

Université de Montréal

Le problème du postier chinois cumulatif

par
Nikolaj van Omme

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique



décembre, 2003

©, Nikolaj van Omme, 2003

QA

76

U54

2004

v.010

AVIS

L'auteur a autorisé l'Université de Montréal à reproduire et diffuser, en totalité ou en partie, par quelque moyen que ce soit et sur quelque support que ce soit, et exclusivement à des fins non lucratives d'enseignement et de recherche, des copies de ce mémoire ou de cette thèse.

L'auteur et les coauteurs le cas échéant conservent la propriété du droit d'auteur et des droits moraux qui protègent ce document. Ni la thèse ou le mémoire, ni des extraits substantiels de ce document, ne doivent être imprimés ou autrement reproduits sans l'autorisation de l'auteur.

Afin de se conformer à la Loi canadienne sur la protection des renseignements personnels, quelques formulaires secondaires, coordonnées ou signatures intégrées au texte ont pu être enlevés de ce document. Bien que cela ait pu affecter la pagination, il n'y a aucun contenu manquant.

NOTICE

The author of this thesis or dissertation has granted a nonexclusive license allowing Université de Montréal to reproduce and publish the document, in part or in whole, and in any format, solely for noncommercial educational and research purposes.

The author and co-authors if applicable retain copyright ownership and moral rights in this document. Neither the whole thesis or dissertation, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms, contact information or signatures may have been removed from the document. While this may affect the document page count, it does not represent any loss of content from the document.

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:
Le problème du postier chinois cumulatif

présenté par:
Nikolaj van Omme

a été évalué par un jury composé des personnes suivantes:

Jean-Yves Potvin

(président-rapporteur)

Michel Gendreau

(directeur de recherche)

Patrick Soriano

(co-directeur)

Jean-Marc Rousseau

(co-directeur)

Bernard Gendron

(membre du jury)

Mémoire accepté le 14 avril 2004

Résumé

Le sujet de ce mémoire 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.

Ce problème est, à notre connaissance, nouveau et nous en faisons une première étude théorique. Le lecteur ne trouvera donc pas de résultats expérimentaux.

Notre contribution est de plusieurs ordres. Premièrement nous définissons le problème. Comme expliqué dans ce mémoire, la définition retenue est en rupture avec la tradition des problèmes du postier chinois. Deuxièmement, nous proposons une modélisation mathématique et démontrons la NP-difficulté du problème. Troisièmement, deux approches exactes pour résoudre le problème sont proposées. D'une part, le caractère cumulatif nous permet une énumération implicite des solutions et d'autre part, le modèle peut se résoudre par une méthode de séparation et évaluation progressive. Quatrièmement, nous développons des bornes inférieures ainsi qu'une heuristique. Cinquièmement, divers cas résolubles en temps polynomial sont exhibés. Trois cas sont adaptés d'algorithmes existants. Le cas de la ligne droite ainsi que du cercle sont adaptés d'un algorithme développé par Afrati, Cosmadakis, Papadimitriou et Papakostantinou, alors que la résolution du cas de l'arbre unitaire provient de l'adaptation d'un algorithme de Minieka. A notre connaissance, le cas du cercle n'a jamais été traité. Nous démontrons aussi que le cas de la bande unitaire est résoluble en temps polynomial et conjecturons que c'est aussi le cas pour la grille unitaire.

Mots clefs

Tournées sur les arcs, fonction cumulative.

Abstract

The subject of this master thesis is the cumulative chinese postman problem. We focus on the time of treatment completion of each arc. This introduces a cumulative and dynamical aspect in the objective function therefore changing the structure of the chinese postman problem.

This problem is, to our knowledge, entirely new and we only cover it theoretically. No experimental results are shown.

However, our contribution to the understanding of this problem covers several areas. First, we propose a definition of the problem. As shown in this work, this is done in a novel way that contrasts with the usual definition of chinese postman problems. Secondly, we propose a mathematical model and prove the NP-hardness of this problem. Thirdly, two exact algorithms are presented. On one side, the cumulative aspect of the problem allows us to implicitly enumerate the solutions. On the other, the proposed model is well suited for a branch-and-bound approach. Fourthly, lower bounds and a heuristic are provided. Fifthly, special cases defined on particular network structures are studied. Three polynomial cases are adapted from existing algorithms. The circle and the line are adapted from an algorithm of Afrati, Cosmadakis, Papadimitriou et Papakostantinou while the unweighted tree is adapted from an algorithm of Minieka. To our knowledge, the circle has never been treated before. We also prove that the unweighted strip has a polynomial time complexity and conjecture that this is also the case for the unweighted grid.

Keywords

Cumulative Chinese Postman Problem, Arc routing, cumulative functions.

Table des matières

Liste des figures	vii
Liste des tableaux	ix
Conventions	x
Sigles français	x
Sigles anglais	x
Conventions de notation	xi
Conventions graphiques	xi
Remerciements	xiv
Introduction	1
1 Le problème du postier chinois cumulatif (PPCC)	3
1.1 Coûts cumulatifs	3
1.2 Le problème	5
1.3 Un exemple	7
1.4 Motivations	7
1.4.1 Un problème de tournée sur les arêtes	8
1.4.2 Clients versus serveur	9
2 Un problème nouveau	11
2.1 Divergences avec le problème du postier chinois classique	11
2.2 Similitudes avec d'autres problèmes	13
2.2.1 Le problème du postier chinois	14
2.2.2 Le TSP cumulatif	16
3 Un modèle mathématique pour le PPCC	20
3.1 Idée de base du modèle et définition des variables de décision	20
3.2 Les contraintes	23

3.3	Le modèle	25
3.4	Un exemple	26
4	Complexité du PPCC et une première approche de résolution du problème	28
4.1	Le PPCC est NP-difficile	28
4.2	Approches exactes	31
4.2.1	Première approche (énumération implicite)	31
4.2.2	Deuxième approche (méthode de séparation et évaluation progressive)	33
4.3	Une heuristique	37
4.4	Bornes inférieures	37
4.4.1	Première borne (borne algébrique)	39
4.4.2	Deuxième borne (relaxation du modèle)	40
5	Cas polynomiaux adaptés	41
5.1	Le cercle	41
5.1.1	Idée générale et principe d'optimalité	41
5.1.2	Implémentation	43
5.2	La ligne droite	46
5.3	L'arbre avec coût unitaire	48
6	La grille unitaire	52
6.1	La bande unitaire	52
6.2	La grille unitaire	56
6.3	Importance de la localisation du dépôt	59
6.3.1	La bande unitaire	59
6.3.2	La grille unitaire	62
	Conclusions	63
	Bibliographie	67
	Annexes	xv
	Modèle linéaire du PPCC de l'exemple simple	xvi
	Solution de CPLEX pour l'exemple simple	xx

Liste des figures

1.1	Temps de fin de service d'une arête.	4
1.2	Le caractère cumulatif du problème.	5
1.3	Un exemple simple.	7
1.4	Une solution optimale de l'exemple simple.	8
2.1	Un tour eulérien ne constitue pas une solution optimale.	12
2.2	Une arête peut être copiée plusieurs fois dans le graphe augmenté.	12
2.3	La solution optimale n'est pas plane.	13
3.1	La construction de la solution optimale.	20
3.2	Un chemin partiel.	21
3.3	Un diamètre.	21
3.4	Une solution détaillée.	27
4.1	Différence entre le MLP et le PPCC.	29
4.2	Transformation du MLP en PPCC.	30
4.3	Un exemple pour illustrer l'approche énumérative.	32
4.4	L'arbre de recherche de la méthode de séparation et évaluation progressive.	36
4.5	L'heuristique.	38
5.1	Numérotation des sommets du cercle.	42
5.2	Une solution optimale est composée d'arcs de cercle.	43
5.3	Comment former un arc de cercle gauche.	44
5.4	Une ligne droite et sa solution optimale.	47
5.5	Une recherche en profondeur ne donne pas une solution optimale dans le cas général.	50
5.6	Pour l'arbre unitaire, les recherches en profondeur donnent le même résultat.	50
5.7	Le calcul des coûts cumulatifs pour deux recherches en profondeur.	51

6.1	Une bande unitaire.	52
6.2	Deux plus longs chemins simples pour la bande unitaire.	53
6.3	L'arbre des possibilités pour trouver les chemins simples les plus longs.	54
6.4	Les deux solutions optimales pour la bande unitaire.	54
6.5	Augmentation minimale pour rendre les sommets de degré pair.	55
6.6	Une grille unitaire.	57
6.7	Un exemple de grille unitaire dont la solution optimale ne commence pas par un chemin simple le plus long.	57
6.8	La solution optimale pour la grille 3×3	58
6.9	Comment remplir optimalement l'intérieur d'une grille.	58
6.10	Notre conjecture pour la grille unitaire.	60
6.11	Le cas de la grille unitaire 2×2	61
6.12	Une solution optimale quand le dépôt est au deuxième sommet.	61
6.13	Une solution optimale quand le dépôt est au troisième sommet.	62

Liste des tableaux

I	Sigles français utilisés dans ce mémoire.	x
II	Sigles anglais utilisés dans ce mémoire.	xi
III	Conventions de notation utilisées dans ce mémoire.	xii
IV	Conventions graphiques	xiii
I	Les différentes solutions pour les deux problèmes MLP et PPCC de la figure 4.1.	29
II	Les différentes solutions parcourues par l'approche d'énumération implicite.	34
III	Les différents temps de résolution par CPLEX de la borne inférieure.	40
I	Le calcul de $L(i,j)$ pour l'exemple de la figure 5.4.	48
II	Le calcul de $R(i,j)$ pour l'exemple de la figure 5.4.	48
III	Le calcul de $f(i,j,1)$ pour l'exemple de la figure 5.4.	49
IV	Le calcul de $f(i,j,0)$ pour l'exemple de la figure 5.4.	49
I	Description des termes de la borne inférieure pour la bande unitaire.	56
II	La conversion entre les noms des arêtes.	xvi

Conventions

Tout au long de ce mémoire plusieurs conventions sont respectées à moins d'indications contraires explicites. Ces conventions sont principalement de deux ordres : d'une part les sigles utilisés et d'autre part les conventions graphiques et de notation.

Sigles français

Nous n'utiliserons les sigles français que pour désigner certains problèmes du postier chinois et en particulier celui de PPCC pour le problème du postier chinois cumulatif. Le tableau I résume les règles utilisées.

PPCC	Problème du Postier Chinois Cumulatif.
PPC	Problème du Postier Chinois.
PPCA	Problème du Postier Chinois asymétrique.
PPR	Problème du Postier Rural.
PPCH	Problème du Postier Chinois Hiérarchique.

TAB. I – *Sigles français utilisés dans ce mémoire.*

Sigles anglais

Nous utilisons des sigles anglais dans un texte français. Ceci peut paraître inapproprié, mais nous nous soumettons à la pratique des chercheurs qui utilisent couramment les expressions anglophones. Si certaines appellations en français comme le *PVC* (Problème du voyageur de commerce) sont courantes et bien comprises de tous, d'autres, comme *PRI* pour le *Problème du réparateur itinérant*, le sont nettement moins alors que leur équivalent anglais (*Traveling Repairman Problem* ou *TRP*) est généralement compris et accepté de la communauté scientifique francophone.

Enfin, il y a le problème du sigle *PPCC* qui est aussi utilisé pour le *Problème du postier chinois avec capacité*. Pour lever toute ambiguïté, nous n'utilisons ce dernier sigle que pour le sujet de ce mémoire. Le tableau II rassemble ces sigles anglais.

CCCP	Capacitated Chinese Postman Problem (Problème du postier chinois avec capacité).
CTSP	Cumulative Traveling Saleman Problem (Problème du voyageur de commerce cumulatif).
DMP	Delivery Man Problem (Problème du livreur).
MLP	Minimum Latency Problem (Problème de latence minimum).
k-MST	k-Minimum Spanning Tree (Problème de l'arbre couvrant à k sommets).
TRP	Traveling Repairman Problem (Problème du réparateur itinérant).
TSP	Traveling Salesman Problem (Problème du voyageur de commerce).

TAB. II – *Sigles anglais utilisés dans ce mémoire.*

Conventions de notation




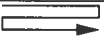
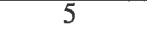


Les conventions de notation sont rassemblées dans le tableau III.

Conventions graphiques

Le tableau IV présente les conventions graphiques utilisées dans les figures tout au long de ce mémoire.

A, B, C	Des lettres latines majuscules sont utilisées pour désigner les sommets d'un graphe.
a, b, c	Des lettres latines minuscules sont utilisées pour désigner les arêtes d'un graphe.
A	L'ensemble des arêtes du graphe.
n	Le nombre d'arêtes du graphe.
S	L'ensemble des sommets du graphe.
p	Le nombre de sommets du graphe.
G	Le graphe lui-même.
a_i	L'arête desservie en i ème place lors d'une tournée.
S_i	Le sommet visité en i ème position dans un MLP.
$c(a_i)$	Le coût de l'arête a_i .
$c(i, i + 1)$	Le coût de l'arête désignée par ses sommets i et $i + 1$.
$\text{coût}(a_{i-1}, a_i)$	La somme des coûts des arêtes permettant de rejoindre l'arête a_i à partir de l'arête a_{i-1} dans la tournée du PPCC.
$t(S_i, S_j)$	La somme des coûts des arêtes permettant de rejoindre le sommet S_i au sommet S_j dans la tournée du MLP.
$\text{coût cumulatif}(a_i)$	Le coût cumulatif de l'arête a_i .
$T(S_i)$	La latence du sommet S_i dans la solution du MLP.
$A - B - C$	La tournée ou le chemin passant par les sommets A, B et C .
$a - b - c$	La tournée ou le chemin desservant les arêtes a, b et c dans cet ordre s'il n'y a pas de répétition des arêtes, sinon le chemin ou la tournée indiqués.
V_S^j	Variable indiquant si le sommet S est le dernier sommet du chemin partiel j .
D_S^j	Variable indiquant si le diamètre du chemin partiel j traverse le sommet S .
v_a^j	Variable indiquant si l'arête a est desservie dans le chemin partiel j .
d_a^j	Variable indiquant si l'arête a appartient au diamètre du chemin partiel j .
$L(i, j)$	La valeur optimale pour couvrir par la gauche toutes les arêtes situées entre les sommets i et j dans la récursion avant de l'algorithme dynamique proposée au chapitre 5 pour résoudre la ligne droite et le cercle.
$R(i, j)$	L'équivalent de $L(i, j)$ mais pour une couverture par la droite.
$f(i, j, k)$	L'équivalent pour la récursion arrière de $L(i, j)$ si $k = 1$ et de $R(i, j)$ si $k = 0$.
$d(i, j)$	Le coût du diamètre entre les sommets i et j , c'est-à-dire la somme des coûts des arêtes reliant le sommet i au sommet j .
$p(a_i)$	La profondeur d'une arête dans un arbre.
∞	un coût infini est un coût très grand qui soit est donné explicitement soit peut aisément se déduire du contexte.

TAB. III – Conventions de notation utilisées dans ce mémoire.

Symboles	Descriptions	Significations
	Sommet carré dans un graphe	Désigne le dépôt d'où commence la tournée.
	Arête en trait plein terminée par une flèche	Indique une arête desservie dans la tournée.
	Arête en traits pointillés terminée par une flèche.	Indique une arête parcourue mais non desservie dans la tournée.
	Ligne en trait plein terminée par une flèche	Indique une partie de la tournée.
	Nombre à côté d'une arête	Coût de l'arête.
	Nombre encadré à côté d'une arête.	Coût cumulatif de l'arête.
	Nombre entouré à côté d'une arête.	Coût relatif de l'arête.

