dans ce chapitre et le précédent. Nous appelons cet algorithme BPC1. A la section suivante, nous verrons une amélioration de cet algorithme et nous testerons cette nouvelle version, BPC2, par après.

6.3.1 Les pré-traitements sur les variables

Tester les pré-traitements s'avère cette fois-ci bien plus délicat que dans le chapitre précédent. Nos premiers tests nous ont montré clairement que certaines variables bien qu'inutiles dans une solution optimale, nous permettent néanmoins d'obtenir des raccourcis dans

la méthode du simplex et d'éviter certains pivots avec comme résultat un gain parfois significatif de celle-ci. Il nous faut donc tester ces pré-traitements individuellement mais nous devons faire face à plusieurs difficultés.

Tout d'abord, il n'est plus possible de tester tous les paramètres de manière indépendante puisque le *Branch, Price and Cut* dépend directement de certains d'entre-eux et il se pourrait que l'efficacité de certains pré-traitements en soit affectée. Ensuite, certaines variables peuvent être éliminées par plusieurs pré-traitements. Comment distinguer l'effet de l'un ou de l'autre ? Enfin, nos premiers tests nous ont montré que l'effet des pré-traitements était différent suivant l'ordre de grandeur des graphes testés. Alors que les petits graphes bénéficient de l'utilisation de tous les pré-traitements, les moyens et grands graphes sont résolus plus rapidement sans pré-traitement du tout plutôt qu'avec tous les pré-traitements utilisés simultanément.

Pour trancher cette question de l'œuf ou de la poule, nous utilisons la méthodologie suivante. Tout d'abord, nous adoptons un jeu de paramètres dont nous avons pu observé l'efficacité. Notre expérience nous a démontré[8] que rajouter 2000 variables avec les coûts réduits les plus négatifs à la fois semblait être un bon scénario. Pour être aussi proche que possible des conditions dans lesquelles l'algorithme évoluera, nous utilisons aussi toutes les coupes violées que nous détectons, aussi bien celles vues dans le chapitre précédent que celles vues dans celui-ci. Pour l'instant, nous les ajoutons toutes sans exception. Pour tester si un pré-traitement est contre-productif, nous le testons individuellement sans la présence des autres. Le tableau 6.5 présente les résultats obtenus avec tous les pré-traitements ensemble, aucun pré-traitement et l'utilisation de chacun des pré-traitement pris un à un pour cinq graphes moyens (*g8* et *g13*) à grands (*g9*, *g10* et *g14*) de référence. La dernière ligne présente les résultats pour la stratégie que nous avons finalement adoptée. Celle-ci consiste en l'application des pré-traitements *mandatory edge* et *forward compatibility*.

Pour chacun des cinq graphes de référence et pour chaque stratégie de pré-traitement proposée, le tableau 6.5 présente 3 nombres. Le premier (t) est le temps de résolution en secondes, le deuxième (u% pour utilisées) est le pourcentages de variables effectivement utilisées tout au long de la recherche et finalement le troisième nombre (é% pour éliminées) dénote le pourcentage de variables éliminées par la stratégie de pré-traitement.

8. Voir aussi la section 6.3.2 suivante.

Tableau 6.5 Comparaisons de différentes stratégie de pré-traitement des variables pour un *Branch, Price and Cut* avec le modèle L8$^+$.

Pré-traitement(s)	*g8*			*g9*			*g10*			*g13*			*g14*		
	t	u%	é%	t	u%	é%	t	u%	é%	t	u%	é%	t	u%	é%
Aucun	3,56	19,29	0	129,36	10,76	0	62,99	12,16	0	24,90	13,49	0	114,67	9,03	0
Tous	2,72	17,19	28,14	104,51	10,09	19,22	62,82	10,99	14,95	25,00	13,54	15,00	128,93	8,45	15,47
mandatory edge	3,77	17,44	18,20	124,49	9,92	12,70	61,64	10,90	8,90	23,69	13,03	10,08	108,25	8,20	11,34
sp min nbr of edges	3,05	19,75	6,46	141,36	11,06	3,84	67,18	11,44	4,54	25,26	14,15	2,93	120,42	8,82	2,53
reachable edge	2,81	19,26	7,33	117,40	11,09	5,77	66,73	11,65	3,92	23,81	13,34	3,55	110,07	9,07	2,84
reachable sp	2,88	19,80	7,50	116,03	10,86	5,85	79,40	12,52	4,02	23,03	12,62	3,13	129,85	9,18	3,15
depot	3,20	19,55	0,79	116,25	11,20	0,25	62,81	12,16	0,34	24,34	13,58	0,57	111,54	9,15	0,38
backward compatibility	3,09	18,58	0,47	141,54	11,34	0,09	62,49	12,41	0,07	23,92	13,51	0,03	119,17	9,30	0,03
forward compatibility	3,53	20,06	1,42	125,50	10,94	0,76	65,19	11,69	0,42	21,78	12,67	0,23	123,79	9,27	0,36
Stratégie retenue	2,98	17,85	19,47	92,48	9,73	13,42	64,64	11,84	9,31	22,58	12,65	10,29	110,56	8,20	11,67

La première impression que donne ce tableau est que la situation est plutôt confuse. En effet, nous n'avons pu trouver clairement une stratégie l'emportant sur toutes les autres. Un constat pourtant s'impose : nous obtenons des différences significatives dans les temps de résolution suivant l'application ou non de certains pré-traitements. Les appliquer tous aveuglement peut avoir un effet négatif comme nous l'enseigne le cas du graphe *g14* pour lequel rajouter tous les pré-traitement est contre-productif.

Après bien des tests supplémentaires, nous sommes finalement arrivés à la conclusion que l'application d'une stratégie prudente de pré-traitement était sans doute le meilleur moyen de ne pas subir trop d'effets négatifs tout en profitant dans la plupart des cas du bienfait de ceux-ci. La stratégie retenue est l'application de seulement deux pré-traitements : *mandatory edge* et *forward compatibility*. C'est la stratégie que nous adopterons dans ce chapitre. La dernière ligne du tableau 6.5 nous montre l'effet de cette stratégie sur les cinq graphes de référence. Cette stratégie n'est bien sûr pas la panacée et nous pouvons facilement obtenir des contre-exemples pour lesquels cette stratégie est moins efficace que l'utilisation de tous les pré-traitements ensemble ou de n'utiliser aucun pré-traitement. Une étude plus poussée demanderait une analyse détaillée des algorithmes du simplex utilisés et la maîtrise de ceux-ci, ce qui est en dehors du domaine de cette thèse.

6.3.2 Résolution de la relaxation à la racine

Nous avons vu précédemment à la section 5.8.4 que la résolution des PL, en particulier du premier PL à la racine, était ce qui consommait le plus de temps dans la résolution du

modèle L8. C'est aussi le cas avec le modèle L8$^+$.

Notre approche de *Branch, Price and Cut* nous permet d'accélérer grandement cette résolution. Nous le démontrons pas ici mais lancer notre recherche sans partir avec un jeu de variables intéressantes donne des résultats médiocres. Même si nous utilisons seulement les variables d'une solution optimale comme point de départ, l'algorithme traîne beaucoup à chercher un jeu de variables pour démarrer. Pour obtenir ce jeu de variables de départ, nous utilisons toutes les variables obtenues dans les solutions fournies par notre heuristique que nous lançons cinq fois avec des paramètres différents. Nous obtenons en moyenne $\approx\ 3m$ variables intéressantes dans des temps relativement courts pour commencer notre recherche et nous pouvons éviter dans la plupart des cas à devoir rechercher dès le départ des variables avec des coefficients de farkas négatifs pour compléter notre jeu de variables.

Notre méthode de génération de colonnes consiste à parcourir toutes les variables, détecter si celles-ci ont un coût réduit négatif et prendre les x meilleures d'entre-elles que nous introduisons ensuite dans le problème. Nous réoptimisons le PL ainsi obtenu et recommençons la procédure jusqu'à ce que nous ne puissions plus détecter de telles variables. Ce nombre de variables sélectionnées à chaque fois a un grand impact dans la vitesse de résolution des PL comme nous le montre le tableau 6.6 pour nos plus grands graphes de référence. La première ligne du tableau donne le temps en secondes requis pour résoudre le premier PL à la racine lorsque nous prenons toutes les variables dans le problème. Nous comparons ces temps avec le temps de résolution de ces mêmes PL mais en appliquant cette fois-ci notre approche de génération de colonnes. Les lignes suivantes correspondent à l'introduction de x variables avec les coûts les plus négatifs à la fois. Le premier nombre est le temps en secondes et le second nombre est le nombre de PL intermédiaires qui ont dû être solutionnés. Les temps indiqués sont les temps totaux non seulement pour calculer les PL intermédiaires mais aussi pour tester les variables et les introduire.

Nous pouvons voir dans le tableau 6.6, que les meilleures valeurs pour x se situent autour des 1500-2000. En fait, il n'existe pas de valeur universelle convenant pour tous les graphes. Nous avons expérimenté plusieurs valeurs proches de ces nombres mais à chaque fois une valeur convenait mieux pour tel ou tel graphe alors qu'une autre valeur convenait mieux à d'autres. La valeur x = 2000 est un bon compromis et dorénavant nous injecterons x = 2000 variables de coûts réduits les plus négatifs à la fois. Dans le tableau 6.6, nous pouvons voir que pour ce paramètre notre algorithme est de 2 à 5 fois plus rapide qu'une

Tableau 6.6 Comparaisons des temps de résolution (en secondes) ainsi que du nombre de PL intermédiaires nécessaires pour résoudre le premier PL à la racine pour le modèle L8$^+$.

	g9		*g10*		*g13*		*g14*		*g15*	
	Résolution avec toutes les variables									
	71,7		114,0		47,8		195,0		708,0	
	Résolution avec x variables									
	t	#PL	t	#PL	t	#PL	t	#PL	t	#PL
x = 500	40,5	28	44,5	33	16,9	22	98,0	38	177,0	54
x = 1000	38,4	19	36,6	20	14,6	15	78,3	24	163,0	34
x = 1500	39,4	15	33,3	16	13,2	12	75,3	18	138,0	24
x = 2000	31,8	12	32,0	15	15,9	11	57,3	15	145,0	21
x = 2500	37,8	12	33,1	12	18,7	10	81,4	13	191,0	19
x = 3000	44,5	10	33,9	11	20,1	10	86,8	13	167,0	17

résolution classique par simplex dual avec toutes les variables.

Qu'en est-il des PL suivants ? Notre expérience nous a montré que dans la plupart des cas, très peu[9] de variables supplémentaires doivent être introduites de sorte que le paramètre x n'a plus tellement d'importance. Nous utiliserons aussi la valeur x = 2000 pour solutionner les PL suivants.

Il y aurait sans doute moyen de solutionner encore plus rapidement les PL intermédiaires en réutilisant l'information obtenue à chaque fois[10] : nous ne sommes pas obligés de revérifier le coût réduit de toutes les variables. Nous pourrions nous contenter de recalculer celui-ci pour les variables dont nous savons qu'elles étaient de coût réduit négatif au tour précédent. Bien sûr, à un moment donné, il faut calculer tous les coûts réduits pour être certain d'être à l'optimum. Nous n'avons pas implémenté cette approche car, pour la taille des graphes envisagés dans cette thèse, le passage en revue de toutes les variables est relativement peu coûteux. Ainsi par exemple, pour le graphe *g15*, le temps nécessaire pour passer en revue toutes les variables les 21 fois est d'environ 1,42 secondes soit en moyenne 0,068 seconde par PL, un temps relativement modeste comparé aux 145 secondes nécessaires à toute l'opération.
Par contre, lorsqu'il s'agit de tester les variables pour des PL qui sont modifiés dynamiquement par l'ajout d'inégalités supplémentaires, cette approche apporterait sans doute un gain non négligeable. En effet, l'introduction de nouvelles inégalités rend le calcul des coûts réduits particulièrement dispendieux par la mise à jour du dual comme nous le verrons plus

9. De l'ordre de 0,001 % à maximum 1 %.
10. Plusieurs stratégies sont proposés et disséquées dans (Applegate *et al.*, 2007).

loin. Cette approche n'a toutefois pas été testée dans cette thèse.

6.3.3 Les inégalités valides

Pour voir l'effet individuel de chacune des nouvelles familles d'inégalités, nous les testons une à une en prenant comme base de notre algorithme tout ce que nous avons vu jusqu'ici avec les inégalités de parité et les inégalités *generalized co-circuit*[11]. Nous prenons à chaque fois toutes les inégalités violées détectées.

Le tableau 6.7 présente ces résultats. Nous commençons par ajouter seulement les inégalités de parité et les inégalités *generalized co-circuit* dans les deux premières lignes. La toute première ligne représente le temps en secondes pour la résolution du problème alors que la deuxième ligne présente d'abord le nombre de nœuds nécessaires à la recherche suivi entre parenthèses du nombre d'inégalité *odd set*, *even-set* et *generalized co-circuit* respectivement. Ainsi, pour le graphe *g9*, la résolution prend 151,23 secondes, nécessite 7 nœuds dans l'arbre de recherche et utilise 39 inégalités *odd-set*, 3 inégalités *even-set* et 5 inégalités *generalized co-circuit*. Les deux prochaines lignes présentent les mêmes données mais cette fois-ci en ajoutant les inégalités *generalized no joint flow*. La première ligne donne le temps de résolution en secondes, la deuxième ligne présente le nombre de nœuds requis lors de la recherche ainsi que, entre parenthèses, le nombre d'inégalités *odd-set*, *even-set*, *generalized co-circuit* et *generalized no joint flow*. Les lignes suivantes reprennent les mêmes données mais cette fois-ci en remplaçant les inégalités *generalized no joint flow* par les inégalités *generalized one way* et ainsi de suite. A chaque fois, nous conservons nos trois familles de base, les inégalités *odd-set*, *even-set* et *generalized co-circuit*, et testons une nouvelle famille d'inégalités séparément.

Nous n'avons mis que les deux premières versions des inégalités *generalized no bridge flow* car la troisième version n'a jamais été détectée pour ces graphes.