TAB. IV – Conventions graphiques

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 ce mémoire.

J'aimerais remercier 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 enquéri de mon état.

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 général m'ont fortement inspiré.

Mademoiselle Maria 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.

Ces remerciements n'auraient aucun sens si je ne mentionnais Madame Catherine de Borghrave et Monsieur Albert Carton. Leur amitié indéfectible depuis maintenant 12 ans est 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.

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

Finalement, je remercie mademoiselle Cathy Renaud pour son soutien au cours des dernières semaines de rédaction de ce mémoire. J'espère pouvoir l'encourager autant dans ses études qu'elle ne l'a fait dans les miennes.

Introduction

Le problème du postier chinois (PCC) est bien connu et résolu dans sa version classique [9]. Petit à petit, des contraintes supplémentaires sont venues s'ajouter pour obtenir une meilleure modélisation du réel. Ainsi, nous avons le problème du postier chinois asymétrique (PPCA) où l'arête peut être traversée différemment 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 chinois hiérarchique (PPCH) où des classes d'arêtes sont desservies prioritairement à d'autres.

Ces problèmes posent encore de fameux défis mais dans ce mémoire, nous nous tournons vers un autre genre de modifications de la version classique du postier chinois. Nous n'ajoutons aucune contrainte mais nous considérons une nouvelle fonction objectif. La particularité de cette fonction est son caractère cumulatif qui donne son nom au problème : le problème du postier chinois cumulatif (PPCC). Ce problème est nouveau et à notre connaissance il n'existe aucun article sur le sujet.

Notre motivation n'était pas tant de modifier la fonction objectif que d'aborder 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. Comme nous le verrons dans ce mémoire, le PPCC est un bon compromis entre les attentes des clients et ce que peut offrir le fournisseur. Dans le PPCC nous essayons de minimiser le temps d'attente total des clients, ce qui introduit le caractère cumulatif dans la fonction objectif.

Notre étude se veut une première approche du problème. Avant de commencer ce mémoire, nous espérons pouvoir répondre aux questions suivantes :

- Quelle est la complexité du problème?
- Comment pouvons-nous le modéliser?
- Peut-on développer un algorithme exact pour le cas général?
- Si le problème est NP-difficile, peut-on identifier des cas polynomiaux?

Nous avons pu donner un début de réponse à toutes ces questions dans ce mémoire. Nous ne présentons que les idées, laissant de côté les questions d'implémentations in-

formatiques. Le lecteur ne doit donc pas s'attendre à trouver du code ou des résultats expérimentaux. Par contre, plusieurs théorèmes avec démonstration sont proposés.

Ce mémoire comporte six chapitres. Le premier décrit et formule le PPCC. Bien que le PPCC soit nouveau, il existe un problème fort similaire, le problème du commis voyageur cumulatif qui est l'équivalent sur les nœuds de notre PPCC sur les arêtes. Pour certains graphes, tels l'arbre ou la ligne droite, ces deux problèmes sont quasi identiques. Le deuxième chapitre passe en revue ce qui a déjà été fait pour les problèmes du postier chinois ainsi que ce qui a été développé pour le problème du commis voyageur cumulatif. Notre modèle mathématique du PPCC est présenté dans le troisième chapitre. Le quatrième chapitre propose l'étude de la complexité du PPCC ainsi que les outils que nous avons développés pour ce problème: deux algorithmes exacts pour le résoudre, deux bornes inférieures et une heuristique. A partir d'algorithmes pour résoudre le problème du voyageur de commerce cumulatif, trois cas polynomiaux ont été adaptés dans le cinquième chapitre. Dans le sixième chapitre nous étudions le cas particulier de la grille unitaire. Nous démontrons que la bande unitaire se résout en temps polynomial et conjecturons que c'est aussi le cas de la grille unitaire. Finalement, nous terminons par nos conclusions et décrivons quelques perspectives de recherche future.

Chapitre 1

Le problème du postier chinois cumulatif (PPCC)

Nous introduisons d'abord le problème en procédant par étapes puis, nous motivons son étude ainsi que sa définition.

Dans la première section, nous définissons ce que nous entendons par un coût cumulatif aussi appelé coût réel ou coût effectif. Au passage, nous en profitons pour définir ce que sont les coûts relatifs. La deuxième section nous permet de définir rigoureusement le problème du postier chinois cumulatif (PPCC). Nous illustrons ce dernier par un exemple dans la troisième section.

La dernière et quatrième section est entièrement consacrée à la motivation du problème. Celle-ci est double. D'une part, le problème du postier chinois cumulatif est intéressant parce qu'il privilégie le point de vue des clients (modélisés par les arêtes) contrairement aux problèmes du postier chinois classiques qui ne se préoccupent que du point de vue du fournisseur de services. En fait, le PPCC est un bon compromis entre les intérêts des clients et ceux du fournisseur. D'autre part, nous montrons que le PPCC introduit un aspect égalitaire dans le traitement des clients, ce qui le différencie nettement du problème du postier chinois classique où aucune distinction entre les clients n'est faite.

1.1 Coûts cumulatifs

Soit une tournée passant par toutes les arêtes et traitant celles-ci dans l'ordre $a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n$. Comme le décrit la figure 1.1, 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 temps de service. 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 : *coût réel* et *coût effectif*. En anglais, on parle de *latence* de l'arête, terme qui pourrait être repris en français aussi.

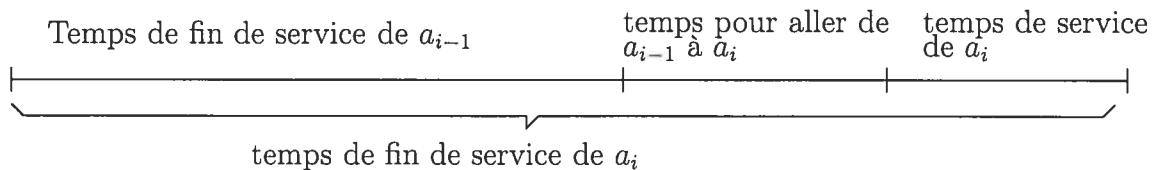


FIG. 1.1 – Temps de fin de service d'une arête.

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 qu'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.

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.

Tout au long de ce mémoire, nous respectons la convention graphique suivante : un nombre à côté d'une arête représente son coût, un nombre encadré à côté d'une arête indique son coût cumulatif et finalement un nombre entouré à côté d'une arête dénote son coût relatif. Ces conventions et bien d'autres encore se retrouvent à la section et suivantes sur les conventions au début de ce mémoire.

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 1.2. Le coût cumulatif d'une arête résulte de l'addition répétée des coûts des arêtes desservies précédemment.

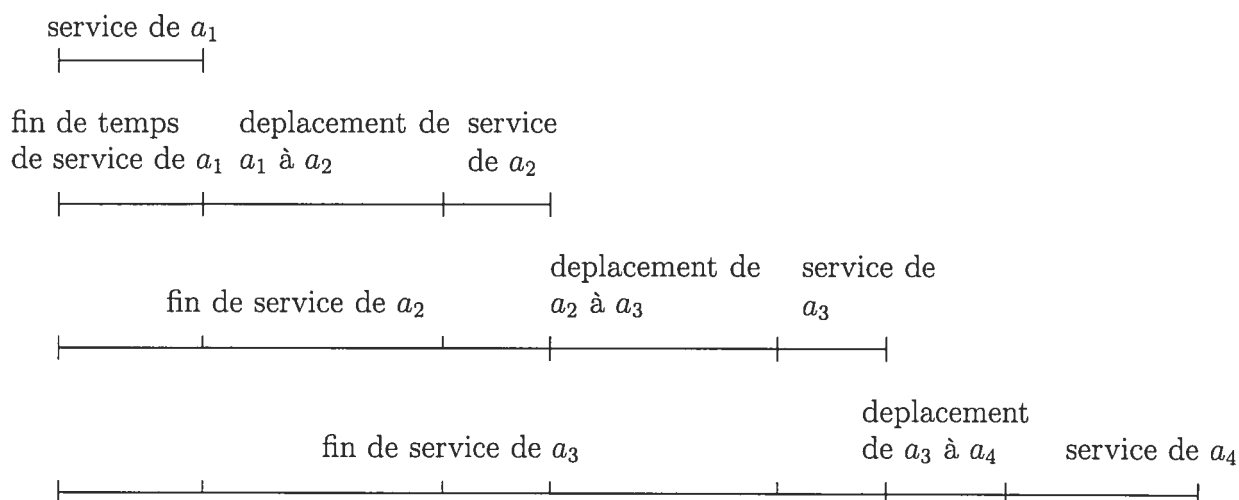


FIG. 1.2 – Le caractère cumulatif du problème.

L'aspect cumulatif provient de l'accumulation des coûts. Par exemple le coût associé au service de la première arête se retrouve compté n fois dans la valeur de la fonction objectif.

1.2 Le problème

Dans un premier temps et dans ce mémoire, nous nous intéressons au problème non orienté et 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. Bien que cela ne soit pas réaliste (et le problème où la dissociation est faite mérite le détour), cette hypothèse correspond à notre souci de privilégier le temps de fin de service global. Nous réfutons ainsi la possibilité de pouvoir passer sur une arête sans la servir puisque le passage est aussi coûteux que le service.

Définissons le problème.

Soit $G = (\mathbf{S}, \mathbf{A})$ un graphe connexe non orienté où \mathbf{S} est l'ensemble des p sommets et \mathbf{A} l'ensemble des n arêtes. Soit une fonction c de coûts sur les arêtes $c : \mathbf{A} \rightarrow \mathbb{N}$. Nous ne considérons que des coûts entiers positifs.

Un sommet privilégié est choisi dès le départ, c'est le sommet à partir duquel la tournée est effectuée. (Il serait intéressant aussi d'étudier le problème de rechercher à partir de quel sommet la tournée est la moins coûteuse.)

Le coût d'une arête au sens cumulatif dépend du chemin parcouru. Pour calculer le coût cumulatif d'une arête a_i on additionne le coût cumulatif de l'arête précédente a_{i-1} avec le coût pour aller de a_{i-1} à a_i plus le coût $c(a_i)$ de a_i :

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

Nous voyons que le coût relatif d'une arête est simplement le coût de déplacement plus le coût de traitement de cette arête :

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

A tout chemin commençant par le sommet privilégié passant par les n arêtes a_i du graphe dans l'ordre i_1, i_2, \dots, i_n , la valeur de la fonction objectif* z est la somme de tous les coûts cumulatifs pour toutes les arêtes :

$$z = \text{coût cumulatif}(a_1) + \text{coût cumulatif}(a_2) + \dots + \text{coût cumulatif}(a_n).$$

De sorte que

$$z = \begin{array}{l} c(a_1) + \\ c(a_1) + \text{coût}(a_1, a_2) + c(a_2) + \\ c(a_1) + \text{coût}(a_1, a_2) + c(a_2) + \text{coût}(a_2, a_3) + c(a_3) + \\ \dots \end{array} \left| \begin{array}{l} \text{coût cumulatif}(a_1) \\ \text{coût cumulatif}(a_2) \\ \text{coût cumulatif}(a_3) \\ \dots \end{array} \right.$$

ou encore

$$z = \begin{array}{l} n \cdot c(a_1) + \\ (n-1) \cdot \text{coût}(a_1, a_2) + (n-1) \cdot c(a_2) + \\ (n-2) \cdot \text{coût}(a_2, a_3) + (n-2) \cdot c(a_3) + \\ \vdots \\ (n-i) \cdot \text{coût}(a_i, a_{i+1}) + (n-i) \cdot c(a_{i+1}) + \\ \vdots \\ (1) \cdot \text{coût}(a_{n-1}, a_n) + (1) \cdot c(a_n) \end{array}$$

Ce qu'on peut résumer en

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

Définition 1.2.1 *Le problème du postier chinois cumulatif consiste à trouver une tournée qui, partant d'un sommet privilégié (le dépôt), desserve toutes les arêtes du graphe et ce au coût minimal au sens cumulatif.*

* Nous avons remplacé les indices i_1, i_2, \dots, i_n par $1, 2, \dots, n$ pour simplifier l'écriture.

Même si nous parlons de tournée, que le lecteur ne soit pas induit en erreur. Nous n'exigeons pas le retour au dépôt après que la dernière arête ait été desservie. D'une part, nous ne nous intéressons qu'au fait que les arêtes soient desservies. Une fois la tournée effectuée, nous ne nous intéressons pas à un éventuel retour au dépôt. Nous reviendrons plusieurs fois sur cet aspect de la définition. D'autre part, cette façon de faire n'est en rien restrictive. Il est aisé de construire un PPCC qui force le retour au dépôt. Ce dernier point est traité en détails dans la section 4.1.

1.3 Un exemple

La figure 1.3 représente un graphe avec trois arêtes a , b et c de coût respectif 1, 1 et 50 (on peut penser à une durée de service sur ces arêtes). Une solution optimale est représentée sur la figure 1.4. 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

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

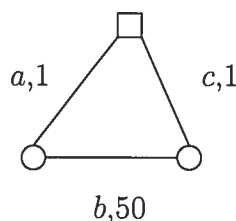


FIG. 1.3 – Un exemple simple.

Un exemple de graphe tout simple qui pourtant 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 l'arête correspondante.

1.4 Motivations

Le problème du postier chinois cumulatif est un problème nouveau. Dès lors, nous avons une relative liberté pour le définir à notre convenance. La définition que nous avons vue est en rupture avec la tradition des problèmes du postier chinois. En effet,

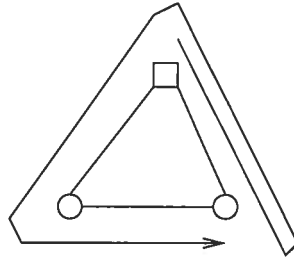


FIG. 1.4 – Une solution optimale de l'exemple simple.

Une solution optimale pour le PPCC de la figure 1.3

nous n'exigeons plus le retour au dépôt une fois toutes les arêtes desservies. Il n'est plus question de trouver un circuit eulérien mais plutôt un chemin eulérien. Cette façon de faire n'est pas restrictive (bien au contraire) comme nous le verrons tout au long de ce mémoire.

Dans les deux sous-sections qui suivent, nous motivons notre définition du PPCC et expliquons en quoi l'approche cumulative favorise le point de vue du client. Premièrement, c'est le temps d'attente total de chacun des clients qui est pris en compte et non plus celui du fournisseur de services. Deuxièmement, notre version du postier chinois est plus égalitaire dans le sens que des clients exigeant moins de temps de service seront servis avant un client demandant plus de temps pour le servir. Dans les versions précédentes du postier chinois, ce client important pouvait mobiliser la plus grande partie du temps de la tournée avant que les petits clients ne soient servis.

1.4.1 Un problème de tournée sur les arêtes

Comme dans le cas du problème du postier chinois, il est question de desservir toutes les arêtes d'un graphe. Il est donc bien question d'un problème de postier chinois. Dans un souci de simplicité, nous nous concentrons uniquement sur le cas où toutes les arêtes du graphe doivent être desservies mais une version rurale de notre problème, c'est-à-dire où seulement quelques arêtes doivent être desservies et non l'ensemble du graphe, serait à étudier ultérieurement.

Toutes les applications connues de service sur les arêtes sont donc d'application pour notre version cumulative. Citons entre autres, le déblaiement de la neige dans les rues d'une ville, la distribution du courrier, le ramassage des ordures ménagères, l'épandage de déglçants ou d'abrasifs sur les trottoirs et les chaussées en périodes hivernales, etc.

Dans les cas où il y a distribution, l'aspect cumulatif traduit le coût plus élevé au début de la tournée pour desservir les premières arêtes. Dans le cas de la distribution

de courrier on peut imaginer le fourgon postal plus chargé au début de la tournée et se déplaçant pour un coût plus élevé à pleine charge que vide. Même chose pour l'épandage de déplaçants ou d'abrasifs sur les routes.

Dans le cas où il y a enlèvement, nous ne pouvons plus invoquer aussi facilement un coût qui diminuerait avec le trajet. L'avantage des coûts cumulatifs réside alors dans l'aspect égalitaire et la prise en considération des clients comme expliqué à la sous-section suivante.

Toujours dans un souci de simplicité, les arêtes n'auront qu'un seul coût indiquant le temps (ou le coût financier, la difficulté ou quoi que ce soit d'autre) nécessaire pour (des)servir l'arête, c'est-à-dire que nous ne faisons pas de distinction entre le temps de service et le temps de passage d'une arête. C'est l'approche traditionnelle du problème du postier chinois mais qui ne reflète pas toujours la réalité. S'il s'agit de déneigement par exemple, on peut facilement envisager trois coûts différents par arête. La charrue passera plus rapidement et pour un coût moindre par une rue déjà déneigée, puis un peu plus lentement (coût supérieur) pour parcourir une rue enneigée mais sans la *traiter*, et enfin, le temps nécessaire et le coût le plus élevé correspondent évidemment au service (déneigement) lui-même.

Nous l'avons déjà dit et répété mais ce point est tellement essentiel que nous nous permettons d'encore insister. Contrairement au problème classique où le retour au dépôt est exigé, nous ne nous occupons que du fait que toutes les arêtes soient desservies. Ce qui se passe une fois la dernière arête servie ne nous concerne pas. On peut dire que le retour au dépôt a un coût de zéro. La solution optimale consiste dès lors en un chemin et plus en un tour. Cette version ouverte (pas de retour au dépôt) nous a semblé plus intéressante car non seulement elle traduit mieux que la version fermée (retour au dépôt) notre souci d'améliorer les temps de fin de traitement des arêtes (le retour au dépôt nous semble secondaire) mais en plus elle peut aisément se transformer en version fermée comme il est montré à la section 4.1.

1.4.2 Clients versus serveur

Dans les versions plus classiques du problème du postier chinois, l'accent est toujours mis sur le service de toutes les arêtes peu importe leur ordre de service (sauf pour le problème dans sa version hiérarchique) pourvu que la tournée se fasse dans un temps minimum pour le fournisseur de services. Dans notre problème, nous nous mettons plutôt du côté des clients. Les clients qui peuvent être desservis le plus rapidement auront plus de chance d'être desservis avant les autres. Mais il serait ridicule de vouloir servir à tout prix un client peu coûteux à servir 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 devoir parcourir de grandes distances inutilement pour servir un client peu coûteux mais lointain. Ainsi, il s'agit d'un compromis entre les considérations des clients et celles du fournisseur.

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 soient si mauvaises que nous perdrons le contrôle du robot assez vite. Si le petit 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 après s'il est encore fonctionnel.

Ce caractère *égalitaire* est si important que nous nous permettons de prendre un deuxième exemple pour l'illustrer. Prenons cette fois-ci des coûts vraiment différents les uns des autres. Par exemple, pensons à l'ordre de différents traitements des rues d'une ville par la voirie. Imaginons trois coûts 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 (qui consiste à enlever puis à remplacer la surface de roulement en asphalte ou en béton bitumeux). C'est la même équipe qui s'occupe de tout. Si l'équipe s'occupe d'abord de resurfer des rues, les citoyens attendront longtemps avant que de petits travaux de réparation soient effectués. De plus, les rues qui nécessitaient un traitement léger demanderont peut-être des soins plus importants car leurs fissures se seront peut-être entre-temps transformées en nids de poules et ces derniers en problèmes exigeant un resurfaçage. Alors que 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 correspondant à l'ordre planifié des interventions.

Chapitre 2

Un problème nouveau

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 pour le TSP est le sujet d'étude de bien des articles depuis une dizaine d'années (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 TSP n'a pas tendance à éclipser son petit frère sur les arêtes. Longtemps considéré comme plus intéressant, car sa complexité est plus grande que celle du problème du postier chinois en général, le TSP pourrait bien perdre ce statut dominant dans la version qui nous occupe.

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 TSP mais surtout complique sérieusement le problème. Nous conjecturons que la version cumulative du problème sur les arêtes est plus difficile que sur les nœuds.

Avant de passer en revue le problème classique du postier chinois et puis le problème du voyageur de commerce cumulatif, nous voyons pourquoi le paradigme de résolution du postier chinois classique ne nous sert pas à grand chose ici.

2.1 Divergences avec le problème du postier chinois classique

Comme pour le TSP cumulatif [5], le PPCC se différencie nettement du problème classique du postier chinois. Un changement local dans la structure du graphe, même minimal comme le changement du coût d'une arête, peut modifier globalement la structure d'une solution optimale.

Le paradigme du postier chinois *on recherche une augmentation minimale du*

graphe pour trouver un tour eulérien à coût minimal n'est plus d'application 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. Un tour eulérien ne constitue pas forcément une solution optimale comme nous le démontre la figure 2.1. C'est un résultat dont on pouvait se douter étant donné l'aspect cumulatif des coûts.

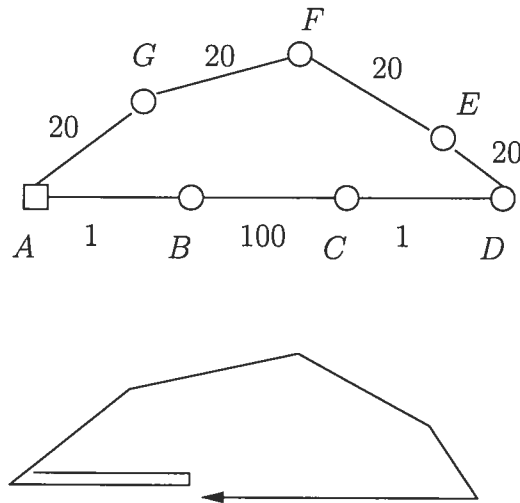


FIG. 2.1 – Un tour eulérien ne constitue pas une solution optimale.

Le tour optimal est $A - B - A - G - F - E - D - C - B$ pour un coût de 475 alors que le tour eulérien $A - B - C - D - E - F - G$ coûte 812 et le tour eulérien $A - G - F - E - D - C - B - A$ revient à 644.

Là où cela devient encore plus intéressant, c'est que le concept de graphe augmenté n'a pas l'air de nous être d'un grand secours. On peut construire aisément des problèmes où le graphe augmenté possède n copies d'une arête, ce qui contraste fortement avec la version classique du problème du postier chinois où le graphe augmenté contient au plus un dédoublement des arêtes. La figure 2.2 représente un tel problème.

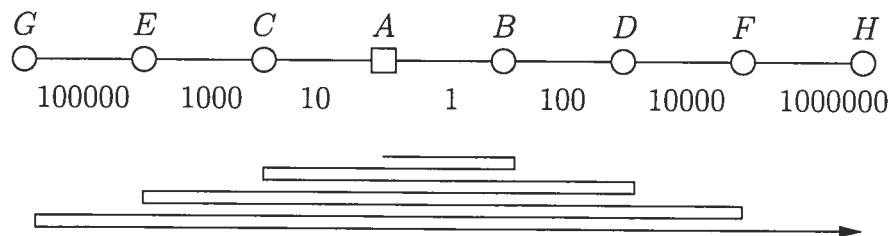


FIG. 2.2 – Une arête peut être copiée plusieurs fois dans le graphe augmenté.

Il y a sept copies de l'arête AB dans le graphe augmenté qui donne la solution optimale $A - B - C - D - E - F - G - H$.

Le coup de grâce provient du caractère global du problème. Il n'y a pas de moyen aisé de le couper en morceaux pour attaquer le problème partie par partie. Il n'est pas possible de séparer aisément un graphe en parties disjointes pour traiter chacune des parties séparément comme nous l'indique la figure 2.3 par exemple.

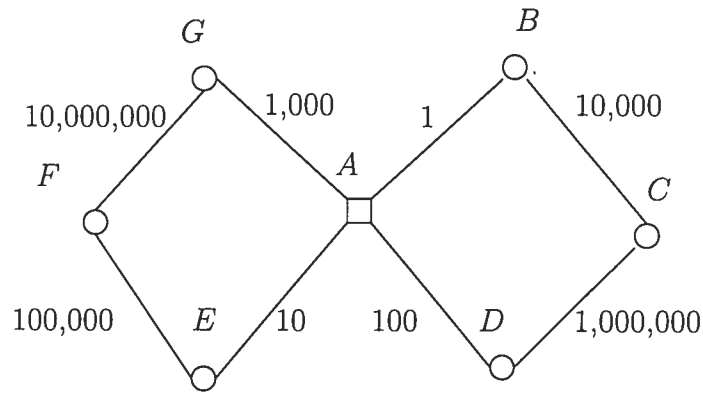


FIG. 2.3 – La solution optimale n'est pas plane.

La solution optimale consiste à parcourir les arêtes en ordre de grandeur croissant. Il n'y a pas moyen de séparer le problème en traitant séparément la boucle $A - B - C - D$ et la boucle $A - E - F - G$.

En fait, comme nous le verrons dans le chapitre 4, le PPCC est NP-difficile.

2.2 Similitudes avec d'autres problèmes

Malgré les divergences avec le problème du postier chinois classique, le PPCC reste un problème du postier chinois puisqu'il s'agit de desservir toutes les arêtes.

Nous considérons le problème du postier chinois comme étant canonique en recherche opérationnelle et nous ne retraçons pas un historique des recherches effectuées sur le sujet. Le lecteur intéressé pourra consulter le livre *Arc routing: Theory, Solutions and Applications* paru sous la direction de M. Dror et al [17]. Plus particulièrement, les chapitres 1 et 4 traitent respectivement un point de vue historique et la complexité des problèmes classiques du postier chinois. L'article *Arc Routing Problems, Part I: The Chinese Postman Problem* écrit par H.A. Eiselt, M. Gendreau et G. Laporte [9] constitue une excellente source tant sur l'aspect historique que sur les résultats et méthodes des problèmes du postier chinois. Bien que publiés en 1995, le contenu est toujours d'actualité. Nous reprenons les grandes lignes de cet article que nous résumons dans la sous-section suivante.

Le problème du postier chinois a un problème frère qui est son équivalent sur les nœuds : le problème du voyageur de commerce. Dans ce dernier problème, on s'occupe de visiter non pas les arêtes mais les nœuds pour un coût minimal.

La version cumulative sur les nœuds a déjà été l'objet de nombreux articles et nous retraçons dans les grandes lignes les apports majeurs concernant ce problème. Le lecteur intéressé pourra consulter l'article *The Minimum Latency Problem* écrit par Blum et al. [6] pour un aperçu global et *The Minimum Latency Problem Is NP-Hard For Weighted Trees* écrit par Sitters [20] pour une revue plus récente.

2.2.1 Le problème du postier chinois

Les cas non dirigé et dirigé

Les cas des graphes non dirigés et dirigés sont relativement simples. La complexité dans les deux cas est polynomiale et dans les deux cas, l'approche de résolution est la même : trouver une augmentation minimale du graphe original (c'est-à-dire dupliquer certains arcs ou arêtes) de sorte que le nouveau graphe ainsi construit soit eulérien, c'est-à-dire qu'on puisse trouver un circuit qui passe au moins une fois par tous les arcs ou arêtes.

Dans le cas non dirigé, la condition nécessaire et suffisante est que tous les sommets soient de degré pair. En effet, on peut alors entrer et sortir de n'importe quel sommet ce qui permet à la tournée de passer par toutes les arêtes puisqu'il n'y a aucun endroit où elle pourrait être arrêtée. Dans le cas dirigé, 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.

Une fois cette condition remplie, trouver effectivement un circuit eulérien ne pose pas vraiment de problème. 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 une boucle. Dans ce dernier cas, il faut recommencer à parcourir une autre boucle et fondre les deux boucles en une seule boucle plus grande et ainsi de suite. C'est l'algorithme *End-Pairing* d'Edmonds et Johnson [8]. Le cas dirigé est un peu plus compliqué mais se traite en temps polynomial aussi (algorithmes de Fleury adapté [7] ou de van Aardenne-Ehrenfest et de Bruijn [21]).

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 à coût minimum fera l'affaire.

Le cas mixte

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 compliquée 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 incidente au sommet soit parcourue dans un certain sens. 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 qui sont 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, 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 [11] 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 sortants pour tous les sommets),
- orienter les arêtes restantes et
- maintenant que le graphe est dirigé, 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. Le lecteur intéressé est renvoyé à nouveau vers l'excellent article déjà mentionné de H.A. Eiselt, M. Gendreau et G. Laporte [9] pour des informations plus détaillées.

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.

2.2.2 Le TSP cumulatif

Un problème fort similaire au PPCC est sa version nodale soit le TSP cumulatif. Ce problème est connu sous quatre vocables différents bien que le premier soit tombé en désuétude :

- TSP with cumulative costs (CTSP)
- The Minimum Latency Problem (MLP)
- The Deliveryman Problem (DMP)
- The Traveling Repairman Problem (TRP)

Voici l'énoncé* du problème librement traduit de l'article de Minieka [18] :

Soit un graphe (S, A) non orienté à p sommets et n arêtes. Soit une tournée partant du dépôt S passant par tous les sommets et revenant au dépôt S :

$$S \ S_{i_1} \ S_{i_2} \ \dots \ S_{i_k} \ \dots \ S_{i_{p-1}} \ S.$$

Chaque sommet est visité en un temps $T(S_{i_j})$ mesuré depuis le début de la tournée partant du dépôt S .

$$T(S_{i_j}) = t(S, S_{i_1}) + \sum_{k=1}^{j-1} t(S_{i_k}, S_{i_{k+1}})$$

où $t(S_{i_k}, S_{i_{k+1}})$ représente le temps nécessaire pour aller de S_{i_k} à $S_{i_{k+1}}$.

$T(S_{i_j})$ est appelé la *latence* du sommet S_{i_j} , d'où le nom *Minimum Latency Problem* du problème.

Le problème consiste à minimiser la somme des latences de tous les sommets du graphe (y compris le retour au dépôt) :

* Minieka parle de distance alors que pour garder une certaine cohérence nous utilisons la notion de temps qui est utilisé tout au long de ce mémoire.

$$\min \left\{ \sum_{j=1}^p T(S_{i_j}) \right\} \quad \text{avec } S_{i_p} = S.$$

Comme le fait remarquer Minieka, le problème du commis voyageur (TSP) ne se préoccupe que de minimiser le dernier terme de cette expression. On peut donc se douter que la version cumulative est NP-difficile, ce que Sahni et Gonzalez [19] ont démontré.

Il est à noter que certains auteurs ajoutent des poids aux sommets (notamment Sitters) dans la définition du problème.

Un exemple de MLP (sur un arbre) se trouve à la section 4.1.

Nous résumons maintenant les résultats importants sur le MLP. Nous avons subdivisé ceux-ci en quatre parties : la complexité, les approches exactes, les approximations et finalement, les cas polynomiaux.

La complexité

Le MLP est NP-difficile et cela a été démontré par Sahni et Gonzalez en 1976 [19]. Tout comme pour le TSP dans le cas général, il n'existe pas d'algorithme polynomial pour résoudre par approximation 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.