Il y a plusieurs enseignements à tirer de ce tableau 6.7. Tout d'abord, l'ajout de toutes les coupes d'une même famille améliore parfois la recherche mais pas toujours. Les coupes *generalized no joint flow* nous en donne un bon exemple. Dans le cas du graphe *g10*, elles nous permettent de faire un gain significatif puisque nous passons d'une résolution de 121,63 secondes à seulement 76.67 secondes mais dans le cas du graphe *g9*, elles font quasiment

11. En effet, notre expérience nous a montré que l'ajout des inégalités *generalized co-circuit* provoque une légère amélioration lorsqu'elles sont combinées avec les nouvelles inégalités vues dans ce chapitre comme nous le verrons par la suite.

TABLEAU 6.7 Comparaisons des temps de résolution (en secondes) ainsi que du nombre de nœuds et du nombre de coupes pour chacune des nouvelles familles de coupes vues dans ce chapitre.

inégalités	*g9*	*g10*	*g13*	*g14*	*g15*
base: parité +*gen. co-circuit*	151,23 7 (39/3/5)	121,63 8 (40/4/6)	23,64 3 (17/1/1)	172,91 5 (73/2/10)	188,53 1 (46/2/2)
gen. no joint flow	255,10 21 (78/3/19/101)	76,67 3 (31/3/4/18)	23,79 3 (17/1/1/0)	179,84 5 (73/2/10/0)	189,42 1 (46/2/2/1)
gen. one way	146,14 11 (46/4/9/34)	138,42 8 (37/3/5/21)	26,69 3 (17/1/1/9)	188,93 6 (73/3/9/30)	196.93 1 (47/1/2/29)
gen. no bridge flow version 1	132,19 7 (38/3/6/17)	126,25 6 (45/5/7/23)	23,95 3 (17/1/1/5)	182,80 6 (72/3/9/21)	192,39 1 (46/2/2/11)
gen. no bridge flow version 2	118,75 7 (38/3/6/16)	121,58 6 (46/5/7/24)	24,07 3 (17/1/1/5)	184,59 6 (72/3/9/21)	192,67 1 (46/2/2/11)
no loop	281,66 8 (68/3/17/285)	68,37 1 (19/2/2/29)	22,86 1 (17/1/1/19)	194,21 4 (69/2/8/66)	186,14 1 (47/1/2/38)

doubler le temps de la recherche ! Ensuite, nous pouvons constater que la version 2 des inégalités *generalized no bridge flow* semble meilleure que la version 1. Finalement, nous nous rendons compte qu'il n'y a pas vraiment de fil conducteur pertinent à l'ajout ou non de certaines coupes. Dans le cas du graphe *g14*, il vaut carrément mieux ne pas introduire de nouvelle famille de coupes.

En fait, il est contre-productif de rajouter toutes les coupes violées détectées. Se pose alors la question de comment faire pour rajouter les coupes ? Lesquelles choisir ? Dans le cadre de cette thèse, nous ne sommes pas parvenus à étudier théoriquement nos différentes familles de coupes. Nous devons donc nous baser sur nos tests expérimentaux pour avoir une idée de leur performance. Plutôt que de développer différentes stratégies comme utiliser une famille avant une autre, utiliser certaines coupes en dernier recours, etc, nous avons développé une façon relativement simple de sélectionner nos coupes.

Nous en détectons un certains nombres et puis nous les trions par intérêt. Le critère retenu est celui d'efficacité, *i.e.* la distance de la coupe à son hyperplan (voir Achterberg, 2008, pages 48-49). Soit une coupe $\underline{\gamma} \leqslant d^T x \leqslant \overline{\gamma}$, l'efficacité e_r est calculée par

$$e_r = \max\{\underline{\gamma} - d^T x, d^T x - \overline{\gamma}\}/||d||.$$

Nous prenons les x premières coupes qui ont la plus grande efficacité. SCIP propose par

défaut une autre mesure qui est une combinaison de l'efficacité d'une coupe, de son orthogonalité par rapport aux autres coupes ainsi que son parallélisme par rapport à la fonction objectif. Cette approche donne de bons résultats dans la thèse de Tobias Achterberg (voir Achterberg, 2008, pp 114-116) mais nous avons dû abandonner cette approche qui ne nous permettait pas de retenir certaines coupes pourtant essentielles dans la recherche et que SCIP éliminait de lui-même. Cette approche simpliste à le mérite de tester toutes les familles de coupes voulues et de choisir parmi celles-ci les meilleures coupes. Nous pouvons nous permettre cette approche car le problème de séparation se résout relativement vite pour toutes les familles de coupes que nous utilisons vu la grandeur relativement modeste des graphes que nous solutionnons.

La tableau 6.8 présente les temps de résolution en secondes lorsqu'on se limite à prendre x coupes par résolution d'un PL. x = 35 présente un bon compromis pour nos graphes de référence.

TABLEAU 6.8 Temps de résolution en secondes en prenant les x coupes les plus efficaces.

nbr de coupes	*g9*	*g10*	*g13*	*g14*	*g15*
x = 20	93,57	60,14	23,71	109.95	227,03
x = 25	91,63	63,71	23,58	195,77	227,08
x = 30	115,96	69,88	23,87	108.30	223,38
x = 35	91,74	70,95	24,27	107,22	193,96
x = 40	103,92	74,37	23,98	129,62	188,78
x = 45	90,56	70,43	23,69	125,61	182,97
x = 50	95,22	68,24	23,82	135,39	219,60
x = 55	101.88	66,14	23,79	118,67	195.95
x = ∞	88,00	66,36	23,54	118,85	185,12

Il y a aussi un autre avantage à limiter le nombre de coupes que nous prenons effectivement dans chacun des PL. Le temps pour vérifier si une contrainte duale est violée est proportionnel au nombre de coupes supplémentaires ajoutées dans un PL. En effet, pour chacune des contraintes primales supplémentaires, nous obtenons une variables duale supplémentaire dans les contraintes duales correspondantes aux variables primales à tester. Vu la taille relativement modeste de nos graphes, nous aurions pu choisir de déterminer une fois pour toute l'appartenance d'une variable duale à une contrainte duale. Dans notre implémentation, cette appartenance est parfois recalculée ou parfois mise en mémoire une fois pour toute.

Pour x = 35, on peut se demander combien de coupes ont été acceptées et combien

ont été rejetées. Le tableau 6.9 répond à cette question. Par graphe, nous avons deux colonnes : la colonne "a" pour les coupes acceptées et la colonne "r" pour les coupes refusées. Ce sont les résultats finaux. Il est bien possible qu'au cours de la recherche une coupe soit rejetée à un stade donné pour être acceptée par après car nous retestons les anciennes coupes précédemment violées avec la solution fractionnaire optimale du nœud courant.

TABLEAU 6.9 Nombre de coupes acceptées ou rejetées en prenant les 35 coupes les plus efficaces par résolution d'un PL.

familles	*g9*		*g10*		*g13*		*g14*		*g15*	
	a	r	a	r	a	r	a	r	a	r
odd set	25	13	16	3	14	3	17	5	26	24
even set	3	0	1	0	0	1	2	0	1	0
gen. co-circuit	4	0	2	0	1	0	1	1	2	0
gen. no joint flow	13	2	2	1	0	0	0	0	0	0
gen. one way	15	3	13	0	9	0	14	0	27	2
gen. no bridge	4	9	7	1	4	2	5	2	3	12
no loop	76	45	27	7	10	9	20	9	11	28

En voyant les piètres résultats des coupes *generalized co-circuit* dans le chapitre précédent, on peut légitiment se demander si elles valent la peine d'être introduites ici. Le tableau 6.10 qui reprend les temps de résolution pour x = 35 avec et sans ces inégalités nous apporte un début de réponse. Si pour les plus petits graphes (*g9*, *g10* et *g13*), ces inégalités sont contre-productives, pour nos grands graphes (*g14* et *15*), elles nous permettent de faire un gain de temps relativement modeste mais néanmoins présent.

TABLEAU 6.10 Temps de résolution en secondes en prenant les 35 coupes les plus efficaces avec et sans inégalités *gen. co-circuit*.

	g9	*g10*	*g13*	*g14*	*g15*
avec *gen. co-circuit*	91,74	70,95	24,27	107,22	193,96
sans *gen. co-circuit*	84,85	64,29	23,04	110,40	197,40

6.3.4 Comparaisons des temps de résolution

La figure 6.15 compare les temps de résolution relatifs pour les mêmes graphes que la figure 5.34 page 143. Comme nous utilisons le modèle L8$^+$ et non plus le modèle L8, nous

utilisons les temps de résolution pris par CPLEX pour résoudre ce modèle L8$^+$ comme temps de référence, *i.e.* les temps pris par CPLEX correspondent à 100 %. Rappelons que des problèmes d'instabilité numérique dans la résolution des PL ne nous permettent plus d'utiliser le solveur CLP.

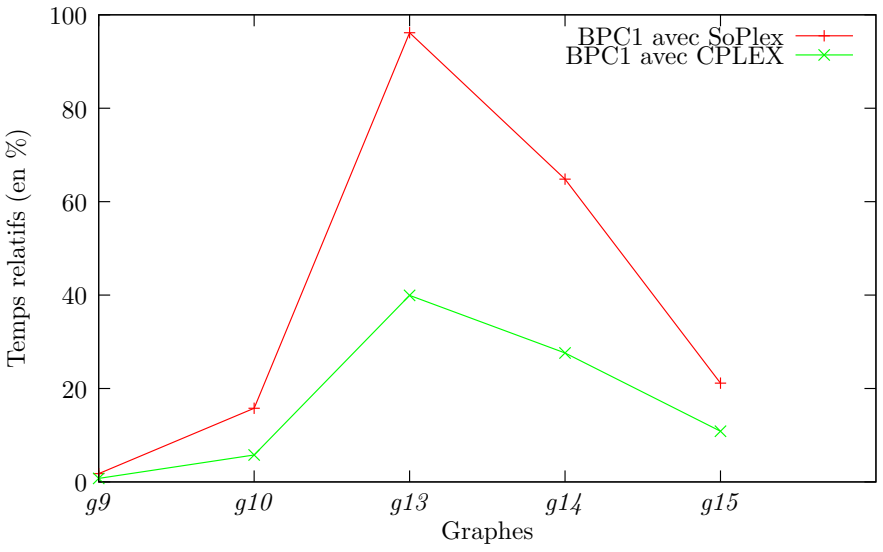


FIGURE 6.15 Comparaison des temps de résolution pour le BPC1. Le temps de référence est le temps pris par CPLEX, *i.e.* pour tous les graphes ce temps vaut 100 %.

Un résultat hors norme saute aux yeux : le cas du graphe *g13*. Il semblerait à première vue que notre nouvelle méthode ne donne pas le même taux d'amélioration que pour les autres graphes. Il n'en est rien comme on peut le voir sur le tableau 6.11 qui reprend les temps en secondes. C'est le solveur CPLEX qui arrive à solutionner ce graphe très rapidement.

Dans le tableau 6.11, nous avons repris les temps de résolution en secondes aussi bien pour le solveur CPLEX que pour notre méthode avec les solveurs SoPlex et CPLEX. Nous avons aussi repris les meilleurs résultats du chapitre précédent pour pouvoir comparer.

Le modèle L8$^+$ ne diffère du modèle L8 que par n contraintes supplémentaires[12] et pourtant quelle différence. CPLEX tire avantageusement parti de cette information supplémentaire comme on peut le voir dans le tableau 6.11.

12. Autant de contraintes supplémentaires qu'il y a de sommets de degré plus grand ou égal à deux plus exactement.

Tableau 6.11 Comparaisons des temps (en secondes) entre les algorithmes BC1 et BPC1 avec différents solveurs ainsi que le solveur CPLEX avec les modèles L8 et L8$^+$.

Méthodes	*g9*	*g10*	*g13*	*g14*	*g15*
CPLEX avec L8	8235,26	782,47	717,74	576,56	4700,50
BC1 avec SoPlex	296,64	174,03	58,33	402,65	794,44
BC1 avec CPLEX	140,44	63,36	20,65	196,81	316,29
CPLEX avec L8$^+$	5152,23	450,34	25,24	165,45	917,79
BPC1 avec SoPlex	91,74	70,95	24,27	107,22	193,96
BPC1 avec CPLEX	38,63	25,97	10,08	45,67	100,20

Nous comparons encore une fois notre algorithme utilisant CPLEX comme solveur de PL avec le solveur CPLEX. Notre algorithme est de 2 (graphe *g13*) à 133 (graphe *g9*) fois plus rapide. Encore une fois, nous possédons une connaissance que CPLEX ne possède pas.

Regardons maintenant comment est réparti le temps nécessaire pour la résolution du problème du postier chinois cumulatif avec notre méthode BPC1. La figure 6.16 nous montre une répartition du temps typique pour nos plus gros graphes de référence. A nouveau, la grande majorité du temps (90,90 % - 176,32 secondes) est consacrée à la résolution des PL (cette fois-ci par une combinaison des méthodes primale (80,16 % - 155,48 secondes) et duale (10,74 % - 20,84 secondes) du simplex puisque nous générons nos variables, et de ce temps, toujours une part très importante (74,76 % - 145,00 secondes) est consacrée à la résolution du premier PL.

Le reste de la machinerie — génération des variables (2,45 % - 4,75 secondes), séparation et création des coupes (0,83 % - 1,61 secondes), l'heuristique (5,58 % - 10,83 secondes) ainsi que le reste (pré-traitements, sélection des nœuds, maintien de l'arbre de recherche, échanges d'information entre SCIP et SoPlex, etc.) (0,23 % - 0,45 secondes) — prend un temps relativement modeste. L'heuristique pour trouver une bonne solution de départ est encore une fois invoquée cinq fois avec des paramètres différents.

Il y a bien sûr des différences entre la répartition du temps pour notre BC1 et notre BPC1. Les plus notables sont les suivantes. D'une part, nous utilisons maintenant la méthode du simplex primale. Celle-ci est plus lente mais est indispensable pour notre approche. D'autre part, nous générons nos variables et il faut donc y dédier du temps. Dans la section suivante, nous examinerons un peu plus en détails ce processus de génération de variables.