La complexité du MLP sur les arbres a longtemps été une question ouverte qui n'a été résolue qu'il y a peu de temps par Sitters [20] qui a démontré que même sur les arbres avec seulement des coûts 0 – 1 sur les arêtes, le MLP est NP-difficile. Ceci nous confirme que la fonction objectif cumulative introduit une complexité assez forte.

Les approches exactes

Il existe plusieurs approches exactes. Citons Lucena en 1990 [16] qui arrive à résoudre des instances avec au plus 30 villes, puis Bianco et al. en 1993 [5] qui montent jusqu'à 60 villes. Dans les deux cas, une méthode de séparation et évaluation progressive est appliquée. Fischetti et al utilisent un algorithme d'énumération en 1993 [10] pour résoudre des problèmes dont les plus grandes instances contiennent jusqu'à

60 sommets.

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 a été proposé par Blum et al. en 1994 [6]. Il s'agit d'un algorithme d'approximation-144. 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 pas 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 MLP sur p sommets débutant en A . Nous avons alors la définition suivante.

Définition 2.2.2.1 *Un approximateur-PVC- (α, β) est un algorithme polynomial capable de trouver, étant donné 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 [15] et donc obtiennent un algorithme d'approximation-144 pour le MLP.

Goemans et Kleinberg [14] améliorent cette approche en 1998. D'une part ils démontrent comment concaténer au mieux les sous-tours grâce à un algorithme de plus court chemin sur un graphe dont les sommets sont les ensembles de points des tours des k -MST pour les différentes valeurs de k , permettant ainsi d'obtenir un algorithme d'approximation- γ du MLP avec $\gamma \simeq 3,5912$. 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- $2 \cdot 3 \cdot \gamma \sim 21,55$ du MLP.

L'utilisation de l'algorithme d'approximation-3 de Garg [13] pour le k -MST permet de faire descendre la borne à $1 \cdot 3 \cdot \gamma \simeq 10,78$ qui était la meilleure borne pour un algorithme d'approximation du MLP dans le cas général jusqu'il y a peu. Arora et Karakostas ont développé un algorithme d'approximation- $(2 + \epsilon)$ du k -MST en 2000 [3], ce qui permet de faire descendre la borne à $7,18 + \epsilon$. C'est la meilleure borne actuellement. Ceci dit, en 2003, A. Archer and D. P. Williamson [2] 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.

Les cas polynomiaux

Afrati et al. démontrent en 1986 [1] comment résoudre le cas de la ligne droite par un algorithme de programmation dynamique en $O(p^2)$ (celui-ci est repris et adapté dans la section 5.2. Peu après, en 1989, Minieka [18] prouve qu'une recherche en profondeur donne une solution optimale pour l'arbre unitaire (cette approche est reprise à la section 5.3. Dans leur article [6], 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. Finalement, García et al. [12] propose un algorithme en $O(p)$ pour le cas de la ligne droite en 2001.

Chapitre 3

Un modèle mathématique pour le PPCC

Le modèle proposé ici est un programme linéaire en nombres entiers et se base sur la construction, étape par étape, d'une solution optimale. Les variables sont toutes binaires et indiquent si oui ou non des morceaux du graphe font partie de la solution optimale.

3.1 Idée de base du modèle et définition des variables de décision

L'idée du modèle se base sur la construction d'une solution. Reprenons notre exemple de la section 1.3 dont la figure 3.1 représente la construction d'une solution optimale.

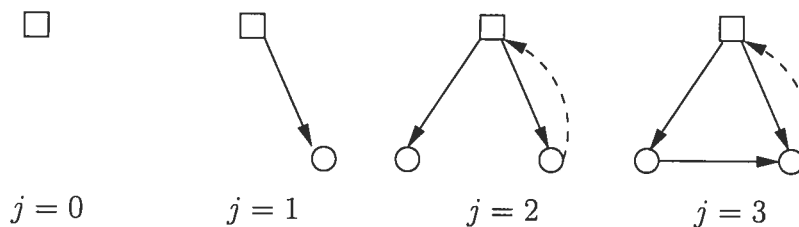


FIG. 3.1 – La construction de la solution optimale.

La construction, étape par étape, de la solution optimale de la figure 1.4.

Cette solution optimale est construite en autant d'étapes qu'il y a d'arêtes. Pour chacune de ces étapes, le modèle indique si oui ou non des arêtes feront partie de la solution partiellement construite. Un indice j parcourra les étapes 1 à n . Parce que le dépôt est une donnée a priori du problème, une étape $j = 0$ permettra d'initialiser la solution.

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

Définition 3.1.1 *Un chemin partiel est une partie de la tournée du dépôt jusqu'à une arête nouvellement desservie.*

La figure 3.2 représente un chemin partiel pour notre exemple.

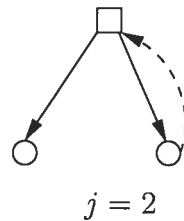


FIG. 3.2 – Un chemin partiel.

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

Nous avons encore besoin d'un concept que nous utiliserons tout au long de ce mémoire. Pour passer d'un chemin partiel à un autre il nous faut désigner les arêtes qui permettent ce passage. Nous appellerons ce chemin un *diamètre*.

Définition 3.1.2 *Le diamètre d'un chemin partiel j est le chemin qui raccorde le dernier sommet du chemin partiel $j - 1$ au dernier sommet du chemin partiel j .*

La figure 3.3 représente le diamètre du chemin partiel $j = 2$ de notre exemple.

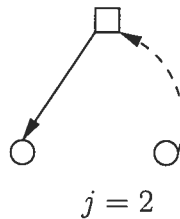


FIG. 3.3 – Un diamètre.

Le diamètre de l'étape $j = 2$ pour la solution optimale de la figure 1.4.

Nous allons tout simplement modéliser la construction d'une solution optimale en exigeant qu'à chaque étape un nouveau chemin partiel soit construit à partir de

l'ancien. Si toutes les arêtes sont desservies et ce pour un coût cumulé minimum, alors nous avons une solution optimale.

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 fait partie d'une solution optimale en construction ou non.

Les arêtes

$$v_a^j = \begin{cases} 1 & \text{si l'arête } a \text{ est desservie (visitée pour la première fois)} \\ & \text{dans le chemin partiel } j, \\ 0 & \text{sinon.} \end{cases}$$

$$d_a^j = \begin{cases} 1 & \text{si l'arête } a \text{ est utilisée dans le diamètre du chemin partiel } j, \\ 0 & \text{sinon.} \end{cases}$$

Les sommets

$$V_S^j = \begin{cases} 1 & \text{si le sommet } S \text{ est le dernier sommet du chemin partiel } j, \\ 0 & \text{sinon.} \end{cases}$$

$$D_S^j = \begin{cases} 1 & \text{si le diamètre traverse le sommet } S \text{ (} S \text{ n'est ni le premier} \\ & \text{ni le dernier sommet du diamètre),} \\ 0 & \text{sinon.} \end{cases}$$

L'objectif sera alors

$$\min z = \sum_{a \in \mathbf{A}} c(a) \left[\sum_{j=1}^n (n-j+1) \cdot d_a^j \right].$$

Pour notre exemple de la section 1.3, la solution optimale obtenue a bien un coût total de 57 :

$$\begin{aligned} z &= c(a) \cdot [3d_a^1 + 2d_a^1 + d_a^1] + \\ &\quad c(b) \cdot [3d_b^1 + 2d_b^1 + d_b^1] + \\ &\quad c(c) \cdot [3d_c^1 + 2d_c^1 + d_c^1] \\ &= 1 \cdot [3 \cdot 1 + 2 \cdot 1 + 1 \cdot 0] + \\ &\quad 50 \cdot [3 \cdot 0 + 2 \cdot 0 + 1 \cdot 1] + \\ &\quad 1 \cdot [3 \cdot 0 + 2 \cdot 1 + 1 \cdot 0] \\ &= 5 + 50 + 2 \\ &= 57 \end{aligned}$$

3.2 Les contraintes

Nous avons subdivisé les contraintes en quatre groupes : un premier groupe sur les arêtes, un deuxième groupe sur les sommets, un troisième groupe sur les contraintes liantes entre les sommets et les arêtes, et le dernier groupe pour l'initialisation.

Sur les arêtes

Chaque arête ne peut être desservie qu'une seule fois :

$$\forall a \in \mathbf{A} \quad \sum_{j=1}^n v_a^j = 1.$$

Seulement une arête peut être desservie à la fois dans un chemin partiel j :

$$j = 1..n \quad \sum_{a \in \mathbf{A}} v_a^j = 1.$$

On ne peut utiliser l'arête a dans un diamètre pour construire un nouveau chemin partiel que si elle a déjà été traitée (voir la deuxième remarque de la section 4.2 :

$$j = 1..n, \forall a \in \mathbf{A} \quad v_a^j \leq d_a^j.$$

$$\forall a \in \mathbf{A}, j = 2..n, k < j \quad v_a^j \leq 1 - d_a^k.$$

Sur les sommets

Les sommets à l'intérieur d'un diamètre (c'est-à-dire les sommets reliant deux arêtes appartenant au diamètre) ont leur variable indicatrice $D_S^j = 1$ alors que les deux extrémités du diamètre ont la leur à zéro. Si le sommet S débute le diamètre alors il s'agit du dernier sommet du chemin partiel $j - 1$ et $V_S^{j-1} = 1$. Si, par contre, il s'agit du dernier sommet du diamètre alors $V_S^j = 1$. Nous avons donc :

$$\forall S \in \mathbf{S}, j = 1..n \quad D_S^j + V_S^{j-1} + V_S^j \leq 1.$$

Il ne doit y avoir qu'un seul sommet pour terminer un chemin partiel :

$$j = 1..n \quad \sum_{S \in \mathbf{S}} V_S^j = 1.$$

Sur les sommets et les arêtes

Vérifions l'existence d'un diamètre d'un chemin partiel.

Pour tous les sommets nous avons :

$$\forall S \in \mathbf{S}, j = 1..n \quad \sum_{a; S \in a} d_a^j = 2D_S^j + V_S^{j-1} + V_S^j.$$

C'est une contrainte liante entre les arêtes et les sommets. Il y aussi la contrainte sur les degrés :

$$j = 1..n \quad 2 \sum_{a \in \mathbf{A}} d_a^j = \sum_{S \in \mathbf{S}} (2D_S^j + V_S^{j-1} + V_S^j)$$

qui se réduit à

$$j = 1..n \quad 2 \sum_{a \in \mathbf{A}} d_a^j = 2 \sum_{S \in \mathbf{S}} D_S^j + 2,$$

puis à

$$j = 1..n \quad \sum_{a \in \mathbf{A}} d_a^j = \sum_{S \in \mathbf{S}} D_S^j + 1,$$

pour finalement s'écrire de la façon classique :

$$j = 1..n \quad \sum_{a \in \mathbf{A}} d_a^j - 1 = \sum_{S \in \mathbf{S}} D_S^j.$$

Si une arête appartient à un diamètre, il faut que ses deux sommets soient compris sur le diamètre. Soit comme un sommet appartenant au diamètre ($D_S^j = 1$), soit comme le début du diamètre ($V_S^{j-1} = 1$) soit comme la dernière extrémité du diamètre ($V_S^j = 1$):

$$\forall a \in \mathbf{A}, j = 1..n, S \in a \quad d_a^j \leq D_S^j + V_S^{j-1} + V_S^j.$$

Il faut un dernier sommet à la dernière arête d'un diamètre :

$$\forall a \in \mathbf{A}, \forall j = 1..n \quad v_a^j \leq \sum_{S \in a} V_S^j.$$

Initialisation

La tournée commence au sommet privilégié. Appelons celui-ci A .

$$V_A^0 = 1.$$

Pour tous les autres sommets, nous avons

$$\forall S \in \mathbf{S} \setminus \{A\} \quad V_S^0 = 0.$$

3.3 Le modèle

Le problème du postier chinois cumulatif (PPCC) peut se formuler ainsi :

$$\min z = \sum_{a \in \mathbf{A}} c(a) \left[\sum_{j=1}^n (n-j+1) \cdot d_a^j \right].$$

sujet à

$$\forall a \in \mathbf{A} \quad \sum_{j=1}^n v_a^j = 1 \quad (3.1)$$

$$j = 1..n \quad \sum_{a \in \mathbf{A}} v_a^j = 1 \quad (3.2)$$

$$j = 1..n, \forall a \in \mathbf{A} \quad v_a^j \leq d_a^j \quad (3.3)$$

$$\forall a \in \mathbf{A}, j = 2..n, k < j \quad v_a^j \leq 1 - d_a^k \quad (3.4)$$

$$\forall S \in \mathbf{S}, j = 1..n \quad \sum_{a; S \in a} d_a^j = 2D_S^j + V_S^{j-1} + V_S^j \quad (3.5)$$

$$j = 1..n \quad \sum_{a \in \mathbf{A}} d_a^j - 1 = \sum_{S \in \mathbf{S}} D_S^j \quad (3.6)$$

$$\forall S \in \mathbf{S}, j = 1..n \quad D_S^j + V_S^{j-1} + V_S^j \leq 1 \quad (3.7)$$

$$j = 1..n \quad \sum_{S \in \mathbf{S}} V_S^j = 1 \quad (3.8)$$

$$\forall a \in \mathbf{A}, \forall j = 1..n \quad v_a^j \leq \sum_{S \in a} V_S^j \quad (3.9)$$

$$\text{Le dépôt } A \quad V_A^0 = 1 \quad (3.10)$$

$$\forall S \in \mathbf{S} \setminus \{A\} \quad V_S^0 = 0. \quad (3.11)$$

et toutes les variables sont binaires.

Un graphe avec n arêtes et p sommets sera modélisé par $2n^2 + (2n+1)p$ variables et $n^3/2 + 3/2n^2 + 4n + (2n+1)p$ contraintes.

A cause de sa taille exorbitante, un tel modèle n'est pas d'une grande utilité pratique. C'est pourquoi il n'est pas étudié plus en détail ici. Néanmoins, il a pu nous servir pour résoudre de petites instances ($n \leq 15$; $p \leq 15$) avec CPLEX (<http://www.ilog.com>). Pour donner un exemple, une grille unitaire (3,3) de 16

sommets et 24 arêtes a été résolue à l'optimum avec CPLEX 7.1 sur une machine Ultra Sparc III avec un processeur à 1,2 Ghz et 2 Giga de RAM. Le modèle est composé de 1936 variables et 8656 contraintes. Malgré la puissance de la machine, il a fallu approximativement 24 jours (2 068 497,06 secondes) pour y arriver ! Pour ce faire, 1 569 893 505 itérations ont été nécessaires et 8 216 425 nœuds de l'arbre de séparation et évaluation progressive (B&B) ont été examinés . . . Evidemment, l'algorithme de résolution est le B&B standard de CPLEX et nous n'avons pas tenté de l'adapter à la structure du problème pour le rendre plus efficace, vu que ce n'était pas là le but de notre démarche. Ces résultats illustrent toutefois assez bien la difficulté du problème traité.

3.4 Un exemple

Pour mieux comprendre le modèle, utilisons-le pour résoudre le problème de la section 1.3. Alors qu'il ne s'agit que d'un graphe minuscule ne comprenant que 3 arêtes et 3 sommets; le modèle contient déjà 39 variables et 60 contraintes !

Le lecteur trouvera le fichier avec le modèle de ce problème en annexe à la section 6.3.2 à la section *Modèle linéaire du PPCC de l'exemple simple* ainsi que le fichier généré par CPLEX (*cplex.log*) à la section 6.3.2 à la section *Solution fournie par CPLEX pour l'exemple simple*.

CPLEX trouve bien une solution optimale. La construction de celle-ci est résumée dans la figure 3.4. L'arête reliant les sommets A et B est appelé a , celle reliant les sommets B et C est appelé b et l'arête restante c .