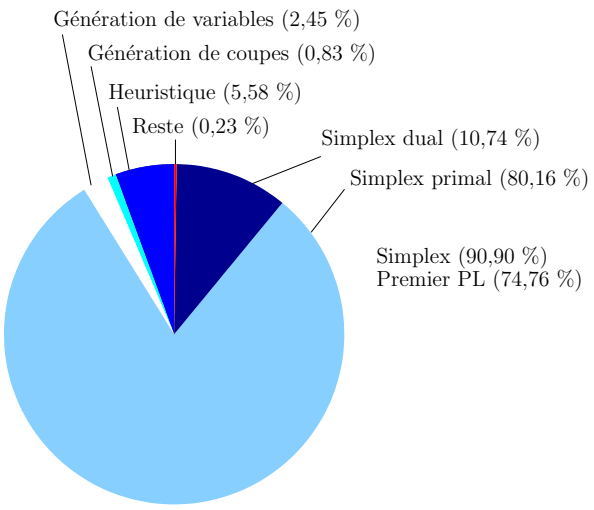


FIGURE 6.16 Répartition du temps pour l'algorithme BPC1 pour le graphe *g15*.

6.4 Évaluation implicite du dual

Pendant la recherche d'une solution optimale dans une approche de *Branch, Price and Cut*, nous rajoutons des coupes de façon dynamique à chaque nœud de l'arbre de recherche. Nous ne pouvons déterminer à l'avance quelles seront ces coupes puisque c'est pendant la recherche que nous les générons. Comme nous testons le coût réduit des variables primales en cherchant les contraintes duales correspondantes violées, nous devons utiliser un mécanisme pour mettre à jour le dual. Nous avons essayé plusieurs approches mais toutes sans exception se sont révélées coûteuses en temps[13]. Le problème est qu'à chaque nœud, nous avons potentiellement un dual relativement différent. Si nous pouvons mettre le dual à jour une fois pour toute lorsque nous introduisons une contrainte supplémentaire, il faut encore à chaque nœud adapter cette mise à jour au véritable dual correspondant à ce nœud. Après avoir testé 6 variantes de la mise à jour du dual, nous avons opté pour une mise-à-jour « mixte » dans notre algorithme BPC1. Pour certaines contraintes, nous faisons cette mise à jour une fois pour toute, pour d'autres nous la calculons explicitement pour chaque variables.

Pour l'instant, le nombre de coupes que nous rajoutons est relativement modeste et nous ne sommes pas encore vraiment confronté à un problème de lenteur rédhibitoire dans la mise à jour du dual. Prenons le cas du graphe *g15*. Pour chaque PL, nous passons à chaque fois en revue les 348 579 variables primales (contraintes duales). Lorsque nous le faisons pour le

13. Nous n'avons nulle part trouvé de documentation ou d'exemples sur la démarche à suivre dans SCIP.

PL de base (*core LP*), le temps nécessaire[14] est de 0,042 seconde. Ce temps est relativement modeste, surtout si on le compare au temps nécessaire à l'ensemble des opérations effectuées dans le *pricer* : la mise à jour des nouvelles variables (création de celles-ci et puis mise à jour des anciennes contraintes), tri et sélection des meilleures variables, échanges d'informations avec SCIP prennent 0,0854352 seconde. Nous consacrons donc 49,16 % du temps à cette mise à jour. Par contre, lorsque nous ajoutons des contraintes (coupes ou contraintes locales de branchement) pendant la recherche, ce temps augmente considérablement. Prenons un nœud où nous avons ajouté 100 contraintes supplémentaires. Le temps pour passer en revue toutes les variables primales (contraintes duales) est maintenant de 0,847583 seconde. Une augmentation de 992,08 % ! Il s'agit pratiquement uniquement d'une augmentation de la mise à jour car le temps global passé dans le *pricer* est maintenant de 0,847628 seconde. Notre processus de mise à jour accapare donc 99.99 % de ce temps !

Plusieurs stratégies sont envisageables pour aborder ce problème de la mise à jour du dual. En voici une : et si nous évitions tout simplement cette mise à jour ? A priori, ce n'est pas possible car il faut explicitement tenir compte des nouvelles contraintes ajoutées. Sauf si ... celles-ci sont écrites de telle façon que nous puissions faire une mise à jour implicite et non plus explicite.

Mettons-nous dans la situation simple de la figure 6.17 qui représente un modèle primal et son modèle dual correspondant. Nous rajoutons une contrainte η dynamiquement pendant la recherche d'une solution optimale. Pour simplifier notre propos, notons les variables primales par y et notons aussi le coût de ces variables par $c(y)$.

Nous avons déjà vu que si la contrainte duale y

$$\sum_j \beta_j \lambda_j + \alpha_i \eta \leqslant c(y) \tag{6.13}$$

n'est pas respectée, la variable primale correspondante y a un coût réduit négatif. Si la contrainte primale η est de la forme

$$\sum_i \alpha_i y_i \leqslant Q, \tag{6.14}$$

14. Nous avons choisi un PL en particulier mais les données sont très semblables pour chacun d'entre-eux.

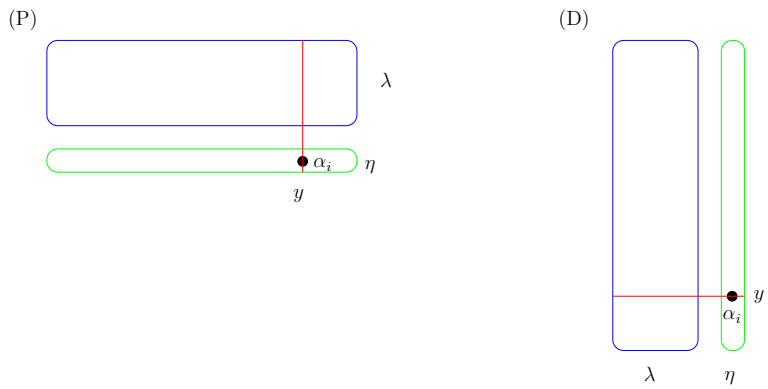


FIGURE 6.17 Mise à jour implicite du dual.

la variable duale η est $\leqslant 0$. Si de plus, $\alpha_i \geqslant 0$, alors $\alpha_i\eta \leqslant 0$ et

$$\sum_j \beta_j\lambda_j + \alpha_i\eta \leqslant \sum_j \beta_j\lambda_j.$$

Au lieu de vérifier si (6.13) est respectée, ce qui demande le calcul de $\alpha_i\eta$, nous nous contentons de vérifier si

$$\sum_j \beta_j\lambda_j \leqslant c(y) \tag{6.15}$$

est respectée ou pas. Si (6.13) n'est pas vérifiée, alors (6.15) n'est pas vérifiée non plus. Autrement dit, nous pouvons vérifier de façon implicite (sans le calcul explicite du terme $\alpha_i\eta$) si (6.13) est vérifiée ou non. Il est évidemment possible que (6.15) ne soit pas vérifiée alors que (6.13) l'est. Dans ce cas, nous générerions une variable primale peut-être inutile. Dans la pratique ce nombre de variables supplémentaires est restreint comme nous le verrons dans la section 6.3 concernant les résultats expérimentaux.

Si certaines inégalités sont déjà naturellement de la forme (6.14), qu'en est-il des autres? Dans le reste de cette section, nous montrons comment transformer certaines inégalités pour leur donner la forme (6.14).

Les inégalités de parité

Nous avons déjà vu dans le tableau 5.4 page 120 comment faire pour les inégalités *odd set* et *even set*.

Les inégalités *generalized co-circuit*

Partons du lemme 20 page 104 :

$$y(\delta(S \setminus F)) + y(F) + 2\ y(\gamma(S)) = |S|. \tag{5.1}$$

Injectons la valeur correspondante de $y(\delta(S \setminus F))$ dans 5.22 pour obtenir :

$$2\,[y(F) + y(\gamma(S))] \leqslant |S| + |F| - 1.$$

La règle de branchement *fix max flow two vertices*

La branche $\sum_k (y_{ij}^k + y_{ji}^k) = 0$ se traduit aisément en

$$\sum_k (y_{ij}^k + y_{ji}^k) \leqslant 0.$$

Quant à la branche $\sum_k (y_{ij}^k + y_{ji}^k) = 1$ qui n'est rien d'autre que $\sum_k (y_{ij}^k + y_{ji}^k) \geqslant 1$, elle peut se mettre sous la forme 6.14 grâce à $\sum_k \sum_{(i,j)\in\overline{A}_{sp}} y_{ij}^k = m + 1$:

$$\sum_k \sum_{(i,j)\in\overline{A}_{sp}} y_{ij}^k - (\sum_k (y_{ij}^k + y_{ji}^k)) \leqslant m.$$

Les inégalités de flots

Elles sont toutes de la forme 6.14 et n'ont donc pas besoin d'être adaptées.

6.5 Résultats expérimentaux : BPC2

Dans cette section, nous reprenons l'algorithme BPC1 mais en faisant une mise à jour implicite et non plus explicite. Le nouvel algorithme ainsi obtenu est appelé BPC2. Nous passons brièvement sur les nouveaux résultats obtenus.

6.5.1 Les pré-traitements sur les variables

L'évaluation implicite du dual nous oblige à générer plus de variables que véritablement nécessaire. Par contre, ces variables supplémentaires nous permettent de réintroduire tous nos pré-traitements sans exception et sans pénalité notable.

6.5.2 Comparaison entre BPC1 et BPC2 pour le nombre de variables générées

La tableau 6.12 présente le pourcentage de variables générées pour 5 de nos graphes de référence lorsque nous utilisons BPC1 et BPC2.

TABLEAU 6.12 Pourcentages des variables générées avec BPC1 et BPC2.

Algoritmes	*g9*	*g10*	*g13*	*g14*	*g15*
BPC1 avec SoPlex	9,63	11,79	12,71	8,22	9,27
BPC2 avec SoPlex	13,12	12,11	14,31	9,16	10,58
BPC1 avec CPLEX	13,17	12,26	17,32	11,67	11,15
BPC2 avec CPLEX	15,69	13,40	16,58	11,96	12,07

A part pour le cas du graphe *g13* avec CPLEX, BPC2 génère plus de variables que BPC1 ce qui est normal vu qu'en évitant une mise à jour explicite coûteuse du dual nous testons une condition qui a plus de chance d'être violée. Comme nous le verrons dans la sous-section suivante, cette abondance de variables permet aussi à la méthode du simplex d'utiliser moins de pivot.

Nous obtenons des résultats similaires pour l'utilisation des inégalités valides ainsi que pour la résolution du premier PL à la racine.

6.5.3 Comparaisons des temps de résolution

Nous utilisons les deux algorithmes BPC1 et BPC2 avec les mêmes jeux de paramètres. Il y a seulement trois différences entre les deux. Tout d'abord, BPC2 utilise une évaluation implicite du dual alors que BPC1 utilise une version explicite du dual pour évaluer les variables primales. Ensuite, BPC2 utilise tous les pré-traitement et enfin BPC2 utilise une écriture différente de certaines contraintes.

Le tableau 6.13 reprend les données du tableau 6.11 en y ajoutant les temps de l'algorithme BPC2,

Nous obtenons donc un gain de temps relativement modeste mais bien réel. Voyons où nous avons amélioré notre algorithme. Le tableau 6.14 compare les temps requis par les divers ingrédients de nos deux algorithmes BCP1 et BCP2. Les quatre premières colonnes reportent

Tableau 6.13 Comparaisons des temps (en secondes) entre les algorithmes BC1, BPC1 et BPC2 avec différents solveurs ainsi que le solveur CPLEX avec les modèles L8 et L8$^+$.

Méthodes	*g9*	*g10*	*g13*	*g14*	*g15*
CPLEX avec L8	8235,26	782,47	717,74	576,56	4700,50
BC1 avec SoPlex	296,64	174,03	58,33	402,65	794,44
BC1 avec CPLEX	140,44	63,36	20,65	196,81	316,29
CPLEX avec L8$^+$	5152,23	450,34	25,24	165,45	917,79
BPC1 avec SoPlex	91,74	70,95	24,27	107,22	193,96
BPC1 avec CPLEX	38,63	25,97	10,08	45,67	100,20
BPC2 avec SoPlex	68,43	56,51	24,67	113,27	181,40
BPC2 avec CPLEX	29,28	23,81	9.04	44,87	65,39

les temps pour la génération des coupes, la génération des variables, la méthode primale du simplex et la méthode duale du simplex respectivement. Les deux dernières colonnes donnent le nombre de variables générées en pourcentage ainsi que le nombre de coupes générées tout au long de la recherche.

Tableau 6.14 Temps en secondes des différents ingrédients pour les *Branch, Price and Cut* pour le graphe *g15*.

Algorithmes	coupes	pricing	primal	dual	#vars (%)	#coupes
BPC1 avec SoPlex	1,61	4,75	155,48	20,84	9,27	70
BPC2 avec SoPlex	1,62	1,76	153,26	13,52	10,58	70
BPC1 avec CPLEX	2,22	4,98	65,05	16,60	11,15	70
BPC2 avec CPLEX	2,63	1,94	29,09	20,36	12,07	70

Nous pouvons voir dans le tableau 6.14 qu'effectivement lorsque nous faisons une mise à jour implicite du dual, nous gagnons du temps dans le problème du *pricing* et générons plus de variables. Nous pouvons voir aussi que l'abondance de variables lorsque nous utilisons l'algorithme BPC2 permet à la méthode du simplex d'être plus rapide. Nous gagnons donc sur les deux tableaux.

6.6 Conclusions

Notre approche de génération de colonnes donne déjà des résultats surprenants en battant un solveur comme CPLEX de 2 à 133 fois sur nos graphes de référence. Cette approche

peut s'améliorer[15] et est certainement une voie à suivre pour la résolution du PPCC avec les outils de la programmation linéaire en nombres entiers.

Les inégalités valides supplémentaires font elles aussi une différence. Ceci dit, rajouter toutes les inégalités n'est pas la meilleure stratégie à adopter. Comme la génération des coupes prend un temps relativement modeste, nous pouvons les générer toutes et choisir les meilleures d'entre-elles. Cette stratégie semble être excellente tant pour la rapidité globale de l'algorithme que pour le choix des meilleures coupes.

Nous l'avons vu, la mise-à-jour du dual prend du temps. Pour remédier à cela plusieurs possibilités s'offrent à nous. Nous avons choisi l'une des plus simples : nous passer totalement de la mise à jour du dual. Pour faire cela, il nous a fallu écrire les contraintes sous une forme spéciale. Cette forme n'est pas forcément la plus adaptée pour une résolution rapide des PL mais le gain total en temps, du moins sur nos graphes de référence, est net avec cette nouvelle méthode. Comme il s'agit d'une approximation du dual, on peut craindre que le nombre de variables primales générées ainsi soit beaucoup trop grand. Il n'en est rien comme nous l'avons vu.

15. Voir la section 7.3.4 page 186 des conclusions.

Chapitre 7

CONCLUSIONS

Dans ce dernier chapitre, nous résumons notre travail, évaluons les solutions proposées et regardons quelles améliorations futures seraient souhaitables.

7.1 Synthèse des travaux

Le PPCC est, à notre connaissance, un problème nouveau qui représente bien des défis. Nous avons introduit une définition qui, tout comme pour certaines versions de son petit frère sur les sommets, le PLM, ne comptabilise pas le retour au dépôt et nous avons vu que cette définition n'est en rien restrictive, bien au contraire. D'une complexité fortement NP-difficile, nous avons montré en quoi le PPCC se distingue des problèmes classiques de postier chinois et pourquoi sans doute le paradigme des problèmes de postier chinois — à savoir trouver l'augmentation minimale d'un graphe pour le rendre eulérien puis trouver un tour eulérien — n'est pas adapté à notre version du PPCC. Nous avons aussi montré comment transformer simplement le PPCC en une version du PLM sans retour au dépôt. En fait, les deux problèmes montrent bien des similitudes et mériteraient qu'on les étudie en parallèle.

Parmi les cas polynomiaux que nous exhibons, le cas du cercle et de la bande unitaires sont nouveaux alors que le cas de la droite et de l'arbre unitaire sont adaptés d'algorithmes existants pour le PLM sur ces graphes. Plus généralement, nous pensons que le cas de la grille unitaire rectangulaire sans trou est lui aussi résolvable en temps polynomial. Nous avons reformulé la conjecture que nous proposions dans notre mémoire de maîtrise (van Omme, 2003) et qui ne distinguait pas assez de cas.

Le but de cette thèse est de résoudre exactement le PPCC pour n'importe quel type de graphe. Nous avons fait essentiellement deux hypothèses pour aborder ce problème difficile. D'une part, nous utilisons les outils de la programmation linéaire en nombres entiers et d'autre part nous utilisons les solveurs de PL comme des boîtes noires auxquelles nous ne touchons pas.

Dans un premier temps, nous nous sommes attelés à trouver un modèle linéaire en nombres entiers pour résoudre le PPCC. Dans notre quête, nous avons développé une vingtaine de modèles. Dans cette thèse, nous présentons les huit modèles qui possèdent les meilleures bornes inférieures obtenues par relaxation des conditions d'intégralité et nous démontrons toutes les relations de dominances entre ces huit modèles. Notre meilleur modèle, le modèle L8 sort du lot et domine tous les autres modèles sauf un. Ce modèle L8 est l'aboutissement de nos recherches et c'est celui que nous utilisons pour résoudre le PPCC.

Nous présentons trois algorithmes exacts et une simple heuristique pour résoudre le PPCC.

Notre heuristique consiste en une énumération implicite et limitée de toutes les solutions. Nous l'utilisons pour obtenir des bornes supérieures mais aussi pour obtenir un lot de variables primales dans notre approche de génération de colonnes.

Notre premier algorithme, BC1, est un algorithme d'évaluation et séparation progressive qui utilise le modèle L8. Dans la foulée, nous avons développés trois branchements, sept pré-traitements, six familles de coupes dont trois que nous généralisons. Ces outils nous permettent déjà de battre le solveur CPLEX par un facteur de 3 à 58 sur nos graphes de référence.

Notre deuxième algorithme, BPC1, utilise une meilleure variante du modèle L8 : le modèle L8$^+$ ainsi qu'une approche de génération de colonnes. Il s'agit d'un *Branch, Price and Cut* où à chaque nœuds de l'arbre de recherche sont introduites non seulement de nouvelles variables primales mais aussi de nouvelles coupes. Cette fois-ci, nous développons en plus des outils précédents, cinq familles de coupes dont nous en généralisons quatre. Cette nouvelle approche, plus rapide que celle de l'algorithme BC1 d'un facteur de 2 à 4, nous permet d'être de 2 à 133 fois plus rapide que le solveur CPLEX utilisant le modèle L8$^+$ sur nos graphes de référence.

Finalement, nous améliorons notre approche de génération de colonnes avec le troisième algorithme, BPC2, avec une évaluation implicite du dual, ce qui nous permet de faire des gains relativement modestes mais bien réels.

7.2 Évaluation des solutions proposées

Si nos algorithmes fonctionnent mieux que CPLEX, ils ne peuvent résoudre le PPCC que sur des graphes relativement petits[1]. Dans nos tests, nous considérons un temps raisonnable quand il ne dépasse pas les 3600 secondes. Si nous prenons des graphes complets avec des coûts compris entre 1 et 100 pour les arêtes par exemple, nous pouvons résoudre en un temps raisonnable K_9 (23 secondes), K_{10} (63 secondes) et K_{11} (1232 secondes) mais pas K_{12}. Pour des graphes quelconques, nous pouvons résoudre n'importe quel graphe avec $|V| \leqslant 11$ et/ou $|E| \leqslant 55$ en moins d'une heure mais si nous passons à 12 sommets où considérons des graphes avec plus de 55 arêtes, nous rencontrons des graphes qui nous résistent même après de nombreuses heures. Bref, l'extensibilité ou mise à l'échelle de nos algorithmes ne fonctionne pas très bien. La raison majeure est le temps nécessaire pour la résolution des PL ainsi que les écarts relatifs entre les valeurs optimales du problème et les valeurs optimales du tout premier LP qui sont de plus en plus grands.

7.3 Améliorations futures

Nous rassemblons dans cette section quelques pistes d'améliorations possibles qui mériteraient plus de développements.

7.3.1 Utilisation de la programmation non linéaire en nombres entiers

Une des hypothèses de base de cette thèse est l'utilisation des outils de la programmation linéaire en nombres entiers mais comme nous l'avons déjà fait remarquer il est possible d'exprimer le PPCC comme un problème non linéaire en nombres entiers. D'une part la fonction objectif, que nous voyons linéaire comme la somme de toutes les latences de chacune des arêtes, peut très bien s'exprimer comme une fonction non linéaire. Le modèle Q1 présenté dans l'annexe D page 201 en est un exemple. D'autre part et de manière peut-être plus convaincante, certaines contraintes peuvent s'exprimer plus naturellement comme des contraintes non linéaires. Par exemple, si nous reprenons le modèle L8, une contrainte sans

1. On observe le même phénomène pour la résolution du PLM où les plus grands problèmes comprennent des graphes de 60 sommets (Fischetti *et al.*, 1993). Quelques problèmes avec des graphes jusqu'à 107 sommets ont été résolus aussi (Abeledo *et al.*, 2010).

doute fort utile est celle-ci :

$$\prod_{j,k} y_{ij}^k = 0 \qquad \forall\ i \in \overline{V}.$$

Cette contrainte permet de forcer la non utilisation d'un plus court chemin issu d'un sommet i. Bref, il serait intéressant de voir dans quelle mesure l'utilisation d'un modèle non linéaire permettrait de mieux décrire le PPCC.

7.3.2 Utilisation d'algorithmes adaptés pour la résolution des PL

La dégénérescence aussi bien primale que duale ainsi que les nombreuses instabilités numériques rencontrées lors de la résolution des PL fournis par le modèle L8 rendent la tâche ardue à n'importe quel solveur. Rappelons que le solveur CLP a très vite dû être abandonné car il n'arrivait tout simplement pas à résoudre ces PL. Rappelons également que jusqu'à la version 9, CPLEX retournait lui aussi parfois des résultats fantaisistes.

Un code spécialement adapté à la résolution du modèle L8 et de ses variantes pourrait sans doute apporter de sérieuses améliorations. Nous pensons par exemple à l'approche de Raymond *et al.* (Voir par exemple Raymond *et al.*, 2010a,b) qui a été spécialement développée pour les PL dégénérés et bat CPLEX d'un facteur de 7 à 12 pour certains d'entre-eux.

Une autre amélioration plus accessible est de démarrer *à chaud* en fournissant une bonne base de départ pour la méthode du simplex.

Comme l'essentiel du temps passé à la recherche d'une solution optimale en nombres entiers est consacré à la résolution des PL, en particulier du tout premier PL, c'est très certainement une voie à explorer.

7.3.3 Meilleures heuristiques

La qualité principale de l'heuristique proposée dans cette thèse est sa rapidité. Il serait sans doute intéressant de consacrer plus de temps à la recherche d'une première solution réalisable par exemple avec l'utilisation d'une méta-heuristique. En effet, pour nos graphes de référence, les écarts relatifs entre la valeur optimale du PPCC et la valeur optimale du premier PL obtenus avec le modèle L8$^+$ varient entre 0 et 3 %[2]. Ces écarts sont déjà

2. Voir le tableau 6.4 page 163.

relativement importants. L'obtention de solutions réalisables optimales ou quasi-optimales nous permettrait de réduire de manière non négligeable la taille de l'arbre de recherche.

7.3.4 Le modèle L8

Meilleures heuristiques

Pendant la recherche d'une solution optimale, nous pourrions aussi exploiter l'information obtenue à partir des solutions fractionnaires fournies pour construire une solution réalisable. Mentionnons tout de suite que l'extraction d'une information pertinente n'est pas aisée : ni SoPlex, ni CLP ni même CPLEX n'arrivent à exploiter ces informations à leur plein potentiel. De fait, les liens entre les solutions fractionnaires et la ou les solutions optimales trouvées ne sont pas évidents. Toutefois, une analyse plus détaillées des solutions fractionnaires nous permettraient peut-être de construire des solutions réalisables de bonnes qualités. Une piste est de découper la solution fractionnaire en plusieurs segments de tournée et puis de recoller ceux-ci entre eux. La difficulté réside d'une part dans la manière de découper la solution fractionnaire — comment découper et quoi découper ? — et d'autre part dans le recollage de ces segments : il faut les recoller entre eux mais dans un ordre différent de celui proposé par la solution fractionnaire. Enfin, ce recollage ne peut s'effectuer que jusqu'à une certaine période de temps. Au-delà, il faut construire les fins de solutions. Une telle approche mériterait d'être étudiée plus en détails et pourrait donner de bons résultats.

Pré-traitements sur les variables

Les pré-traitements présentés sont très effectifs mais il y aurait moyen d'en trouver d'autres. Par exemple, nous pourrions développer des pré-traitements adaptés à certaines structures de graphe. Nous pensons par exemple à la présence d'un point d'articulation. Celui-ci introduit certaines impossibilités qui nous permettent d'éliminer certaines variables. Il nous faut alors déterminer les points d'articulation ce qui demande l'utilisation d'algorithmes non triviaux. Comme nous l'avons vu dans la section 6.3.1 page 163, il n'est pas forcément avantageux de retirer toutes les variables éliminables dans une approche de génération de colonnes. Il faut donc voir au cas par cas si le jeu en vaut la chandelle.

Inégalités valides localement

Toutes nos inégalités valides sont globales, *i.e.* valides pour tous les nœuds de l'arbre de recherche, en particulier pour la racine. Les inégalités locales reposant sur des hypothèses

plus fortes puisque seulement valides localement, nous pourrions penser qu'elles seraient plus efficaces que des inégalités valide globalement. Pour générer ces coupes locales, nous devons pouvoir accéder de manière efficace à l'historique de la recherche. Là aussi, nous pensons que des améliorations importantes sont possibles.

Amélioration du calcul des coûts réduits

Dans nos deux algorithmes BCP1 et BCP2, nous calculons à chaque itération le coût réduit de toutes les variables. Plusieurs améliorations sont possibles. D'une part, nous pouvons envisager des stratégies où nous ne nous occupons que d'un sous-ensembles de variables (Voir par exemple *partial pricing* dans Lübbecke et Desrosiers, 2005). Par exemple, les x meilleures, *i.e.* qui ont les coûts réduits les plus négatifs lors d'une itération ou bien un quota de variables par type de variables, etc. D'autre part, il y a moyen de calculer le coût réduit des variables par groupes de variables plutôt qu'individuellement. On peut bien sûr combiner les deux techniques ensemble.

7.3.5 Programmation par contraintes

Une approche intéressante est la programmation par contraintes ou mieux encore une approche hybride entre la programmation linéaire en nombres entiers et la programmation par contrainte (Voir par exemple Achterberg *et al.*, 2008; Ottosson, 2000). Voici une approche possible.

Si nous prenons comme variable $v_e = k$ si on dessert l'arête $e \in \mathrm{E}$ en $(m-k+1)^{\mathrm{e}}$ position, nous pourrions peut-être bénificier de la fameuse contrainte `alldifferent`$(v_{e_1}, \ldots, v_{e_m})$?

Références

ABELEDO, H. G., FUKASAWA, R., PESSOA, A. A. et UCHOA, E. (2010). The time dependent traveling salesman problem: Polyhedra and branch-cut-and-price algorithm. *SEA*. 202–213.

ACHTERBERG, T. (2008). *Constraint Integer Programming*. Thèse de doctorat.

ACHTERBERG, T. (2009). Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1, 1–41.

ACHTERBERG, T., BERTHOLD, T., KOCH, T. et WOLTER, K. (2008). Constraint integer programming: A new approach to integrate cp and mip. L. Perron et M. A. Trick, éditeurs, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008*. vol. 5015 de *Lecture Notes in Computer Science*, 6 – 20.