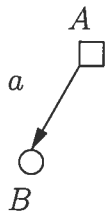
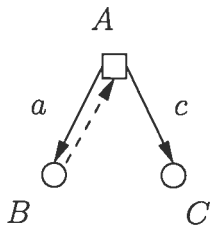
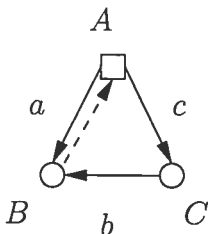
j	diamètre (d)	dernier sommet (V)	arête visitée (v)	Sommet traversé (D)	Solution partielle
0	/	A ($V0A = 1$)	/	/	A □
1	a ($d1a = 1$)	B ($V1B = 1$)	a ($v1a = 1$)	/	
2	a c ($d2a = 1$) ($d2c = 1$)	C ($V2C = 1$)	c ($v2c = 1$)	A ($D2A = 1$)	
3	b ($d3b = 1$)	B ($V3B = 1$)	b ($v3b = 1$)	/	

FIG. 3.4 – Une solution détaillée.

La solution donnée par CPLEX.

Chapitre 4

Complexité du PPCC et une première approche de résolution du problème

Dans ce chapitre, nous commençons par démontrer dans la première section que le PPCC est NP-difficile. La deuxième section couvre deux approches exactes et la troisième section propose une heuristique. Deux bornes inférieures sont proposées dans la quatrième section.

Nous rappelons au lecteur que ce mémoire constitue une première approche théorique du PPCC, qu'il ne soit donc pas étonné de ne pas retrouver ici ni implémentations informatiques ni résultats expérimentaux.

4.1 Le PPCC est NP-difficile

Le PPCC est NP-difficile et notre démonstration repose sur la transformation du MLP sur les arbres en un PPCC sur les arbres. C'est Sitters [20] qui a démontré en 2002 que le MLP sur les arbres est NP-difficile.

Bien que très similaires sur les arbres, le MLP et le PPCC se différencient par l'exigence supplémentaire pour le MLP de retourner au dépôt une fois la dernière arête desservie alors que pour le PPCC, la tournée s'arrête une fois la dernière arête desservie. Cette différence peut jouer dans la construction de la solution optimale comme le montre la figure 4.1. On ne peut donc pas utiliser exactement le même graphe pour les deux problèmes. Le tableau I montre les différentes solutions possibles pour les deux problèmes.

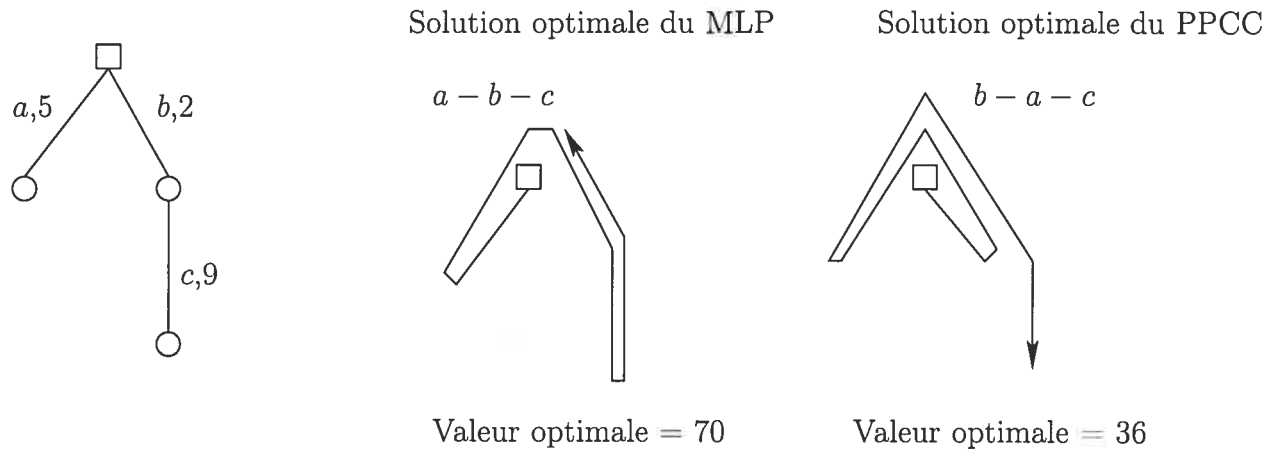


FIG. 4.1 – Différence entre le MLP et le PPCC.

Suivant le type de problème, MLP ou PPCC, la solution optimale est différente.

	$a - b - c$	$b - a - c$	$b - c - a$
MLP	$5 + 12 + 21 + 32 = 70$	$2 + 9 + 25 + 36 = 72$	$2 + 11 + 27 + 32 = 72$
PPCC	$5 + 12 + 21 = 38$	$2 + 9 + 25 = 36$	$2 + 11 + 27 = 40$

TAB. I – Les différentes solutions pour les deux problèmes MLP et PPCC de la figure 4.1.

Cette remarque faite, démontrons maintenant que le PPCC est NP-difficile.

Théorème 4.1.1 *Le PPCC est NP-difficile.*

Démonstration

La démonstration se compose de deux parties : une première partie consiste à montrer la transformation du MLP sur les arbres en un PPCC sur les arbres et à vérifier l'équivalence entre le MLP sur les arbres et sa version transformée en PPCC sur les arbres. La deuxième partie consiste à démontrer que cette transformation s'effectue en temps polynomial fonction du nombre des arêtes.

La transformation est illustrée sur la figure 4.2 et consiste à rajouter une arête de coût infini (c'est-à-dire très élevé) au dépôt.

Démontrons l'équivalence entre les deux problèmes. Le rajout de cette arête au dépôt dans le PPCC force le retour de la tournée au dépôt. En effet, comme le coût de cette arête est infini elle ne sera desservie qu'en dernier, forçant ainsi la tournée à repasser par le dépôt. Si on retire cette

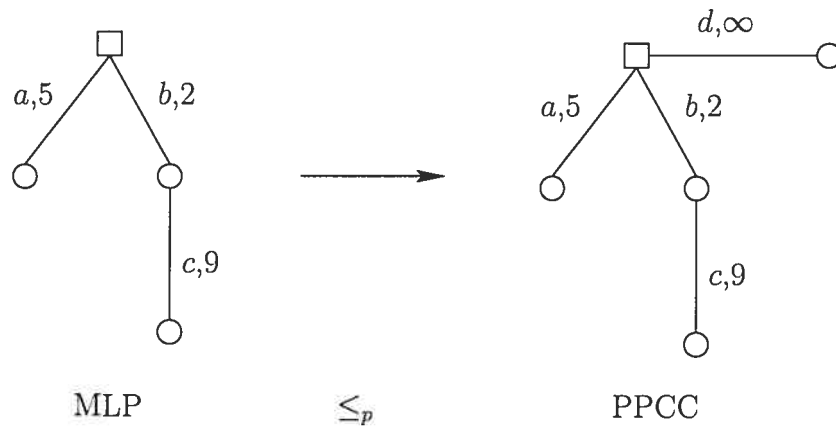


FIG. 4.2 – Transformation du MLP en PPCC.

Pour transformer un MLP en un PPCC sur les arbres, il suffit d'ajouter au graphe une arête de longueur M , avec M très grand, (Sur le graphe, M est représentée par ∞ comme mentionné dans les conventions) au dépôt pour forcer le retour à celui-ci.

arête de coût infini d'une solution optimale du PPCC, on retrouve exactement une solution optimale du MLP. Les deux problèmes sont donc bien équivalents.

La transformation s'effectue évidemment en temps polynomial. Nous pouvons donc conclure que le PPCC est NP-difficile sur les arbres et donc NP-difficile en général.

□

Ainsi, nous voyons que notre choix de définition ouverte pour la tournée (pas de retour au dépôt) n'est en rien restrictive puisqu'il suffit de rajouter une arête de coût infini pour forcer le retour au dépôt.

Le PPCC est donc NP-difficile. Mais il y a plus. Dans son article, Sitters démontre que même si on se restreint à des coûts de 0 ou 1 pour les arêtes, le MLP est NP-difficile. Notre démonstration ne changeant pas les coûts dans la transformation du MLP en PPCC, nous pouvons utiliser la même démarche pour réduire un MLP sur les arbres de coûts 0 et 1 en un PPCC sur les arbres de coûts 0 et 1. Ceci implique que non seulement le PPCC est NP-difficile mais qu'il est structurellement difficile, c'est-à-dire *fortement* NP-difficile.

4.2 Approches exactes

Nous proposons deux approches exactes pour la résolution du PPCC. La première est une énumération implicite de toutes les solutions possibles. La deuxième approche est une résolution du modèle par une méthode de séparation et évaluation progressive.

4.2.1 Première approche (énumération implicite)

Comme il faut couvrir toutes les arêtes pour obtenir une solution réalisable au PPCC, on peut caractériser une solution par une permutation des arêtes du graphe. À une permutation des arêtes correspond au moins une façon optimale au sens cumulatif de les parcourir dans cet ordre.

L'approche brute consiste donc à énumérer toutes les permutations des arêtes et à comparer les solutions réalisables ainsi obtenues. Un graphe G comprenant n arêtes pourra être parcouru selon les $n!$ permutations différentes, ce qui est bien sûr trop pour être utilisé de manière raisonnable. Néanmoins quelques remarques permettent de réduire significativement le nombre de solutions candidates.

Remarque 1

Tout d'abord, dans le problème qui nous intéresse, le nœud de départ est fixé a priori. Nous ne devons donc considérer que les solutions qui partent de ce nœud, c'est-à-dire les permutations des arêtes qui commencent par une arête partant du dépôt.

Remarque 2

Ensuite, considérons une solution optimale a_1, a_2, \dots, a_n . Le chemin pour aller de a_i à a_{i+1} passera nécessairement par les arêtes déjà couvertes avant de desservir a_{i+1} puisque nous ne faisons pas de différence entre le coût de passage et de service. En effet, si une tournée passe sur une arête avant de la desservir, non seulement le coût cumulatif de celle-ci sera plus élevé mais surtout le temps de passage sera comptabilisé dans le coût cumulatif de toutes les arêtes desservies après le passage sur l'arête considérée.

Nous avons donc

$$\text{chemin}(a_i, a_{i+1}) \subset \{a_1, a_2, \dots, a_i\}$$

ce qui nous permet ici aussi d'élaguer le nombre de candidats.

Ces deux premières remarques nous permettent de construire des solutions réalisables en partant du dépôt sans nul besoin de générer des permutations. En effet, il suffit de rajouter une à une toutes les arêtes joignables à partir d'un chemin partiel.

Remarque 3

L'aspect cumulatif joue en notre faveur. En effet, si les valeurs des arêtes sont assez différenciées, des solutions commençant par des arêtes de coût élevé seront éliminées assez vite car les coûts de ces arêtes s'accumuleront avec chaque nouvelle arête desservie dans la suite de la tournée. Une borne supérieure permettra d'éliminer ces solutions trop coûteuses. Meilleure sera la borne, plus vite ces solutions disparaîtront de l'éventail des solutions réalisables à considérer.

Une heuristique est donnée à la section 4.3 à cet effet.

Remarque 4

Finalement, nous pouvons éliminer certains chemins partiels dans la construction de solutions réalisables. Ainsi, si nous construisons les deux chemins suivants :

$$a_1 - a_2 - a_3 - a_4 - a_5$$

$$a_1 - a_3 - a_4 - a_2 - a_5$$

et qu'ils se terminent dans le même sens, une comparaison de leur coût cumulatif respectif permettra peut-être d'en éliminer un. En effet, les deux chemins se rendent de a_1 à a_5 en passant par les mêmes arêtes a_2 , a_3 et a_4 . On peut se permettre de ne garder que le moins coûteux au sens cumulatif.

Un exemple

Illustrons cette approche avec l'exemple décrit sur la figure 4.3.

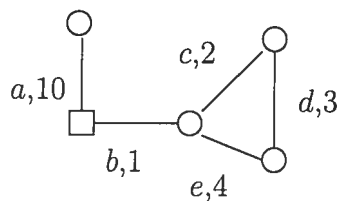


FIG. 4.3 – Un exemple pour illustrer l'approche énumérative.

Cette exemple a été résolu par l'approche d'énumération implicite en 18 itérations.

La première chose qu'il nous faut est une borne supérieure que nos débuts de solution ne peuvent pas franchir. Utilisons l'heuristique de la section 4.3. Celle-ci nous donne comme solution réalisable, la tournée $b - c - d - e - e - d - c - b - a$ pour un coût de 50.

Utilisons l'approche et énumérons toutes les solutions de manière implicite. Le tableau II montre les résultats produits itération après itération.

Tout d'abord, observons que la première remarque nous a permis de ne considérer que les permutations qui commencent soit avec l'arête a soit avec l'arête b .

La deuxième remarque nous autorise à ne regarder que les chemins faits d'arêtes joignables à partir d'un chemin partiel. Ainsi par exemple, si nous envisageons de commencer la tournée avec l'arête b , nous n'avons regardé que les possibilités $b - a$, $b - c$ et $b - e$, c'est-à-dire les arêtes joignables depuis l'arête b . Le cas $b - d$ n'a pas été considéré.

L'arête de coût 10 illustre bien la troisième remarque. Grâce au caractère cumulatif de la fonction objectif, les solutions qui commencent avec cette arête coûteuse ont assez vite été éliminées (ligne 1 et ligne 2). Et plus les différences entre les coûts sont marqués, plus le caractère cumulatif permet de dissocier les solutions.

Finalement, la quatrième remarque prend tout son sens si on observe les lignes 10 ($b - c - d - e - a$), 12 ($b - c - e - d - a$), 16 ($b - e - c - d - a$) et 18 ($b - e - d - c - a$). Il s'agit de solutions réalisables complètes parce que, pour des raisons d'économie de place, nous avons considéré un petit graphe qui ne contient que cinq arêtes. On peut très bien imaginer un graphe bien plus grand avec des arêtes qui partent depuis l'arête a . Dans ce cas, nous pourrions comparer ces débuts de solution et ne retenir que la meilleure, coupant dans l'arbre des éventualités nombre de solutions inintéressantes.

4.2.2 Deuxième approche (méthode de séparation et évaluation progressive)

Notre modèle se prête bien à une résolution par une méthode de séparation et évaluation progressive. En effet, les variables étant toutes binaires nous pouvons aisément brancher sur celles-ci. Nous présentons une façon de faire mais une étude plus poussée du modèle permettrait sans doute d'améliorer la méthode.

	1 arête	2 arêtes	3 arêtes	4 arêtes	5 arêtes	
1	a	10	$a-b-c$	$31+23=54$		
2		$a-b$	$10+21=31$			
3	b	1	$a-b-e$	$31+25=56$		
4		$b-a$	$1+12=13$	$b-a-c-d$	$38+28=64$	
5			$b-a-c$	$13+25=38$	$b-a-c-e$	$38+31=69$
6			$b-a-e$	$13+27=40$	$b-a-e-c$	$40+33=73$
7			$b-c-a$	$4+16=20$	$b-a-e-d$	$40+30=70$
8		$b-c$	$1+3=4$	$b-c-a-d$	$20+30=50$	
9			$b-c-d$	$4+6=10$	$b-c-a-e$	$20+31=51$
10				$b-c-d$	$10+22=32$	
11			$b-c-e$	$4+9=13$	$b-c-d-e$	$10+10=20$
12				$b-c-e$	$13+24=37$	
13			$b-e-a$	$6+20=26$	$b-c-e-d$	$13+11=24$
14		$b-e$	$1+5=6$	$b-e-a-c$	$26+33=59$	
15			$b-e-c$	$6+11=17$	$b-e-a-d$	$26+38=64$
16				$b-e-c$	$17+24=41$	
17			$b-e-d$	$6+8=14$	$b-e-c-a$	$17+15=32$
18				$b-e-d$	$14+26=40$	
				$b-e-d-c$	$14+10=24$	
					$b-e-c-a-d$	$41+40=81$
					$b-e-c-d-a$	$32+30=62$
					$b-e-d-a-c$	$40+39=79$
					$b-e-d-c-a$	$24+21=45$
					$b-c-d-a-e$	$32+47=79$
					$b-c-d-e-a$	$20+21=41$
					$b-c-e-a-d$	$37+53=90$
					$b-c-e-d-a$	$24+24=48$

TAB. II – Les différentes solutions parcourues par l'approche d'énumération implicite.