AFRATI, F., COSMADAKIS, S., PAPADIMITRIOU, C., PAPAGEORGIOU, G. et PAPAKOSTANTINOU, N. (1986). The complexity of the traveling repairman problem. *RAIRO Informatique Theorique et Applications*, 20, 79–87.

APPLEGATE, D., BIXBY, R., CHVÁTAL, V. et COOK, W. (1994). Finding cuts in the tsp.

APPLEGATE, D. L., BIXBY, R. E., CHVÁTAL, V. et COOK, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA.

ARCHER, A. et BLASIAK, A. (2010). Improved approximation algorithms for the minimum latency problem via prize-collecting strolls. *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, SODA '10, 429–447.

ARCHER, A., LEVIN, A. et D.P. WILLIAMSON, D. P. (2008). A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing*, 37, 1472–1498.

ARCHER, A. et WILLIAMSON, D. (2003). Faster approximation algorithms for the minimum latency problem. *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. 88–96.

ARORA, S. et KARAKOSTAS, G. (2000). A 2+epsilon approximation algorithm for the k -MST problem. *Symposium on Discrete Algorithms*. 754–759.

ARORA, S. et KARAKOSTAS, G. (2003). Approximation schemes for minimum latency problems. *SIAM Journal on Computing*, 32, 1317–1337.

BARAHONA, F. et GRÖTSCHEL, M. (1986). On the cycle polytope of a binary matroid. *J. Comb. Theory Ser. B*, 40, 40–62.

BIANCO, L., MINGOZZI, A. et RICCIARDELLI, S. (1993). The traveling salesman problem with cumulative costs. *Networks*, 23, 81–91.

BIGRAS, L., GAMACHE, M. et SAVARD, G. (2008). The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. *Discrete Optimization*, 5, 685–699.

BLUM, A., CHALASANI, P., COPPERSMITH, D., PULLEYBLANK, B., RAGHAVAN, P. et SUDAN, M. (1994). The minimum latency problem. *Proc. 26th ACM Symposium on the Theory of Computing*. 163–171.

CHAUDHURI, K., GODFREY, B., RAO, S. et TALWAR, K. (2003). Paths, trees, and minimum latency tours. *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. 36.

DROR, M., éditeur (2000). *Arc Routing: Theory, Solutions and Applications - Linear Programming Based Methods for Solving Arc Routing Problems*. Kluwer Academic Publishers.

EDMONDS, J. et JOHNSON, L. (1973). Matching, euler tours and the chinese postman problem. *Math. Prog.*, 5, 88–124.

EISELT, H. A., GENDREAU, M. et LAPORTE, G. (1995a). Arc routing problems, part I : The chinese postman problem. *Operation Research*, 43, 231–242.

EISELT, H. A., GENDREAU, M. et LAPORTE, G. (1995b). Arc routing problems, part II : The rural postman problem. *Operation Research*, 43, 399–413.

EULER, L. (1736). Solutio Problematis ad Geometrian Situs Pertinentis. *Commentarii academiae scientarum Petropolitanae*, 8, 128–140.

EZZINE, I. O., SEMET, F. et CHABCHOUB, H. (2010). New formulations for the traveling repairman problem.

FISCHETTI, M., LAPORTE, G. et MARTELLO, S. (1993). The delivery man problem and cumulative matroids. *Operations Research*, 41, 1055–1064.

FORD, L. R. et FULKERSON, D. R. (1962). *Flows in Networks*. Princeton University Press.

FREDERICKSON, G. N. (1979). Approximation algorithms for some postman problems. *Journal of the Association for Computing Machinery*, 26, 538–554.

GARCÍA, A., JODRÁ, P. et TEJEL, J. (2002). A note on the travelling repairman problem. *Networks*, 40, 27–31.

GARG, N. (1996). A 3-approximation for the minimum tree spanning k vertices. *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*.

GHIANI, G. et LAPORTE, G. (2000). A branch-and-cut algorithm for the Undirected Rural Postman Problem. *Mathematical Programming*, V87, 467–481.

GODINHO, M. T., GOUVEIA, L. et PESNEAU, P. (2010a). Hop-indexed circuit-based formulations for the traveling salesman problem. *Electronic Notes in Discrete Mathematics*, 36, 1049–1056.

GODINHO, M. T., GOUVEIA, L. et PESNEAU, P. (2010b). Natural and extended formulations for the time-dependent travelling salesman problem.

GOEMANS, M. et KLEINBERG, J. (1998). An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82, 111–124.

GOEMANS, M. et WILLIAMSON, D. (1995). A general approximation technique for constrained forest problems. *SIAM J. Computing*, 24, 296–317.

GOUVEIA, L. et VOß, S. (1995). A classification of formulations for the (time-dependent) traveling salesman problem. *European Journal of Operational Research*, 83, 69–82.

GUSFIELD, D. (1990). Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing*, 19, 143–155.

HELD, M. et CARP, R. (1962). A dynamic programming approach to sequencing problems. *SIAM J. Appl. Math.*, 10 (1), 196–210.

HIERHOLZER, C. (1873). Uber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Mathematische Annalen*, VI, 30–32.

JOTHI, R. et RAGHAVACHARI, B. (2007). Approximating the k-traveling repairman problem with repairtimes. *Journal of Discrete Algorithms*, 5, 293–303.

KARA, İ., KARA, B. Y. et YETIŞ, M. K. (2008). *Vehicle Routing Problem*, InTech, chapitre 6. 85–98.

KOUTSOUPIAS, E., PAPADIMITRIOU, C. et YANNAKAKIS, M. (1996). Searching a fixed graph. *ICALP '96: Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*. 280–289.

LAPORTE, G. (1995). Modeling and solving several classes of arc routing problems as traveling salesman problems. Rapport technique CRT-95-81, Centre de Recherche sur les Transports.

LETCHFORD, A. N., REINELT, G. et THEIS, D. O. (2008). Odd minimum cut sets and b-matchings revisited. *SIAM J. Discret. Math.*, 22, 1480–1487.

LÜBBECKE, M. E. et DESROSIERS, J. (2005). Selected topics in column generation. *Oper. Res.*, 53, 1007–1023.

LUCENA, A. (1990). Time-dependent traveling salesman problem - the deliveryman case. *Networks*, 20, 753–763.

MÉNDEZ-DíAZ, I., ZABALA, P. et LUCENA, A. (2008). A new formulation to the traveling deliveryman problem. *Discrete Applied Mathematics*, 156, 3223 —3237. En prensa.

MINIEKA, E. (1989). The delivery man problem on a tree network. *Annals of Operations Research*, 18, 261–266.

NEMHAUSER, U. O. P. G. L. (1994). The age of optimization: solving large-scale real-world problems. *Operations Research*, 5–13.

NOON, C. E. et BEAN, J. C. (1991). An efficient transformation of the generalized traveling salesman problem. Rapport technique, University of Michigan.

OTTOSSON, G. (2000). *Integration of Constraint Programming and Integer Programming for Combinatorial Optimization*. Thèse de doctorat, Uppsala University.

PADBERG, M. W. et RAO, M. R. (1982). Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7, 67–80.

PAPADIMITRIOU, C. H. et YANNAKAKIS, M. (1993). The traveling salesman problem with distances one and two. *Mathematics Of Operations Research*, 18(1), 1–11.

PICARD, J.-C. et QUEYRANNE, M. (1978). The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *OPERATIONS RESEARCH*, 26, 86–110.

RAGHAVACHARI, B. et VEERASAMY, J. (1999). Approximation algorithms for the asymetric postman problem. *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*. 734–741.

RAYMOND, V., SOUMIS, F., METRANE, A. et DESROSIERS, J. (2010a). Positive edge: A pricing criterion for the identification of non-degenerate simplex pivots. Rapport technique G–2010–61, Les Cahiers du GERAD.

RAYMOND, V., SOUMIS, F. et ORBAN, D. (2010b). A new version of the improved primal simplex for degenerate linear programs. *Comput. Oper. Res.*, 37, 91–98.

REINELT, G. (1991). Tsplib - a traveling salesman problem library. *Informs Journal on Computing*, 3, 376–384.

SAHNI, S. et GONZALEZ, T. (1976). P-complete approximation problems. *Journal of the Association for Computing Machinery*, 23, 555–565.

SARRUBI, J. F. M. et LUNA, H. P. L. (2007). A new flow formulation for the minimum latency problem. *Proceedings of the Internacional Network Optimization Conference*.

SHINANO, Y., ACHTERBERG, T., BERTHOLD, T., HEINZ, S. et KOCH, T. (2010). Parascip – a parallel extension of scip. Rapport technique, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB).

SIMCHI-LEVI, D. et BERMAN, O. (1991). Minimizing the total flow time of n jobs on a network. *IIE Transactions*, 23, 236 –244.

SITTERS, R. (2002). The minimum latency problem is np-hard for weighted trees. *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*. 230–239.

STECCO, G., CORDEAU, J.-F. et MORETTI, E. (2008). A branch-and-cut algorithm for a production scheduling problem with sequence-dependent and time-dependent setup times. *Computers and Operations Research*, 35, 2635–2655.

VAN AARDENNE-EHRENFEST, T. et DE BRUIJN, N. (1951). Circuits and trees in oriented linear graphs. *Simon Stevin*, 28, 203–217.

VAN EIJL, C. (1995). A polyhedral approach to the delivery man problem. Rapport technique, Eindhoven University of Technology.

VAN OMME, N. (2003). *Le problème du postier chinois cumulatif*. Mémoire de maîtrise, Université de Montréal.

WIEL, R. J. V. et SAHINIDIS, N. V. (1996). An exact solution approach for the time-dependent traveling-salesman problem. *Naval Research Logistics*, 43, 797–820.

WU, B. (2000). Polynomial time algorithms for some minimum latency problems. *Information Processing Letters*, 75, 225–229.

WU, B., HUANG, Z.-N. et ZHAN, F.-J. (2004). Exact algorithms for the minimum latency problem. *Information Processing Letters*, 92, 303–309.

Annexe A

Les quinze graphes de référence

A.1 Les quinze graphes de référence

Nous reprenons ici brièvement les caractéristiques principales de nos quinze graphes de référence dans le tableau A.1.

Tableau A.1 Caractéristiques principales des quinze graphes de référence.

graphes	n	m	min	max	z_{opt}
g1	3	3	1	50	57
g2	5	4	1	64	146
g3	8	10	1	1	58
g4	7	8	1	10 000 000	12591503
g5	5	5	1	10	32
g6	10	15	8	99	6223
g7	6	7	6	14	315
g8	15	20	1	99	11533
g9	20	35	1	95	24739
g10	15	35	2	96	18941
g11	7	10	5	93	1829
g12	15	20	2	98	6015
g13	10	30	1	100	21407
g14	15	40	9	100	33877
g15	10	45	2	99	38900

Annexe B

Les bornes inférieures des huit modèles sur les quinze graphes de référence

B.1 Les bornes inférieures des huit modèles sur les quinze graphes de référence

Tableau B.1 Les *bi* des huit modèles présentés pour les quinze graphes de référence.

z_{opt}	L1	L2	L3	L4	L5	L6	L7	L8	Graphes
57	56	55	55	56	55	55	57	57	*g1*
146	116.5	115.5	112.03	121.25	140.5	146	146	146	*g2*
58	55	55	55	55	58	55	55	55	*g3*
12591503	12345698	12345695	12345678	12399363.75	12345701.9	12365963.5	12591503	12591503	*g4*
32	32	32	31.0028	32	32	32	32	32	*g5*
6223	5638.62	5607.47	5627.29	5700.28	5647.4	5567.98	5566	5968.5	*g6*
315	290.83	277.38	283.03	290.83	288	277	274.5	305.5	*g7*
11533	9279.64	9277.49	9253.48	9398.17	9422.3	9364.85	9300.25	11127.13	*g8*
24739	21441	21916.9	21431.13	22099.14	21658.47	20951.92	20943.17	23728	*g9*
18941	17524.5	17658.4	17470.8	17747.7	17539	17264.6	17271.7	18342	*g10*
1829	1581.38	1574.5	1559.63	1607.31	1579.525	1558	1561.83	1811.25	*g11*
6015	4877.5	5053.35	4852.83	5169.37	4925.08	4803.45	4771.75	5921.6	*g12*
21407	20537.42	20628.15	20525.57	20700.49	20592.77	20496.17	20492.54	21115.73	*g13*
33877	32367.96	32393.44	32263.37	32449.55	32288.57	32205	32201,11	33523.34	*g14*
38900	38272.6	38324.9	38217.7	38350.8	38250.6	38219.1	38213.6	38732.82	*g15*

Le tableau B.1 reprend les bornes inférieures obtenues par relaxation des conditions d'intégralité pour les modèles L1-L8 pour les quinze graphes de référence.

Annexe C

Équivalence entre les modèles L1, L2 et L3

C.1 Équivalence entre les modèles L1, L2 et L3

Pour les deux démonstrations qui suivent nous ne considérons que des solutions *modèles* définies à la section 1.4.6 page 13.

C.1.1 Équivalence entre les modèles L1 et L2

Lemme 33 *Les modèles L1 et L2 sont équivalents.*

Démonstration

Cette démonstration s'articule en trois temps : tout d'abord, nous montrons comment contruire une solution réalisable de L1 à partir d'une solution modèle de L2. Puis nous faisons de même pour construire une solution réalisable de L2 à partir d'une solution modèle de L1. Enfin, nous montrons que ces deux solutions ont exactement le même coût.

Soit $\{x_{ij}^k, y_{ij}^k\}$ une solution de L2. Construisons une solution équivalente pour le modèle L1 comme suit. Soit $e = (i,j)$ une arête du graphe original. Prenons

$$\begin{aligned} v_e^k &= y_{ij}^k + y_{ji}^k \\ d_e^k &= x_{ij}^k + x_{ji}^k - \sum_{l=1}^{k-1}(x_{ij}^k + x_{ji}^k) && \text{(C.1)} \\ V_i^k &= x_{iT}^k && \text{(C.2)} \end{aligned}$$

et $V_A^0 = 1$ pour le dépôt et $V_S^0 = 0$ pour tout sommet S autre que le dépôt. Les D_S^k sont obtenus grâce à (1.4).