La variable v_a^j indiquant si une arête a est servie en j^{e} position est certainement une bonne candidate pour le branchement. Si celle-ci est à un, toutes les autres variables similaires pour une période donnée sont à zéro.

Nous pourrions brancher sur n'importe quelle arête à priori mais il faudrait alors une méthode pour obtenir une borne supérieure. Nous ne possédons pas de méthode simple et efficace qui nous donnerait une borne supérieure à partir du seul renseignement qu'une arête est traitée en j^{e} position. Notre seul outil est l'heuristique proposée dans la section 4.3. Celle-ci nous permet de construire une solution aisément si nous connaissons les premières arêtes desservies. Nous brancherons donc d'abord sur les variables v^1 , puis v^2 et ainsi de suite dans cet ordre.

Pour éviter une méthode de séparation et évaluation progressive qui reviendrait finalement à utiliser la même approche que notre méthode d'énumération implicite, nous ne choisirons pas forcément les premières arêtes parmi celles qui débutent au dépôt. Le modèle nous permet de considérer n'importe quelle arête pour commencer le branchement. Nous brancherons sur les arêtes par ordre croissant de grandeur. Si une arête ne peut pas être atteinte dans une période donnée, cela fera une branche de moins à explorer dans l'arbre de recherche.

A chaque nœud de l'arbre de recherche, nous utiliserons donc l'heuristique pour calculer une borne supérieure, et pour la borne inférieure, nous utiliserons la solution du modèle obtenu en relaxant les contraintes d'intégralité des variables.

Pour obtenir de bonnes solutions le plus vite possible, nous ferons une recherche en profondeur dans l'arbre de la méthode de séparation et d'évaluation progressive. Comme c'est exactement ce que fait l'heuristique, bien souvent il ne faudra pas la recalculer quand on descendra le long d'une branche où l'on sert les arêtes.

Reprenons l'exemple utilisé pour illustrer la première approche exacte d'énumération implicite. Le graphe est représenté à la figure 4.3. L'arbre de recherche est affiché à la figure 4.4.

Tout d'abord commençons par ordonner les arêtes en ordre croissant de grandeur :

$$b \leq c \leq d \leq e \leq a.$$

Notre première variable de branchement est v_b^1 . Cette arête est accessible depuis le dépôt mais si cela n'avait pas été le cas, nous aurions simplement continué notre recherche avec $v_b^1 = 0$. Posons donc $v_b^1 = 1$. Nous nous retrouvons au premier nœud de la recherche. En relaxant les contraintes d'intégralité du modèle nous trouvons

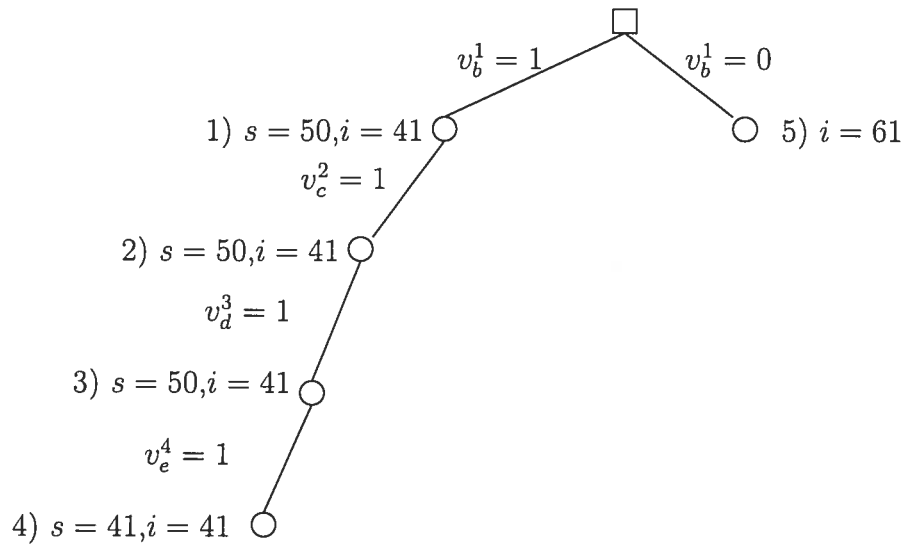


FIG. 4.4 – L'arbre de recherche de la méthode de séparation et évaluation progressive.

La méthode est illustrée sur l'exemple de la figure 4.3. Il suffit d'explorer 5 nœuds.

une borne inférieure de 41. L'heuristique fait le trajet $b - c - d - e - a$ pour un coût de 50. Nous continuons notre descente avec $v_c^2 = 1$ puisque c est l'arête de plus petite longueur encore libre. A nouveau notre borne inférieure est de 41. Pas besoin de recalculer notre borne inférieure puisque dans la tournée choisie en 1) c'était déjà l'arête c qui avait été desservie en deuxième position. La recherche en profondeur continue. Passons tout de suite au nœud 4). A ce niveau-là, une solution est trouvée puisque nous avons déterminé l'ordre de service des 5 arêtes. En général, l'heuristique ne trouvera pas la meilleure solution à partir de la permutation donnée des arêtes et il faut inclure une routine pour trouver cette solution. Dans ce cas-ci, l'heuristique ne retrouve pas une solution optimale puisqu'elle nous redonne encore d'abord le chemin partiel $b - c - d - e$. À partir de e il faut rejoindre l'arête a . Notre heuristique ne pouvant l'atteindre à partir de l'extrémité de l'arête e , elle rebrousse chemin $e - d - c - b - a$. La solution optimale est bien sûr $b - c - d - e - b - a$ pour un coût de 41. Comme c'est précisément aussi la valeur de notre borne inférieure au nœud 1) nous pouvons être sûr d'avoir obtenu une solution optimale pour cette partie de l'arbre. Il faut maintenant essayer $v_b^1 = 0$. Nous obtenons une borne inférieure de 61 qui nous permet de conclure que la solution obtenue précédemment est une solution optimale.

Comment obtenir une solution *optimale* à partir d'une permutation donnée? Les deux méthodes suivantes peuvent être utilisées. La première consiste tout simplement à trouver les plus courts chemins entre le dernier sommet visité et la nouvelle arête desservie dans un sous-graphe constitué des seules arêtes déjà desservies au-

paravant (rappelons que puisque le coût de service et de passage sont identiques, il n'est pas économique de passer sur une arête non encore desservie). Si on ajoute des contraintes supplémentaires au problème, on peut faire appel à une deuxième méthode qui consiste en une méthode de séparation et d'évaluation progressive sur le modèle où les variables v_e^j , qui indiquent l'ordre de service des arêtes, sont toutes fixées. La méthode permet alors de déterminer les autres variables, c'est-à-dire les plus courts chemins, tout en respectant les contraintes supplémentaires.

4.3 Une heuristique

Le principe de cette heuristique est simple. A partir du dépôt, la tournée est construite en prenant systématiquement l'arête non encore couverte de moindre coût à partir du nœud courant. S'il y a plusieurs arêtes candidates de même coût nous nous contenterons d'en choisir une pour notre première étude théorique du PPCC. S'il n'y a plus aucune arête disponible à partir d'un sommet, on retourne en arrière (*backtracking*) pour trouver de nouvelles arêtes disponibles à partir d'un sommet déjà visité et ainsi de suite jusqu'à couvrir toutes les arêtes. Un exemple illustré à la figure 4.5 fera mieux comprendre le fonctionnement de l'heuristique.

L'heuristique choisit d'abord l'arête AB puisque du dépôt c'est l'arête directement atteignable de moindre coût, puis elle emprunte l'arête BC . Arrivée en C , il lui faut reculer en repassant sur l'arête BC . En B , elle dessert l'arête BD puis de proche en proche, n'ayant aucune alternative, les arêtes DE , EF , FG et GB . Comme toutes les arêtes aboutissant au sommet B ont déjà été desservies, il faut donc que l'heuristique rebrousse chemin jusqu'à atteindre un sommet qui possède une arête non visitée. Elle parcourt donc le chemin $B - G - F - E - D - B - C - B - A$. En A , il ne lui reste plus qu'à desservir la dernière arête AH .

Le coût cumulé obtenu de 330 est loin du coût cumulé 216 de la solution optimale mais la principale qualité de cette heuristique est sa rapidité et sa simplicité. Il est clair que d'autres possibilités sont envisageables et nous ne prétendons rien du point de vue de la qualité des solutions que cette heuristique fournit.

4.4 Bornes inférieures

Deux bornes inférieures sont décrites ci-dessous. La première est algébrique et fait seulement appel aux coûts des arêtes. La deuxième est plus fine et découle d'une relaxation des conditions d'intégralité des variables du modèle. Nous avons déjà pu faire sa connaissance dans la section 4.2.2 sur la méthode de séparation et évaluation

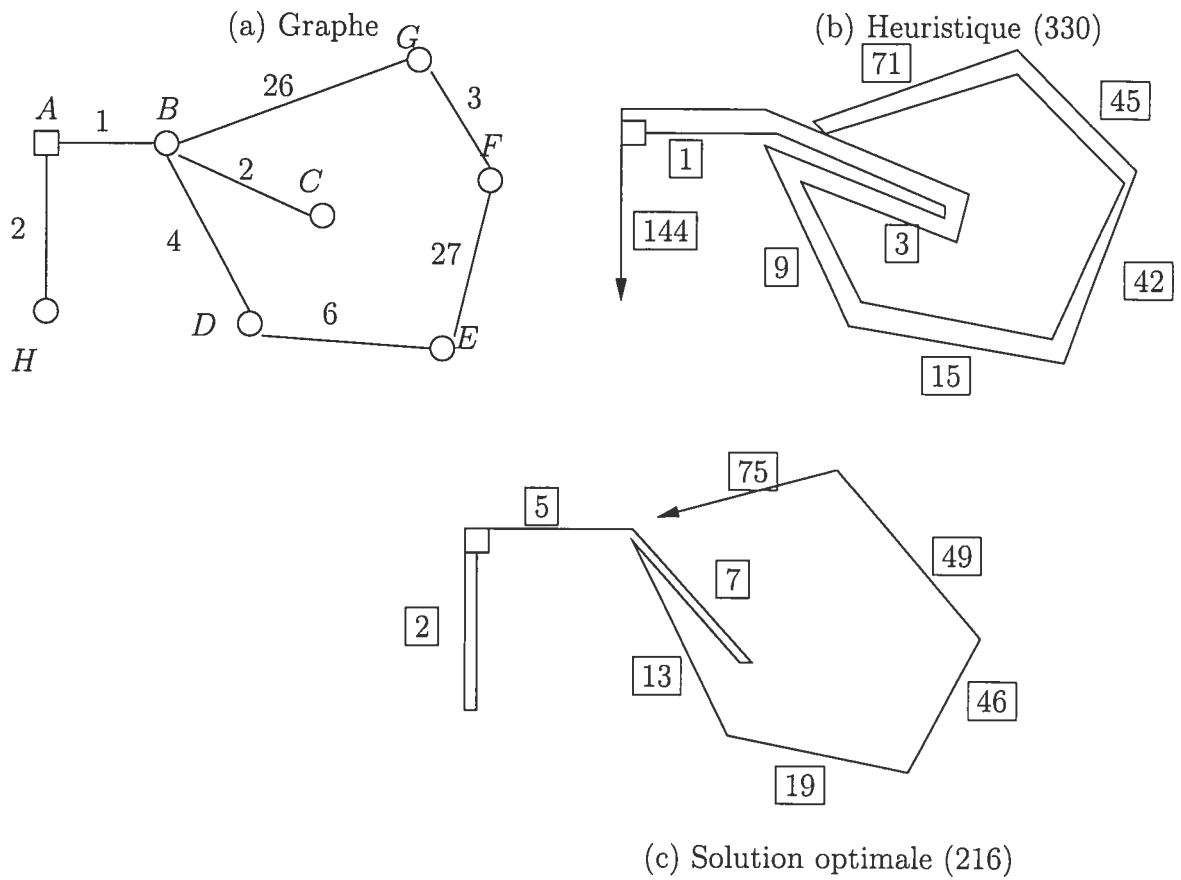


FIG. 4.5 – L'heuristique.

Le graphe (a) est parcouru par l'heuristique en (b) pour une valeur de 330 alors que la solution optimale en (c) ne vaut que 216. A côté de chaque arête se trouve en (a) sa valeur réelle et en (b) et (c) sa valeur cumulative.

progressive.

4.4.1 Première borne (borne algébrique)

Une borne inférieure qui vient relativement vite à l'esprit se construit comme suit. Ordonnons les coûts des arêtes en ordre croissant et renumérotions ces coûts dans cet ordre: $c_1 \leq c_2 \leq \dots \leq c_n$. Alors $n \cdot c_1 + (n-1) \cdot c_2 + \dots + 1 \cdot c_n$ constitue une borne inférieure de la fonction objectif.

Pour s'en convaincre, il suffit de prendre n'importe quelle permutation π des coûts et voir que

$$\text{borne inf} = n \cdot c_1 + \sum_{i=1}^{n-1} (n-i) \cdot c_{i+1} \leq n \cdot \pi(c_1) + \sum_{i=1}^{n-1} (n-i) \cdot \pi(c_{i+1}).$$

En effet, soit $\tilde{\pi}$ une permutation sur $\{1, 2, 3, \dots, n\}$ telle que $\pi(c_i) = c_{\tilde{\pi}(i)}$. Alors $n \cdot \pi(c_1) + (n-1) \cdot \pi(c_2) + \dots + \pi(c_n) = \tilde{\pi}(n) \cdot c_1 + \tilde{\pi}(n-1) \cdot c_2 + \dots + \tilde{\pi}(1) \cdot c_n$.

Décomposons le premier terme $\tilde{\pi}(n) \cdot c_1$ en $\tilde{\pi}(n)$ termes c_1 , le deuxième terme $\tilde{\pi}(n-1) \cdot c_2$ en $\tilde{\pi}(n-1)$ termes c_2 et ainsi de suite. Appelons chacun de ces termes en ordre croissant par b_i . Il y en a $\tilde{\pi}(n) + \tilde{\pi}(n-1) + \dots + \tilde{\pi}(1) = \frac{n(n+1)}{2}$.

Faisons de même avec la somme $n \cdot c_1 + (n-1) \cdot c_2 + \dots + c_n$ et appelons les $\frac{n(n+1)}{2}$ termes en ordre croissant par a_i .

Il est facile de voir que pour tout i (terme à terme):

$$a_i \leq b_i$$

et donc que

$$\sum a_i = n \cdot c_1 + (n-1) \cdot c_2 + \dots + c_n \leq \sum b_i = n \cdot \pi(c_1) + (n-1) \cdot \pi(c_2) + \dots + \pi(c_n)$$

Cette borne inférieure n'est pas très bonne puisqu'elle ne fait aucunement appel à la structure du graphe mais elle nous donne déjà un repère.