Une telle solution est bien une solution réalisable du modèle L1. (1.1) et (1.2) sont vérifiées grâce à (4.7). (1.3) est l'équivalent de (4.5) et (1.5) est l'équivalent de (4.3). (1.6) et (1.7) sont vérifiées par construction. Il reste à démontrer que (1.4) est aussi vérifiée.

Remarquons tout d'abord que les équations de conservation de flots (4.2) demandent la construction d'un chemin partiel débutant par le dépôt pour chaque étape k. Donc, nécessairement à chaque étape k, nous rajoutons un diamètre. Ces équations de flots correspondent aux contraintes analogues (1.4).

En effet, soit une étape $k \neq 1$ et considérons le sommet S. Soit celui-ci est au début du diamètre et donc $x_{ST}^{k-1} = 1 = V_S^{k-1}$, $x_{ST}^k = 0 = V_S^k$ et $\sum_{e:S\in e} d_e^k = 1$ par (C.1) car il s'agit du début du diamètre. Dans ce cas, il suffit de prendre $D_S^k = 0$ et (1.4) est vérifiée.

Soit ce sommet S est au milieu d'un diamètre. Dès lors par (C.1) ; $\sum_{e:S\in e} d_e^k = 2$ et par (C.2) : $V_S^{k-1} = V_S^k = 0$. Il suffit alors de fixer $D_S^k = 1$ et (1.4) est vérifiée.

Soit ce sommet S est à la fin du diamètre. Dès lors par (C.1) : $\sum_{e:S\in e} d_e^k = 1$ et par (C.2) : $V_S^{k-1} = 0$ et $V_S^k = 1$. Il suffit alors de fixer $D_S^k = 0$ et (1.4) est vérifiée.

Le cas où $k = 1$ se traite de manière analogue en tenant compte du fait que $V_A^0 = 1$ pour le dépôt et $V_S^0 = 0$ pour tout sommet S autre que le dépôt.

La construction d'une solution réalisable de L2 à partir d'une solution réalisable de L1 est similaire. Soit $\{v_e^k, d_e^k, V_S^k, D_S^k\}$ une solution réalisable modèle de L1. Pour chaque diamètre d'une étape k $\{d_e^k\}$ commençant en i ($V_i^{k-1} = 1$) et se terminant en j ($V_j^k = 1$), faisons correspondre les diamètres pour les périodes de temps k à m en fixant les variables orientés correspondantes :

$$x_{ip_1}^l = x_{p_1p_2}^l = \ldots = x_{p_aj}^l = 1 \ \forall \ k \leqslant l \leqslant m. \tag{C.3}$$

Si un x_{ab}^l est non nulle, nous incrémentons sa valeur de 1. Le dernier arc du diamètre est considéré en fixant $y_{p_aj}^l = 1 \ \forall \ k \leqslant l \leqslant m$. Fixons aussi les variables

correspondantes aux sommets de fin de diamètre : $x_{iT}^k = V_S^k$ pour $k \geqslant 1$.

Il s'agit bien d'une solution réalisable pour le modèle L2. (4.1), (4.2), (4.3) et (4.7) se vérifient par construction. Vérifions (4.4). Soit le sommet i est le dernier d'un diamètre à l'étape k et donc $x_{iT}^k = V_i^k = 1$. Par construction, $\delta_G^+(i)x_{ij}^k = 0$. À l'étape $k+1$, le sommet i est le début d'un nouveau diamètre. Par (C.3), $\delta_G^+(i)x_{ij}^{k+1} \geqslant 1$ et donc (4.4) est vérifiée. Si i n'est pas le dernier sommet d'un diamètre, alors (4.4) est vérifiée par (4.1).

(4.5) est l'équivalent de (1.3).

Pour les solutions modèles que nous considérons, (4.6) est automatiquement vérifiée.

Vérifions maintenant que ces les deux solutions ont bien le même coût. Pour ce faire, nous pouvons nous concentrer sur le coût de chaque diamètre. À l'étape 1, le diamètre est constitué d'une seule arête e. Pour le modèle L1, la contribution de ce diamètre e dans la fonction objectif est de $c(e) \cdot m$. Pour le modèle L2, nous obtenons

$$\sum_{k=1}^{m} c_e x_e^k = c_e \sum_{k=1}^{m} x_e^k = c_e \cdot m.$$

Soit un diamètre à l'étape k. Pour le modèle L1, sa contribution est de

$$(m-k+1) \cdot \sum_{e \in \text{diamètre}} d_e^k.$$

Pour le modèle L2, cette contribution est de

$$\begin{aligned}
\sum_{l=1}^{m} \sum_{(i,j)\in \text{diamètre}} (x_{ij}^l + x_{ji}^l) &= \sum_{l=k}^{m} \sum_{e\in \text{ diamètre}} x_e^l \\
&= \sum_{e\in \text{ diamètre}} \sum_{l=k}^{m} x_e^l \\
&= \sum_{e\in \text{ diamètre}} x_e (\sum_{l=k}^{m} 1) \\
&= (m-k+1) \sum_{e\in \text{ diamètre}} x_e.
\end{aligned}$$

où $x_e = \begin{cases} 1 & \text{si } e \text{ appartient au diamètre de l'étape } k. \\ 0 & \text{sinon.} \end{cases}$

Comme tous ces termes sont égaux deux à deux, nous avons que les valeurs des deux fonctions objectifs coïncident :

$$\sum_{e\in \mathrm{E}} c(e) \left[\sum_{k=1}^{m} (m-k+1) \cdot d_e^k \right] = \sum_{k=1}^{m} \sum_{(i,j)\in \mathrm{E}} c_{ij} \, (x_{ij}^k + x_{ji}^k).$$

□

C.1.2 Équivalence entre les modèles L2 et L3

Lemme 34 *Les modèles L2 et L3 sont équivalents.*

Démonstration

Comme la démonstration précédente, celle-ci s'articule de la même manière en trois temps : nous montrons d'abord comment définir une solution équivalente pour le modèle L3 à partir d'une solution modèle pour le modèle L2, puis nous faisons l'inverse et enfin nous montrons que ces deux solutions ont le même coût.

Soit $\{x_{ij}^k, y_{ij}^k\}$ une solution de L2. Construisons une solution équivalente pour

le modèle L3 comme suit. Soit un arc $a = (i,j)$.

$$\begin{aligned} y_a^1 &= y_{ij}^1 \\ y_a^2 &= y_{ij}^2 - y_{ij}^1 \\ y_a^k &= y_{ij}^k - y_{ij}^{k-1}. \end{aligned}$$

Les y_a^k respectent les contraintes (4.8), (4.9) et (4.11) du modèle L3 :
(4.8) et (4.9) sont respectées grâce à (4.7) car d'une part pour $k = m$ cette contrainte exige que toutes les arêtes aient été desservies une seule fois et d'autre part comme elles ont été desservies les unes après les autres (pour $k = 1$ jusqu'à $k = m$), il a fallu en desservir une et une seule à chaque étape. (4.11) est vérifiée aussi puisque nous ne considérons que des solutions qui desservent une arête à partir du dépôt à l'étape 1.

Nous nous occuperons des contraintes (4.10) qui permettent de définir les variables z^k lorsque nous comparerons la valeur des fonctions objectifs.

Soit maintenant $\{y_a^k, z^k\}$ une solution modèle de L3. Construisons une solution réalisable correspondante pour L2. Soit un arc $a = (i,j)$ auquel correspond l'arête e dans le graph original. Alors nous prenons :

$$\begin{aligned} y_e^1 &= y_{ij}^1 + y_{ji}^1 \\ y_e^2 &= y_{ij}^1 + y_{ji}^1 + y_{ij}^2 + y_{ji}^2 \\ y_e^k &= \sum_{l=1}^{k} (y_{ij}^l + y_{ji}^l). \end{aligned}$$

Les $\{x_{ij}^k\}$ correspondants sont obtenus en prenant des plus courts chemins constitués d'arcs déjà desservis auparavant entre le sommet d'arrivée de l'arc (a,b) et le sommet de départ de l'arc (c,d) lorsque dans le modèle L3, $y_{ab}^{l-1} = 1$ et $y_{cd=1}^l$ avec $l \leqslant k$. Si la variable x_{ij}^k est non nulle lorsque nous rajoutons un diamètre, nous l'incrémentons de 1.

À chaque période k, le diamètre correspondant se termine en un sommet i de V. Nous prenons alors $x_{iT}^k = 1$.

Une telle solution vérifie bien les contraintes du modèle L2. En effet, (4.1), (4.2), (4.3) et (4.4) sont vérifiées par construction. (4.5) est vérifiée par construction aussi car si $y_{ij}^k = 1$ pour une arête $e = (i,j)$ alors par construction x_{ij}^k ou x_{ji}^k est non nulle. De même, (4.6) est vérifiée pour la même raison et le fait qu'on ne peut repasser plus de $m-1$ sur une même arête pour les solutions considérées. Finalement, (4.7) est vérifiée grâce à (4.8) et (4.9).

Il nous reste maintenant à démontrer que les deux solutions ont le même coût, c'est-à-dire que

$$\sum_{k=1}^{m} \sum_{(i,j)\in \mathrm{E}} c_{ij}\,(x_{ij}^k + x_{ji}^k) = m \cdot \left(\sum_{a\in\bar{V}} c(a) \cdot y_a^1 \right) + \sum_{k=2}^{m} (m-k+1) \left(\sum_{a\in\bar{V}} c(a) \cdot y_a^k + z^k \right).$$

On peut montrer l'égalité pour chaque terme correspondant à une arête.

Pour la première arête $e = (i,j)$ desservie (avec i représentant le dépôt, $y_{ij}^1 \overset{L2}{=} 1$ et $y_{ij}^1 \overset{L3}{=} 1$) on a d'une part que

$$\sum_{k=1}^{m} c_{ij}\,(x_{ij}^k + x_{ji}^k) = \sum_{k=1}^{m} c_{ij}\, x_{ij}^k = c_{ij} \sum_{k=1}^{m} x_{ij}^k = c_{ij} \cdot m$$

car dans les solutions que nous considérons $x_{ij}^k + x_{ji}^k \leqslant 1$ et (4.1) impose que $x_{ij}^1 = x_{ij}^2 = \ldots = x_{ij}^m = 1$. D'autre part, $m \cdot \left(\sum_{a\in\bar{V}} c(a) \cdot y_a^1\right) = m \cdot c(a)$ avec $a = (i,j)$ car par construction, il n'y a que'une seule variable y_a^1 non nulle (ce que (4.9) implique aussi). Donc

$$\sum_{k=1}^{m} c_{ij}\,(x_{ij}^k + x_{ji}^k) = c_{ij} \cdot m = m \cdot \left(\sum_{a\in\bar{V}} c(a) \cdot y_a^1 \right) = m \cdot c(a)$$

avec $a = (i,j)$.

Soit une arête e desservie à la période k et soit $\{x_{ij}^k \ : (i,j) \in A\}$ les variables du modèle L2 correspondantes au diamètre de la période k. Pour les solutions que nous considérons, on a que ce diamètre intervient dans la fonction objectif

pour

$$(m-k+1) \sum_{(i,j)\in \text{ diamètre}} c_{ij}\, x_{ij}^k$$

à cause de la contrainte (4.1). Pour la variable x_{ij}^k correspondante à l'arête e desservie à la période k, c'est précisément ce que nous donne le terme $(m-k+1)\cdot c(a)\cdot y_a$ de la fonction objectif du modèle L3. Montrons que la valeur $(m-k+1)\cdot z^k$ correspond exactement à la contribution de la liaison inter-clients entre l'arête desservie à la période $k-1$ et celle desservie à la période k. Les z^k sont définis par (4.10) :

$$(y_i^{k-1}-1)\cdot M + y_j^k \cdot c(i,j) \leqslant z^k \tag{4.10}$$

où $c(i,j)$ représente le coût du plus court chemin entre le sommet d'arrivée de l'arc i et le sommet de départ de l'arc j. Or c'est précisément la valeur que prend z^k. Si $y_i^{k-1}=1$ et $y_j^k=1$, on a $c(i,j)\leqslant z^k$. Avec la minimisation, nous avons l'égalité $z^k = c(i,j)$. La contribution de cette liaison inter-clients dans le modèle L2 pour la fonction objectif est donc

$$(m-k+1) \sum_{(i,j)\in \text{ liaison inter-clients}} c_{ij}\, x_{ij}^k.$$

Nous avons donc bien que

$$\sum_{k=1}^{m} \sum_{(i,j)\in \mathrm{E}} c_{ij}\,(x_{ij}^k + x_{ji}^k) = m\cdot\left(\sum_{a\in\bar{V}} c(a)\cdot y_a^1\right) + \sum_{k=2}^{m}(m-k+1)\left(\sum_{a\in\bar{V}} c(a)\cdot y_a^k + z^k\right).$$

et les deux solutions sont équivalentes.

□

Comme d'une part les modèles L1 et L2 et d'autre part les modèles L2 et L3 sont équivalents deux à deux, nous pouvons en déduire que les modèles L1 et L3 sont aussi équivalents. Par manque de place, nous omettons les autres démonstrations d'équivalence entre les modèles restants.

Annexe D

Modèles supplémentaires non présentés

D.1 Modèles supplémentaires non présentés

D.1.1 Quelques modèles non présentés

Q1

Définissons[1] $M = \frac{(m+1)m}{2}$. C'est le nombre de fois maximum que l'ensemble des arêtes peuvent être parcourues dans une solution optimale du PPCC. Ce modèle considère chaque étape comme le passage sur une arêtes. Il y a donc au plus M étapes qui seront parcourues par l'indice k.

Les variables

$$x_{ij}^k = \begin{cases} 1 & \text{si l'arc } (i,j) \text{ est visité (ou desservi) à l'étape } k. \\ 0 & \text{sinon.} \end{cases}$$

$$q_k = \begin{cases} 1 & \text{si une nouvelle arête est desservie en } k. \\ 0 & \text{sinon.} \end{cases}$$

1. Il ne s'agit pas d'un grand M représentant une grande valeur mais d'un nombre d'étapes dans la construction d'une solution.

Le modèle

$$\min_{x_{ij}^k, q_k} \sum_{(i,j)\in \mathrm{E}; i<j} \sum_{k=1}^{M} \sum_{p=1}^{k} c_{ij}(x_{ij}^p + x_{ji}^p) q_k$$

s.à :

$$\sum_{k=1}^{M} (x_{ij}^k + x_{ji}^k) \geqslant 1 \qquad \forall\ (i,j) \in \mathrm{E}; i < j \tag{D.1}$$

$$\sum_{j\in \delta^+(i)} x_{ij}^{k+1} \geqslant \sum_{j\in \delta^-(i)} x_{ji}^k \qquad \forall\ i \in \mathrm{V}, k \in \{1,\ldots,M-1\} \tag{D.2}$$

$$1 \leqslant \sum_{j\in \delta^+(A)} x_{Aj}^1 \qquad A \text{ est le dépôt} \tag{D.3}$$

$$\sum_{(i,j)\in \mathrm{E}; i<j} (x_{ij}^k + x_{ji}^k) \leqslant 1 \qquad k \in \{1,\ldots,M\} \tag{D.4}$$

$$x_{ij}^k + x_{ji}^k - \sum_{p=1}^{k-1} (x_{ij}^p + x_{ji}^p) \leqslant q_k \qquad \forall\ (i,j) \in \mathrm{E} : i < j, k \in \{1,\ldots,M\} \tag{D.5}$$

$$q_k \leqslant \sum_{(i,j)\in \mathrm{E}; i<j} (x_{ij}^k + x_{ji}^k) \qquad k \in \{1,\ldots,M\} \tag{D.6}$$

et toutes les variables x_{ij}^k et q_k sont binaires.

(D.1) indique que chaque arête doit être visitée au moins une fois. (D.2) se charge d'assurer l'existence d'un chemin. Le dépôt est caractérisé par (D.3). A chaque étape au plus une seule arête peut être visitée, ce qu'impose (D.4). q_k ne vaut 1 que si l'arête visitée à l'étape k est visitée pour la première fois. C'est exactement ce que demandent (D.5) et (D.6).

L'exemple introductif

Reprenons à nouveau le problème introductif de la page 8.

La construction de la solution optimale trouvée par CPLEX est résumée dans la figure D.1.

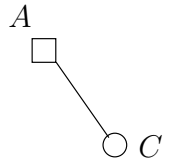
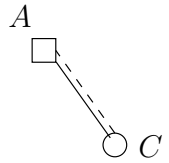
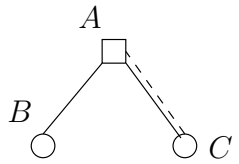
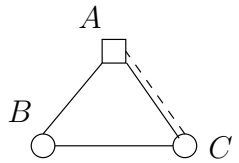
k	arête visitée	coût cumulatif	Chemin partiel
$k = 1$	$x^1_{AC} = 1$	$q_1 = 1$	
$k = 2$	$x^2_{CA} = 1$	$q_2 = 0$	
$k = 3$	$x^3_{AB} = 1$	$q_3 = 1$	
$k = 4$	$x^4_{BC} = 1$	$q_4 = 1$	

FIGURE D.1 Une solution optimale de l'exemple introductif pour le modèle Q1.

L9

On reprend le modèle Q1 de la section D.1.1 dont on linéarise la fonction objectif comme suit. On introduit d'abord les variables dépendantes :

$$Z_k = \begin{cases} & \text{coût cumulatif de l'arête nouvellement desservie en } k. \\ 0 & \text{si en } k \text{ aucune arête n'est desservie.} \end{cases}$$

Soit Q une quantité plus grande que n'importe quel coût cumulatif de n'importe quelle arête pour n'importe quel chemin. On rajoute deux contraintes pour exprimer Z_k et on obtient le modèle suivant :

$$\min_{x_{ij}^k, q_k} \sum_{k=1}^{M} Z_k$$

s.à :

$$(\text{D.1}) - (\text{D.6})$$

$$Z_k \leqslant q_k Q \qquad k \in \{1, \ldots, M\} \tag{D.7}$$

$$(q_k - 1)Q + \sum_{(i,j) \in \mathrm{E}; i<j} \sum_{p=1}^{k} c_{ij}(x_{ij}^p + x_{ji}^p) \leqslant Z_k \qquad k \in \{1, \ldots, M\} \tag{D.8}$$

et les variables x_{ij}^k, q_k sont binaires tandis que les variables Z_k sont entières positives ou nulles.

L10

Nous décomposons le graphe en ses arêtes en dédoublant ses sommets comme sur la figure D.2 et nous exploitons le fait qu'une solution optimale doit nécessairement passer par des plus courts chemins comme nous l'avons démontré dans le lemme 3 page 7. Nous ajoutons pour chaque pair de sommets (i,j) représentant un plus court chemin possible, une arête (en trait discontinue sur la figure) dont le coût est celui du plus court chemin entre les deux sommets.

Un sommet fictif t est rajouté pour permettre la création de la tournée depuis le dépôt. Le nouveau graphe ainsi obtenu est $\overline{G} = (\overline{\mathrm{V}} \cup \{t\}, \mathrm{E} \cup \overline{\mathrm{E}})$ avec $\overline{\mathrm{V}}$ l'ensemble des sommets

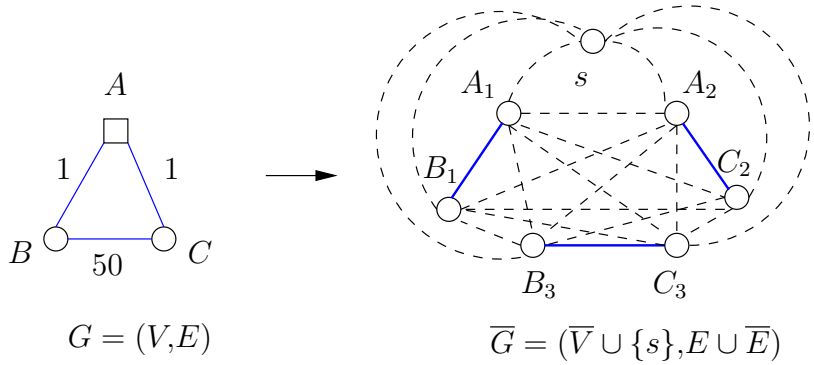


FIGURE D.2 Le graphe auxiliaire utilisé pour le modèle L10.

décuplés de G et $\overline{\mathrm{E}}$ l'ensemble des arêtes représentants les plus court chemins. Les arêtes entre les sommets de $\overline{\mathrm{E}}$ et t sont de coût nul. Il s'agit du *graphe auxiliaire* pour le modèle L10.

Les variables

Soit t_{ij} le temps de fin de service de l'arête (i,j). A chaque sommet i on associe le temps de passage τ_i. En desservant une arête (i,j) on passera soit de i vers j soit l'inverse. Dans le premier cas, nous avons $\tau_j = \tau_i + c_{ij}$, dans le deuxième $\tau_i = \tau_j + c_{ij}$ et le temps de service de (i,j) est soit $t_{ij} = \tau_j$ ou soit $t_{ij} = \tau_i$. Dans les deux cas, nous obtenons $t_{ij} = \frac{\tau_i + \tau_j + c_{ij}}{2}$. Nous pouvons donc transformer la fonction objectif $z = \min \sum_{(i,j)\in \mathrm{E}} t_{ij}$ en $y = \min \sum_{(i,j)\in \mathrm{E}} (\tau_i + \tau_j)$. Pour obtenir la vraie valeur, il suffit de faire $z^* = \frac{y^* + \sum_{(i,j)\in \mathrm{E}} c_{ij}}{2}$

Outre les variables τ_i de temps de passage aux sommets, nous utiliserons les deux jeux suivants de variables :

$$y_{ik} = \begin{cases} 1 & \text{si on parcourt le plus court chemin de } i \text{ à } k. \\ 0 & \text{sinon.} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{si on parcourt l'arête } (i,j) \text{ de } i \text{ à } j. \\ 0 & \text{si le parcourt se fait en sens inverse.} \end{cases}$$

Enfin, on utilise un grand M [2] qui représente une valeur plus grande que le coût cumulatif

2. Nous présentons ce modèle car il est relativement naturel mais il est très instable numériquement : une trop grande valeur de M et nous obtenons des résultats erronés pour la plupart des solveurs utilisés. Voici

du plus grand chemin eulérien possible. Notons par c_{ij} les coûts sur les arêtes originales et par $\tilde{c}_{ik}$ les coûts des plus courts chemins.

Le modèle

$$\min_{x_{ij},y_{ik}} \sum_{(i,j)\in \mathrm{E}} (\tau_i + \tau_j)$$

s.à :

$$\sum_{k\in \overline{\mathrm{V}}\cup\{t\}\setminus\{i,j\}} y_{ik} = 1 - x_{ij} \qquad \forall\,(i,j) \in \mathrm{E} \tag{D.9}$$

$$\sum_{k\in \overline{\mathrm{V}}\cup\{t\}\setminus\{i,j\}} y_{kj} = 1 - x_{ij} \qquad \forall\,(i,j) \in \mathrm{E} \tag{D.10}$$

$$\sum_{k\in \overline{\mathrm{V}}\cup\{t\}\setminus\{i,j\}} y_{ki} = x_{ij} \qquad \forall\,(i,j) \in \mathrm{E} \tag{D.11}$$

$$\sum_{k\in \overline{\mathrm{V}}\cup\{t\}\setminus\{i,j\}} y_{jk} = x_{ij} \qquad \forall\,(i,j) \in \mathrm{E} \tag{D.12}$$

$$\tau_i \geqslant \tau_k + \tilde{c}_{ki} - (1 - x_{ij})M - (1 - y_{ki})M \qquad \forall (i,j) \in \mathrm{E}, \forall k \in \overline{\mathrm{V}} \cup \{t\} \setminus \{i,j\} \tag{D.13}$$

$$\tau_j \geqslant \tau_i + c_{ij} - 2(1 - x_{ij})c_{ij} \qquad \forall\,(i,j) \in \mathrm{E} \tag{D.14}$$

$$\tau_j \geqslant \tau_k + \tilde{c}_{kj} - x_{ij}M - (1 - y_{kj})M \qquad \forall\,(i,j) \in \mathrm{E}, \forall k \in \overline{\mathrm{V}} \cup \{t\} \setminus \{i,j\} \tag{D.15}$$

$$\tau_i \geqslant \tau_j + c_{ij} - 2x_{ij}c_{ij} \qquad \forall\,(i,j) \in \mathrm{E} \tag{D.16}$$

$$\sum_{k\in \overline{\mathrm{V}}} y_{tk} = 1 \tag{D.17}$$

$$\sum_{k\in \overline{\mathrm{V}}} y_{kt} = 1 \tag{D.18}$$

$$\sum_{k\in \overline{\mathrm{V}}|k=\text{dépôt}} y_{tk} = 1 \tag{D.19}$$

et les variables x_{ij} et y_{ik} sont binaires et les variables τ_i sont entières positives ou nulles.

(D.9)-(D.12) permettent la construction de tournées en accordant le sens du traitement d'une arête (i,j) avec le passage sur les plus courts chemins entre les sommets du graphe G. (D.13)-(D.16) forcent les latences τ_i aux sommets à être calculées cumulativement. (D.17) et (D.18) insèrent le sommet fictif t dans la tournée. Finalement, (D.19) indique le dépôt.

une possibilité pour ce grand M mais qui ne fonctionne pas à tous les coups. Soit p la distance du plus grand des plus courts chemins et soit q le coût de l'arête de coût maximum. Nous pouvons borner le coût du plus grand chemin cumulatif par $M = mq + \sum_{k=1}^{m-1}(m-k+1)(p+q) = mq + (p+q)\frac{(m-1)m}{2}$.

L'exemple introductif

Pour l'exemple introductif de la page 8, CPLEX trouve la solution optimale de la figure D.3.

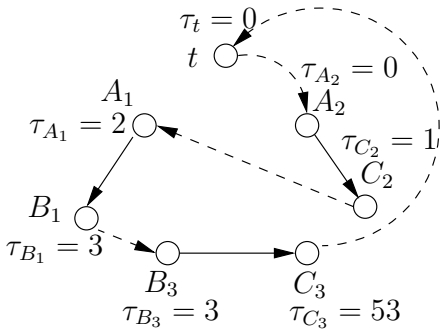


FIGURE D.3 Modèle L10 : une solution optimale de l'exemple introductif. Seules les variables τ sont affichées.

L11

L'approche est celle du modèle L10 de la section D.1.1 dont nous avons légèrement transformé la nomenclature (voir figure D.2). Le sommet de départ est maitenant le sommet s et nous avons ajouté un deuxième sommet fictif représentant un dépôt imaginaire où la tournée se termine. Celui-ci est appelé t.

Nous modélisons la construction d'une solution optimale en exigeant la construction d'une solution réalisable de coût cumulatif minimum.

Trois familles de variables sont utilisées : la première concerne uniquement les liaisons inter-clients et indique si oui ou non celles-ci sont utilisées ou pas. La deuxième permet d'orienter les arêtes alors que la troisième permet de comptabiliser de manière cumulative le coût de la solution en donnant l'ordre de passage des arêtes et des liaisons inter-clients.

$$y_f = \begin{cases} 1 & \text{si on utilise la liaison inter-clients.} \\ 0 & \text{sinon.} \end{cases}$$

$$q_S = \begin{cases} 0 & \text{si le sommet } S \in \overline{\mathrm{V}} \text{ au centre d'une liaison inter-clients et} \\ & \text{d'une arête du graphe dans cet ordre précis.} \\ 1 & \text{sinon.} \end{cases}$$

$$z_e = \begin{cases} k & \text{si on dessert l'arête ou la liaison inter-clients } e \text{ en } (m-k+1)^{\text{e}} \text{ position.} \\ 0 & \text{sinon.} \end{cases}$$

L'orientation des arêtes est « automatique »car le fait d'exiger que le chemin partiel soit composé d'arêtes et de liaisons inter-client couplé avec l'exigence que le premier chemin partiel parte du dépôt permet d'orienter les arêtes automatiquement.

Notons que les variables y_f sont indépendantes.