Pour illustrer notre borne, reprenons le problème de la grille unitaire (3,3) dont nous parlions à la section 3.3 dans la remarque qui suit le modèle. Rappelons que celui-ci a été résolu par CPLEX en 24 jours sur une machine puissante. La valeur de la fonction objectif à l'optimum est de 312. Notre borne algébrique nous donne la valeur de 300.

4.4.2 Deuxième borne (relaxation du modèle)

Une idée qui vient naturellement à l'esprit en analysant la formulation mathématique du modèle du chapitre 3 est de relaxer les conditions d'intégralité des variables. Vu la structure du modèle on peut même se permettre de prendre des variables réelles positives.

Il semblerait qu'entre les quatre méthodes de résolution proposées par CPLEX, i.e. simplexe primal, simplexe dual, simplexe réseau et la méthode de barrière logarithmique, cette dernière l'emporterait pour la vitesse de résolution. Pour donner une idée, reprenons encore une fois le problème de la grille unitaire (3,3) de la section 3.3 qui avait donné tellement de mal à une puissante machine. Notre borne inférieure est de 300 et a été obtenue en 3.97 secondes sur la même machine. Le tableau III indique les différents temps de résolution de la borne inférieure par relaxation obtenue avec les différentes méthodes de résolution de CPLEX 7.1, toujours sur la même machine.

Méthodes	Temps (en secondes)
Simplexe primal	8.67
Simplexe dual	29.14
Simplexe réseau	8.45
Barrière logarithmique	3.97

TAB. III – *Les différents temps de résolution par CPLEX de la borne inférieure.*

Dans ce cas particulier, les deux bornes coïncident mais ce n'est qu'un hasard. On peut s'attendre à ce que cette borne-ci, parce qu'elle fait appel à la structure du graphe, soit généralement meilleure que la borne algébrique.

Chapitre 5

Cas polynomiaux adaptés

Dans ce chapitre, nous introduisons trois cas qui peuvent être résolus en temps polynomial. Nous voyons dans les deux sections suivantes comment 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. [1] présenté pour le MLP 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 MLP et est donc nouveau. La troisième section traite de l'arbre unitaire qui lui se résout en $O(n)$ par une recherche en profondeur. Ce résultat est dû à Minieka [18]. Comme nous l'avons vu dans la section 4.1, le cas de l'arbre est très similaire pour le PPCC et le MLP. Il n'est donc pas surprenant de voir que l'approche de Minieka, malgré la petite différence entre le MLP et le PPCC, convienne aussi pour l'arbre unitaire dans le cas du PPCC.

5.1 Le cercle

Nous commençons par expliquer l'idée générale de l'algorithme de programmation dynamique d'Afrati et al. [1] en voyant le principe d'optimalité qui prévaut ici. Pour des raisons pédagogiques, nous introduisons d'abord la récursion avant et puis dans un deuxième temps, nous passons à la récursion arrière qui est plus efficace.

5.1.1 Idée générale et principe d'optimalité

Appelons les sommets comme sur la figure 5.1. Ceci nous permettra de parler d'une arête en l'appelant par le couple de sommets qu'elle possède. Ainsi on parlera de l'arête $(i, i + 1)$ par exemple.

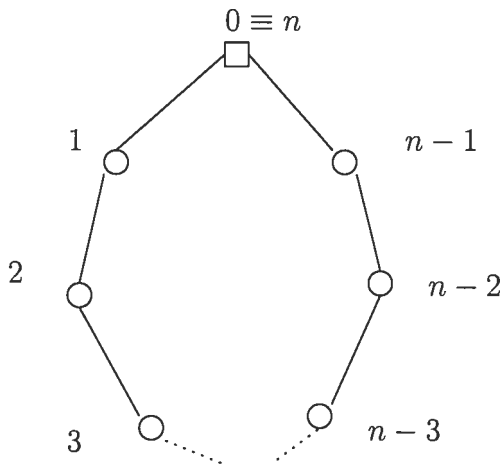


FIG. 5.1 – Numérotation des sommets du cercle.

Les sommets sont numérotés à partir de 0 pour le dépôt dans le sens anti-horaire. Le dépôt reçoit aussi le numéro n .

Théorème 5.1.1 *Le PPCC défini sur le cercle est de complexité polynomiale et se résout en temps $O(n^2)$.*

Démonstration

Le théorème découle de l'affirmation suivante : il existe un algorithme de programmation dynamique qui permet de résoudre le cas du cercle en temps $O(n^2)$. La démonstration se fait en deux temps. D'abord, exhiber le principe d'optimalité et le démontrer, ensuite établir que l'algorithme nous donne la solution optimale en temps $O(n^2)$.

Une solution consiste en une tournée qui part du dépôt et successivement dessert toutes les arêtes les unes après les autres. En effet, nous avons vu (remarque 2 de la section 4.2.1) qu'une tournée optimale dessertira une arête avant de l'emprunter. Vu la structure localement en ligne droite du cercle, les arêtes dans une solution optimale ne pourront être desservies que de proche en proche à partir du dépôt. Donc une solution optimale sera composée d'une succession d'arcs (orientés) de cercle, chacun couvrant une nouvelle arête comme sur la figure 5.2.

Le principe d'optimalité est que chacun de ces arcs de cercle est lui-même obtenu optimalement. En effet, si tel n'était pas le cas dans une solution optimale, il suffirait de remplacer la façon d'aboutir à l'arc de cercle considéré par une meilleure façon d'y parvenir. La solution en serait améliorée, ce qui contredirait le fait qu'il s'agisse d'une solution optimale.

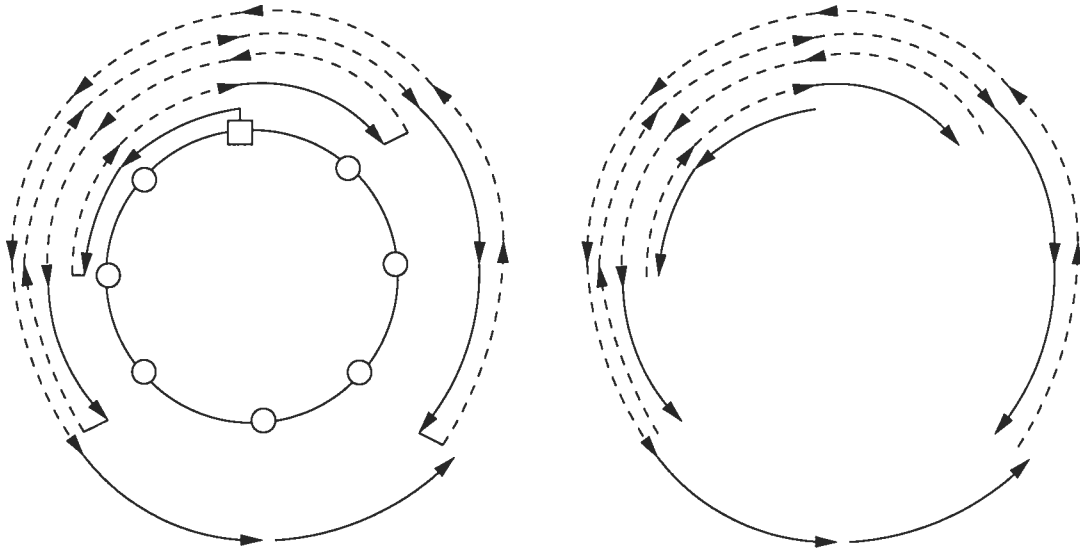


FIG. 5.2 – Une solution optimale est composée d’arcs de cercle.

A gauche est représentée une tournée optimale dont les arcs de cercle gauches et droits ont été séparés à droite.

Un arc de cercle ne peut être construit que de deux façons comme la figure 5.3 le suggère pour un arc de cercle gauche : soit en venant de la gauche, soit en venant de la droite.

Pour chaque paire de sommets, il y a deux possibilités (arc de cercle gauche et arc de cercle droit) ce qui nous donne $2 \cdot n^2$ variables à calculer. Cette décomposition des solutions nous donne donc la solution optimale en un temps $O(n^2)$.

□

5.1.2 Implémentation

Nous pouvons calculer la valeur des arcs de cercle dont nous venons de parler par un algorithme de programmation dynamique. Les deux types de récursions, avant et arrière, sont décrites dans les deux prochaines sous-sections.

Récursion avant

Définissons $L(i, j)$ comme étant la valeur optimale pour construire l’arc de cercle gauche couvrant toutes les arêtes depuis le sommet j en passant par le dépôt jusqu’au sommet $i - 1$ pour desservir l’arête $(i - 1, i)$ et de manière équivalente $R(i, j)$ pour le même arc de cercle, mais dans l’autre sens (celui-ci dessert maintenant l’arête $(j + 1, j)$).

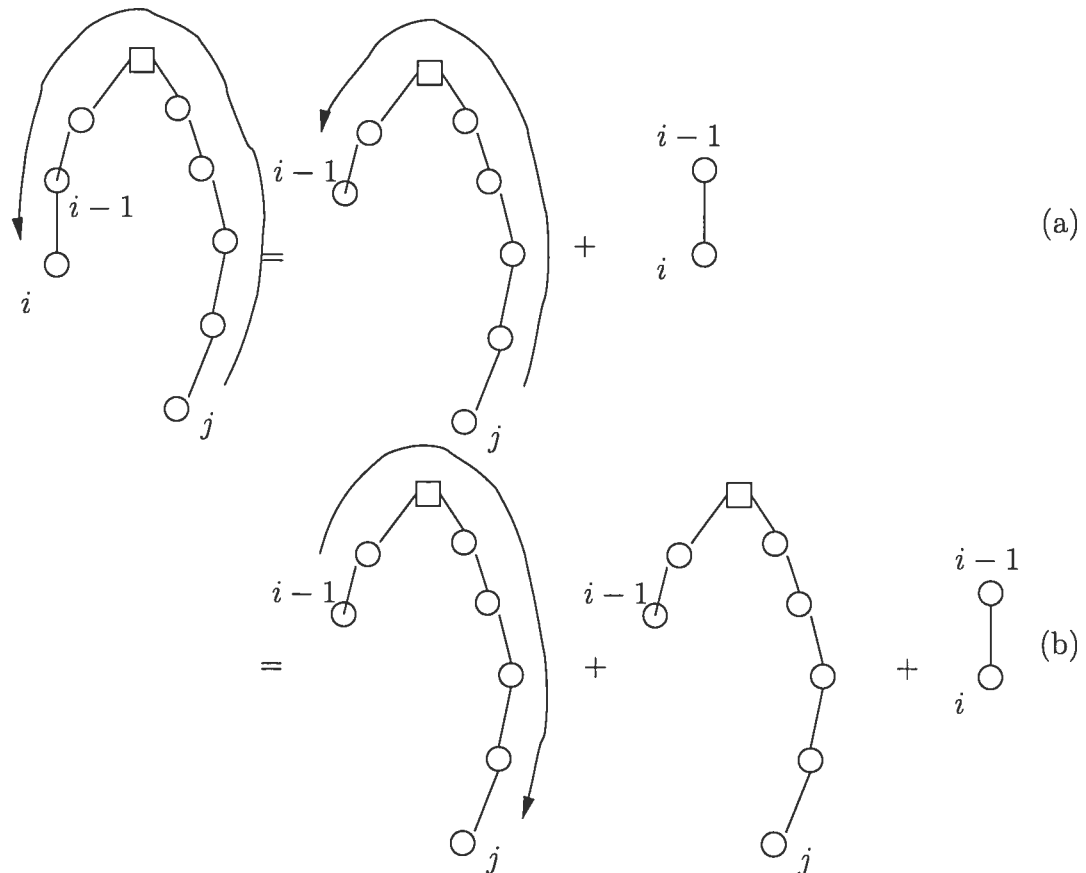


FIG. 5.3 – Comment former un arc de cercle gauche.

L'arc de cercle gauche représenté à gauche des deux signes d'égalité dessert la nouvelle arête $(i-1, i)$ et peut être formé de deux façons : soit (a) par la gauche et l'on va dans le sens anti-horaire du sommet j jusqu'au sommet $i-1$ puis on couvre l'arête $(i-1, i)$, soit (b) par la droite et on va dans le sens des aiguilles d'une montre du sommet $i-1$ jusqu'au sommet j , puis l'on revient dans le sens inverse jusqu'au sommet $i-1$ pour finalement couvrir l'arête $(i-1, i)$.

De manière générale, nous avons

$$L(i,j) = \min \left\{ \begin{array}{l} L(i-1,j) + \alpha \cdot c(i-1,i), \\ R(i-1,j) + \alpha \cdot [d(i-1,j) + c(i-1,i)] \end{array} \right\}$$

comme nous l'indique la figure 5.3. Le coefficient α est déterminé comme suit : avant l'ajout de l'arête $(i-1,i)$ il y a $(i-1)$ arêtes à gauche et $(n-j)$ arêtes à droite, soit $(i-1) + (n-j)$ arêtes au total. On ajoute donc la $((i-1) + (n-j) + 1)^{\text{e}}$ arête.

Or le coefficient de la k^{e} arête est $(n - (k-1))$ dans la fonction objectif et donc notre coefficient α est $n - (i-1 + n-j + 1 - 1) = j - i + 1$ et nous avons

$$L(i,j) = \min \left\{ \begin{array}{l} L(i-1,j) + (j-i+1) \cdot c(i-1,i), \\ R(i-1,j) + (j-i+1) \cdot [c(i-1,i) + d(i-1,j)] \end{array} \right\}.$$

De manière similaire nous avons aussi

$$R(i,j) = \min \left\{ \begin{array}{l} R(i,j+1) + (j-i+1) \cdot c(j-1,j), \\ L(i,j+1) + (j-i+1) \cdot [c(j-1,j) + d(i,j-1)] \end{array} \right\}.$$

Les deux formules pour obtenir $L(i,j)$ et $R(i,j)$ sont valables pour $1 < i < j < n$. Pour les autres cas de figure, nous avons d'abord pour les arcs de cercles impossibles, $L(n,j) = \infty$ et $R(i,0) = \infty$, ensuite pour les arcs de cercles commençant en 0, $L(i,0) = L(i-1,0) + (j-i+1) \cdot c(i-1,i)$ et $R(n,j) = R(n,j-1) + (j-i+1) \cdot c(j+1,j)$.

La valeur optimale est la valeur la moins grande parmi les $R(i,i)$ et $L(i,i) \forall i$ avec

$$\begin{aligned} R(i,i) &= R(i,i+1) + c(i,i+1), \\ L(i,i) &= L(i-1,i) + c(i-1,i). \end{aligned}$$

En effet, si toutes les arêtes, sauf une, sont desservies, la dernière sera desservie à partir du dernier sommet visité. Il est aberrant de vouloir repasser par les $n-1$ arêtes pour desservir la dernière arête, alors qu'elle est directement accessible.

Récursion arrière

Comme souvent, la récursion arrière donne de meilleurs résultats. Nous avons décrit la récursion avant et déclaré deux variables $R(i,j)$ et $L(i,j)$ avant tout dans

un souci pédagogique. Pour la récursion arrière, nous regroupons ces deux variables dans une seule variable $f(i,j,k)$ mais avec bien sûr plus d'états possibles.

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 ($\overleftarrow{f(i,j,1)}$).

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 $\overrightarrow{f(i,i+1,0)} = \min \{ c(i,i+1), d(i,i+1) + c(i,i+1) \}$ par exemple.