Le modèle

$$\min_{y_f} \sum_{e \in \mathrm{E}} z_e \cdot c(e) + \sum_{f \in \overline{\mathrm{E}}} z_f \cdot c(f)$$

s.à :

$$z_f \leqslant m \cdot y_f \qquad \forall\, f \in \overline{\mathrm{E}} \tag{D.20}$$

$$q_i + q_j = 1 \qquad \forall\, e = (i,j) \in \mathrm{E} \tag{D.21}$$

$$\sum_{e \in \mathrm{E}; S \in e} z_e = \sum_{f \in \overline{\mathrm{E}}; S \in f} z_f + q_S \qquad \forall\, S \in \overline{\mathrm{V}} \tag{D.22}$$

$$\sum_{f \in \overline{\mathrm{E}}; S \in f} y_f = 1 \qquad \forall\, S \in \overline{\mathrm{V}} \cup \{s,t\} \tag{D.23}$$

$$q_s = 1 \text{ et } \sum_{f \in \overline{\mathrm{E}}; s \in f} z_f = m \qquad \text{Le dépôt } s \tag{D.24}$$

$$q_t = 0 \text{ et } \sum_{f \in \overline{\mathrm{E}}; t \in f} z_f = 0 \qquad \text{Le dépôt } t \tag{D.25}$$

et les variables $0 \leq z_e \leq m$ sont entières tandis que les variables q_S et y_f sont binaires.

(D.20) nous permet de faire le lien entres les variables z_f et y_f. En particulier, si la liaison inter-clients f n'est pas utilisée alors $z_f = 0$. Pour orienter les arêtes, nous utilisons (D.21). Le calcul des coûts cumulatifs se fait grâce à (D.22). Nous ne voulons qu'une seule liaison inter-clients par sommet, ce que nous assure (D.23). Finalement, les dépôts s et t sont définis

par (D.24) et (D.25) respectivement.

L'exemple introductif

Pour mieux comprendre le modèle, utilisons-le pour résoudre le problème introductif de la section 1.3 page 8.

La construction d'une solution optimale par CPLEX est résumée dans la figure D.4.

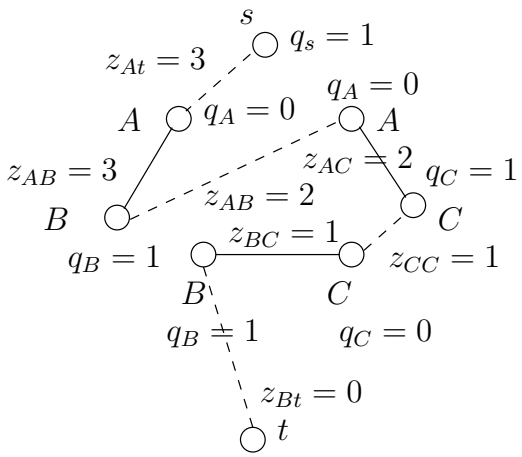


FIGURE D.4 *Une solution optimale détaillée de l'exemple introductif pour le modèle L11.*

L12

Le modèle L1 a des variables dépendantes. Que se passe-t-il si on se limite aux variables indépendantes ? Le modèle L12 est une des différentes tentatives dans ce sens. Nous conservons les variables d_{ij}^k et pour marquer une arête lorsqu'elle est desservie nous conservons les variables V_v^k. Nous avons donc deux familles de variables :

$$d_{ij}^k = \begin{cases} 1 & \text{si l'arête } e = (i,j) \text{ est utilisée dans le chemin partiel } k. \\ 0 & \text{sinon.} \end{cases}$$

$$V_v^k = \begin{cases} 1 & \text{si le sommet } v \text{ est le dernier sommet du chemin partiel } k. \\ 0 & \text{sinon.} \end{cases}$$

Le modèle se complique nettement par l'ajout de nombreuses contraintes.

Le modèle

$$\min_{d_{ij}^k, V_v^k} \sum_{k=1}^{m} \sum_{(i,j)\in E} (m-k+1)\cdot c_{ij}\cdot d_{ij}^k$$

s.à :

$$\sum_{k=1}^{m} d_{ij}^k \geq 1 \qquad \forall\ (i,j)\in E \qquad \text{(D.26)}$$

$$\sum_{v\in V} V_v^k = 1 \qquad \forall\ k\in\{0,\ldots,m\} \qquad \text{(D.27)}$$

$$\sum_{p=1}^{k-1} d_{ij}^p \leq \Big(2-(d_{ij}^k+V_i^k+V_j^k)\Big)\cdot(k-1)\cdot c_{ij} \qquad \forall\ k\in\{2,\ldots,m\},\ \forall\ (i,j)\in E \qquad \text{(D.28)}$$

$$2V_v^k \leq \sum_{e\in E: v\in e} d_e^k + \sum_{e\in E: v\in e} d_e^{k+1} \qquad \forall\ v\in V,\ \forall\ k\in\{1,\ldots,m-1\} \qquad \text{(D.29)}$$

$$V_v^0 \leq \sum_{e\in E: v\in e} d_e^1 \qquad \forall\ v\in V \qquad \text{(D.30)}$$

$$d_{ij}^k \leq V_i^{k-1} + V_i^k + \sum_{(p.q)\in E\setminus\{(i,j)\}} d_{pq}^k \qquad \forall\ k\in\{1,\ldots,m\},\ \forall\ (i,j)\in E \qquad \text{(D.31)}$$

$$d_{ij}^k \leq V_j^{k-1} + V_j^k + \sum_{(p.q)\in E\setminus\{(i,j)\}} d_{pq}^k \qquad \forall\ k\in\{1,\ldots,m\},\ \forall\ (i,j)\in E \qquad \text{(D.32)}$$

$$d_{ij}^k \leq \sum_{p=1}^{k-1} + V_i^k + V_j^k \qquad \forall\ k\in\{2,\ldots,m\},\ \forall\ (i,j)\in E \qquad \text{(D.33)}$$

$$V_i^0 = 1 \qquad i \text{ est le dépôt} \qquad \text{(D.34)}$$

et les toutes les variables d_e^k et V_v^k sont binaires.

(D.26) force le passage et donc le service sur toutes les arêtes alors que (D.27) et (D.28) obligent le service d'une arête par période k. (D.29)-(D.32) font le liens entre les deux familles de variables. (D.33) n'est pas nécessaire mais permet d'obtenir une meilleure borne inférieure lorsque nous relaxons les conditions d'intégralité. Finalement, (D.34) désigne le dépôt.

L'exemple introductif

Pour visualiser la solution, il suffit de se reporter à la figure 1.12 page 17 et de ne conserver à l'esprit que les deux familles de variables d_e^k et V_v^k.

L13

Ce modèle est une des tentatives de réduire le nombre astronomique de variables du modèle L8. Une façon de faire est de transférer la construction d'une solution réalisable à des variables sur les arêtes à desservir (x_{ij}^k) et non plus des plus courts chemins. Pour lier entre elles de telles arêtes, nous utlisons un nombre restreint de variables (y_{ij}). Nous reprenons la nomenclature du modèle L8.

Soit $Q = \max_{(i,j)\in E} c_{ij}$ le coût le plus élevé sur toutes les arêtes du graphe original.

$$x_{ij}^k = \begin{cases} 1 & \text{si on dessert l'arête } (i,j) \in \overline{A}_r \text{ à la période } k. \\ 0 & \text{sinon.} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si on utilise le plus court chemin } i \to j \text{ à la période } k. \\ 0 & \text{sinon.} \end{cases}$$

$z^k \in \mathbb{Z}^+$ est le coût du diamètre emprûnté à la période k.

Notons que seules les variables x_{ij}^k sont indépendantes.

Le modèle

$$\min_{x_{ij}^k} \sum_{k=1}^{m} (m-k+1) \cdot \left[\sum_{(i,j)\in\overline{A}_r} c_{ij} \cdot x_{ij}^k + z^k \right]$$

s.à :

$$\sum_{k=1}^{m} (x_{ij}^k + x_{ji}^k) = 1 \qquad \forall\ (i,j) \in \overline{E}_r \qquad \text{(D.35)}$$

$$\sum_{(i,j)\in\overline{E}_r} (x_{ij}^k + x_{ji}^k) = 1 \qquad \forall\ k \in \{1,\ldots,m\} \qquad \text{(D.36)}$$

$$\sum_{(p,v)\in\overline{A}_{sp}} y_{pv} + \sum_{(v,q)\in\overline{A}_{sp}} y_{vq} = 1 \qquad \forall\ v \in \overline{V} \qquad \text{(D.37)}$$

$$\sum_{v\in\overline{V}} y_{sv} = 1 \text{ et } \sum_{v\in\overline{V}} y_{vt} = 1 \qquad \text{(D.38)}$$

$$x_{\overline{i}i}^k + x_{j\overline{j}}^{k+1} \leq 1 + y_{ij} \qquad \forall\ (ij) \in \overline{A}_{sp},\ \forall\ k \in \{1,\ldots,m\} \qquad \text{(D.39)}$$

$$x_{ij}^1 \leq y_{si} \qquad \forall\ (ij) \in \overline{A}_{sp} : *i \text{ est le dépôt} \qquad \text{(D.40)}$$

$$x_{ij}^m \leq y_{it} + y_{jt} \qquad \forall\ (ij) \in \overline{A}_{sp} \qquad \text{(D.41)}$$

$$(x_{ij}^k - 1)Q + y_{ip}c_{ip} \leq z^k \qquad \forall\ (ij) \in \overline{A}_r,\ \forall (p,i) \in \overline{A}_{sp},\ \forall\ k \in \{2,\ldots,m\} \qquad \text{(D.42)}$$

et les variables x_{ij}^k et y_{ij} sont binaires et les variables $z^k \in \mathbb{Z}^+$ sont entières positives ou nulles.

(D.35) impose que toute arête soit desservie et (D.36) qu'à chaque période une seule arête soit desservie. Pour obtenir un chemin nous utilisons plusieurs contraintes : (D.37) et (D.38) permettent que chaque sommet soit relié par un plus court chemin, (D.39), (D.40) et (D.41) font le lien entre les variables x_{ij}^k et y_{ij}. Finalement, (D.42) permet de calculer le coût des différents plus court chemin utilisés.

L'exemple introductif

Pour mieux comprendre le modèle, utilisons-le pour résoudre le problème introductif de la section 1.3 page 8.

La construction de la solution optimale par CPLEX est résumée dans la figure D.5.

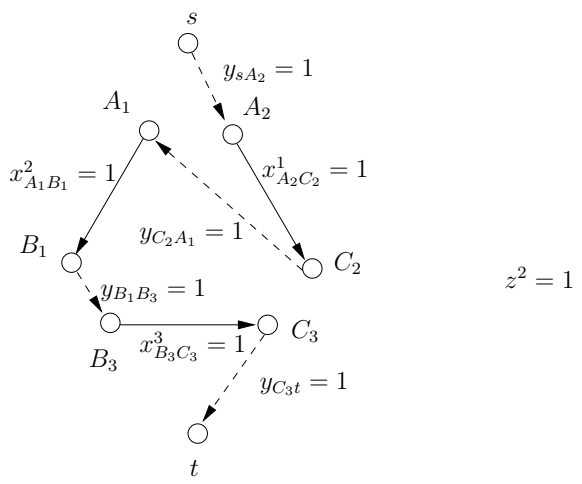


FIGURE D.5 Une solution optimale de l'exemple introductif pour le modèle L13. Toutes les variables non nulles sont affichées.

D.1.2 Comparaisons sommaires des modèles non présentés

Nous commençons par reprendre le nombre de variables ainsi que de contraintes des modèles repris dans cette annexe. Le tableau D.1 est l'équivalent du tableau 4.1 page 70 concernant les huit modèles présentés dans la thèse.

TABLEAU D.1 Bornes supérieures[a b] sur le nombre de variables et de contraintes pour chacun des modèles repris dans cette annexe

Modèles	**#Variables**	**#Contraintes**
Q1	$m^3 + \frac{3}{2}m^2 + \frac{m}{2}$	$m^3 + m^2n + 2m^2 + mn + 2m - n + 1$
L9	$m^3 + \frac{3}{2}m^2 + \frac{m}{2}$	$m^3 + m^2n + 3m^2 + mn + 3m - n + 1$
L10	$4m^2 + m$	$4m^2 + 2m + 3$
L11	$4m^2 + 4m + 2$	$2m^2 + 5m + 6$
L12	$m^2 + mn$	$4m^2 + mn + 2$
L13	$6m^2 - 3m$	$4m^3 - 8m^2 + 11m + n + 2$

[a] Parfois certaines contraintes ou variables facilement éliminables par un pré-traitement sommaire n'ont pas été comptabilisées.
[b] Nous avons systématiquement assumé que le dépôt était de degré n.

Le tableau D.2 reprend les bornes inférieures obtenues par relaxation des conditions d'intégralité pour les modèles repris dans l'annexe. Nous ne considérons que les cinq premiers

graphes de référence et nous avons exclu le modèle Q1 qui a un comportement similaire au modèle L9[3]. La deuxième ligne donne les valeurs optimales du PPCC alors que la troisième ligne donne les moins bonnes bornes pour les huit modèles présentés dans la thèse.

Tableau D.2 Comparaisons des optimums obtenus par relaxation des contraintes d'intégralité pour les différents modèles repris dans l'annexe. Les valeurs en gras représentent des valeurs optimales pour le PPCC. La ligne $\min_{L1-L8} \underline{z}$ correspond aux moins bonnes bornes obtenues pour les modèles L1-L8.

Graphes	*g1*	*g2*	*g3*	*g4*	*g5*
opt	**57**	**146**	**58**	**12591503**	**32**
$\min_{L1-L8} \underline{z}$	55	112.03	55	12345678	31.0028
L9	0	0	0	0	0
L10	26	42,5	5	05555555,5	8,5
L11	29	55	20,5	05555627,0	16,5
L12	55	104,4	38	11616769,05	27
L13	55	112	55	12345678,0	31

3. La valeur optimale du programme linéaire obtenu par relaxation des conditions d'intégralité n'est pas forcément 0 pour les modèles L9 ou Q1.