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

$$\overleftarrow{f(i,i+2,1)} = \min \left\{ 2 \cdot c(i,i+1) + \overleftarrow{f(i+1,i+2,1)}, \right. \\ \left. 2 \cdot \{d(i,i+2) + c(i+1,i+2)\} + \overrightarrow{f(i,i+1,0)} \right\}$$

En général :

$$\overleftarrow{f(i,j,1)} = \min \left\{ (j-i) \cdot c(i,i+1) + \overleftarrow{f(i+1,j,1)}, \right. \\ \left. (j-1) \cdot \{d(i,j) + c(j-1,j)\} + \overrightarrow{f(i,j-1,0)} \right\}$$

et

$$\overrightarrow{f(i,j,0)} = \min \left\{ (j-i) \cdot c(j-1,j) + \overrightarrow{f(i,j-1,0)}, \right. \\ \left. (j-1) \cdot \{d(i,j) + c(i,i+1)\} + \overleftarrow{f(i+1,j,1)} \right\}$$

Nous ne cherchons plus que $\overleftarrow{f(0,n,1)}$ et $\overrightarrow{f(0,n,0)}$ au lieu de devoir comparer n valeurs comme dans le cas de la récursion avant.

5.2 La ligne droite

La résolution du MLP sur la ligne droite est dû à Afrati et al. [1]. Le principe est le même que pour le cercle. Néanmoins, il y a une petite différence : comme la ligne droite ne se rejoint pas, il n'y a pas moyen de rejoindre automatiquement la dernière arête à desservir à partir de l'avant-dernière arête desservie comme dans le cas du cercle.

Voyons un exemple développé. La figure 5.4 montre la ligne droite choisie, ainsi que sa solution optimale, et comment nous en avons numéroté les sommets. A nouveau, nous exprimons les arêtes par leurs sommets. Ainsi, on parlera de l'arête (1,2). Il n'y a aucune ambiguïté car pour une arête (i,j) , si $i < j$ alors il s'agit d'une arête du côté droit du dépôt. Inversement, si $i > j$ alors l'arête se trouve du côté gauche du dépôt. Comme le diamètre $d(i,j)$ désignera un diamètre commençant à gauche en i et finissant à droite par le sommet j , il ne peut y avoir confusion.

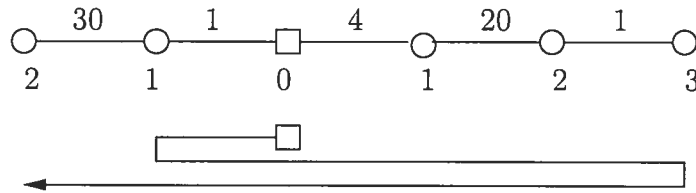


FIG. 5.4 – Une ligne droite et sa solution optimale.

Récursion avant

Commençons par la récursion avant. Les formules de récurrence sont exactement les mêmes que dans le cas du cercle excepté quelques petites retouches, puisque nous numérotions différemment les sommets :

$$L(i,j) = \min \left\{ \begin{array}{l} L(i-1,j) + k \cdot c(i-1,i), \\ R(i-1,j) + k \cdot [c(i,i-1) + d(i-1,j)] \end{array} \right\}$$

et

$$R(i,j) = \min \left\{ \begin{array}{l} R(i,j-1) + k \cdot c(j-1,j), \\ L(i,j-1) + k \cdot [c(j-1,j) + d(i,j-1)] \end{array} \right\},$$

avec $k = n - (i + j) + 1$. Nous recherchons $L(k,l)$ et $R(k,l)$ avec k et l respectivement le sommet le plus à gauche et celui le plus à droite.

Nous commençons par initialiser $L(0,j)$ et $R(i,0)$. $L(0,0) = R(0,0) = 0$ et $L(0,j) = R(i,0) = \infty \quad \forall i,j \geq 1$ car ces mouvements sont impossibles. Puis nous calculons $L(i,0)$ avec $i \geq 1$ et $R(0,j)$ avec $j \geq 1$ pour finalement alterner entre les lignes $L(i,\cdot)$ et $R(i,\cdot)$ par ordre de grandeur croissant de i , remplissant les lignes de gauche à droite. Les tableaux I et II montrent respectivement le calcul des variables $L(i,j)$ et $R(i,j)$ pour l'exemple proposé plus haut. La valeur à l'optimum est de $L(2,3) = 143$ et la figure 5.4 montre cette solution optimale.

i \ j	0	1	2	3
0	0	∞	∞	∞
1	5	40	175	155
2	125	130	195	143

TAB. I – Le calcul de $L(i,j)$ pour l'exemple de la figure 5.4.

i \ j	0	1	2	3
0	0	20	100	103
1	∞	25	85	87
2	∞	230	240	241

TAB. II – Le calcul de $R(i,j)$ pour l'exemple de la figure 5.4.

Récursion arrière

La récursion arrière ne nous épargne pas vraiment de calculs comme dans le cas du cercle dont nous reprenons la notation des $f(i,j,k)$. Ici aussi, il y a un léger ajustement des formules à faire à cause de la numérotation différente des sommets. Les voici :

$$\overleftarrow{f(i,j,1)} = \min \left\{ \begin{array}{l} (n - (i + j)) \cdot c(i + 1, i) + \overleftarrow{f(i + 1, j, 1)}, \\ (n - (i + j)) \cdot [d(i, j) + c(j, j + 1)] + \overrightarrow{f(i, j + 1, 0)} \end{array} \right\}$$

et

$$\overrightarrow{f(i,j,0)} = \min \left\{ \begin{array}{l} (n - (i + j)) \cdot c(j, j + 1) + \overrightarrow{f(i, j + 1, 0)}, \\ (n - (i + j)) \cdot [d(i, j) + c(i + 1, i)] + \overleftarrow{f(i + 1, j, 1)} \end{array} \right\}$$

Nous recherchons $\overleftarrow{f(0,0,1)}$ et $\overrightarrow{f(0,0,0)}$. Pour ce faire, nous commençons par calculer respectivement les dernières colonnes des tableaux $\overrightarrow{f(i,j,0)}$ et $\overleftarrow{f(i,j,1)}$ du bas vers le haut puis chacune de leur dernière ligne de la droite vers la gauche. Ensuite on alterne les calculs entre $\overrightarrow{f(i,j,0)}$ et $\overleftarrow{f(i,j,1)}$. On procède ligne après ligne en allant du bas vers le haut et de droite à gauche. Les tableaux III et IV montrent respectivement le calcul des variables $\overleftarrow{f(i,j,1)}$ et $\overrightarrow{f(i,j,0)}$ pour l'exemple proposé plus haut.

5.3 L'arbre avec coût unitaire

Le MLP 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 [18] s'étend pour le PPCC sur l'arbre unitaire. Que n'importe

i \ j	0	1	2	3
0	143	137	111	32
1	138	133	108	30
2	146	111	56	0

TAB. III – Le calcul de $f(i,j,1)$ pour l'exemple de la figure 5.4.

i \ j	0	1	2	3
0	143	153	85	82
1	134	118	58	56
2	53	41	1	0

TAB. IV – Le calcul de $f(i,j,0)$ pour l'exemple de la figure 5.4.

quelle recherche en profondeur donne une solution optimale peut paraître surprenant à première vue si l'on pense en termes de coûts généraux car cette approche ne donne pas nécessairement une solution optimale en général comme le montre la figure 5.5.

Par contre, quand les coûts sont unitaires toutes les recherches en profondeur donnent la même valeur optimale à la fonction objectif. La figure 5.6 reprend le graphe de la figure 5.5 et montre deux recherches en profondeur différentes qui pourtant donnent le même résultat.

Ce résultat se comprend aisément en adaptant la démonstration élégante de A. Blum et al. [6] pour le cas du MLP sur l'arbre unitaire.

Définissons d'abord la profondeur d'une arête comme la profondeur de son sommet le plus éloigné de la racine de l'arbre et notons celle-ci par $p(\text{arête})$.

Théorème 5.3.1 *Le PPCC dans le cas de l'arbre unitaire est polynomial et résoluble en temps $O(n)$.*

Démonstration

La démonstration s'articule en deux points :

- démontrer que pour n'importe quelle solution, le coût cumulatif d'une $i^{\text{ième}}$ arête est au moins de $2i - p(i^{\text{ième}} \text{ arête})$;
- démontrer que n'importe quelle recherche en profondeur atteint exactement cette borne inférieure de $2i - p(i^{\text{ième}} \text{ arête})$.

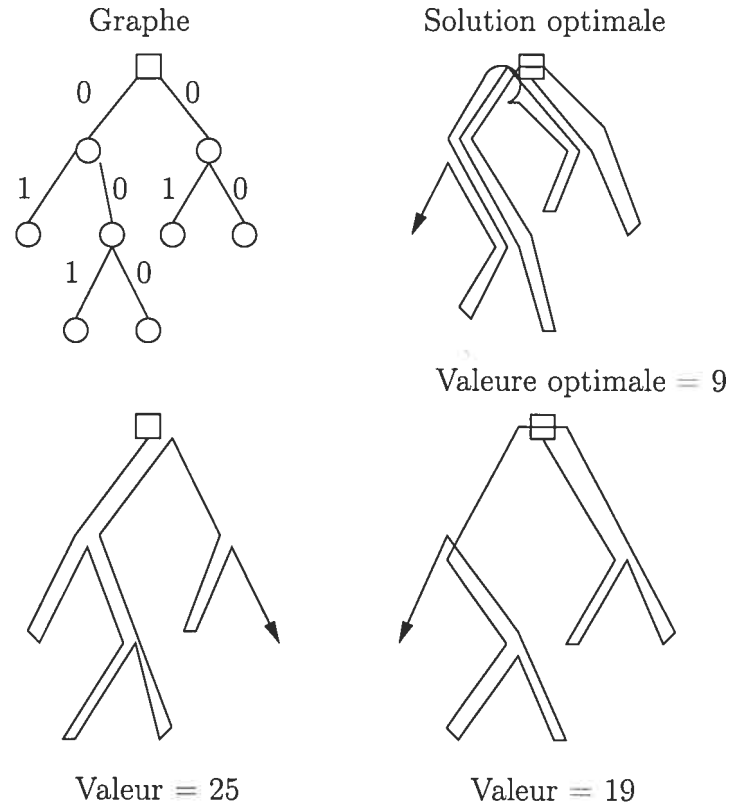


FIG. 5.5 – Une recherche en profondeur ne donne pas une solution optimale dans le cas général.

Deux recherches en profondeur (en bas) donnent deux résultats différents pour un arbre avec des coûts quelconques. La solution optimale a un coût de 9 et une version est montrée en haut à droite.

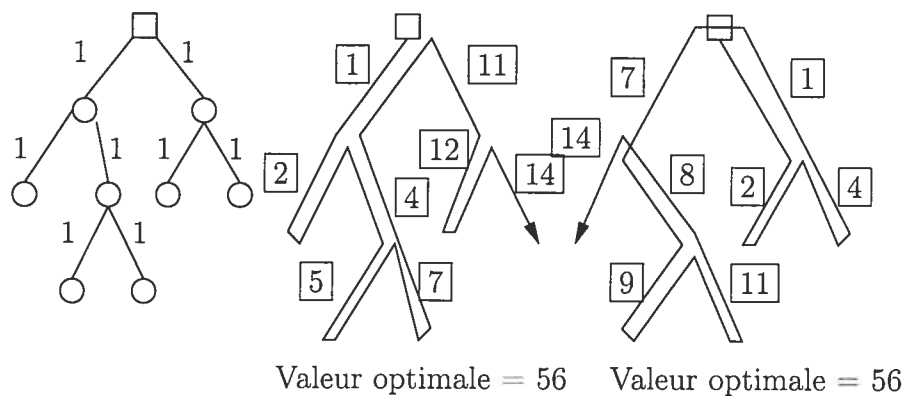


FIG. 5.6 – Pour l'arbre unitaire, les recherches en profondeur donnent le même résultat.

Deux recherches en profondeur différentes donnent le même résultat. A côté de chaque arête figure son coût cumulé.

Si ces deux points sont vérifiés alors n'importe quelle recherche en profondeur donnera la même valeur à la fonction objectif. Soit une recherche en profondeur. La valeur de la fonction objectif vaut $\sum_{i=1}^n (2i - p(i^{\text{ième}} \text{ arête}))$. Réarrangeons les termes en $\sum_{i=1}^n 2i + \sum_{i=1}^n p(i^{\text{ième}} \text{ arête})$. La première somme est invariable mais la deuxième aussi, car elle ne dépend que de la profondeur de chacune des arêtes.

Passons maintenant à la démonstration des deux premiers points. Ceux-ci sont aisément vérifiables si l'on pense à revenir au dépôt après avoir visité la $i^{\text{ième}}$ arête. Une recherche en profondeur atteint exactement cette borne car elle ne repasse jamais plus de deux fois par une arête.

□

La figure 5.7 illustre le théorème, ainsi que sa démonstration pour l'arbre de la figure 5.6.

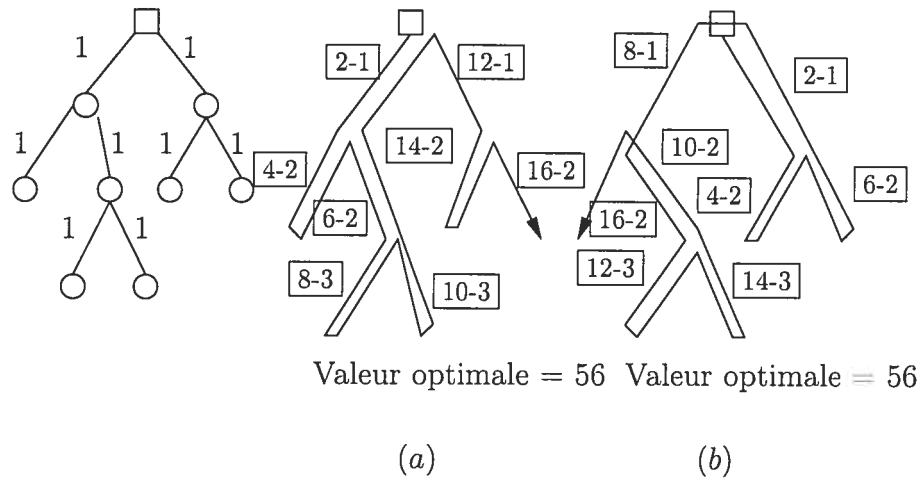


FIG. 5.7 – Le calcul des coûts cumulatifs pour deux recherches en profondeur.

A côté de chaque arête se trouve explicité $2i - p(i^{\text{ième}} \text{ arête})$. Pour la recherche en profondeur (a) nous avons $2 + 4 + \dots + 14 + 16 - \{1 + 2 + 3 + 3 + 2 + 1 + 2 + 2\} = 56$ et pour (b) : $2 + 4 + \dots + 14 + 16 - \{1 + 2 + 2 + 1 + 2 + 3 + 3 + 2\} = 56$.

Chapitre 6

La grille unitaire

Le cas de la grille est particulièrement intéressant car d'une part, il est typiquement un problème de graphe sur les arêtes et non sur les nœuds et d'autre part, il peut servir à modéliser une ville nord-américaine.

Nous n'avons traité que le cas unitaire car nous pensons que le cas général de la grille est NP-difficile bien que nous n'ayons pas réussi à le démontrer.

Nous traitons d'abord le cas de la bande unitaire pour ensuite nous aventurer dans les méandres du cas plus général de la grille unitaire. Pour commencer, nous situons le dépôt dans un des coins comme expliqué dans les deux premières sections. Nous terminons par une discussion de la localisation du dépôt.

6.1 La bande unitaire

Définissons la bande unitaire. Ce type de graphes particulier est illustré sur la figure 6.1.

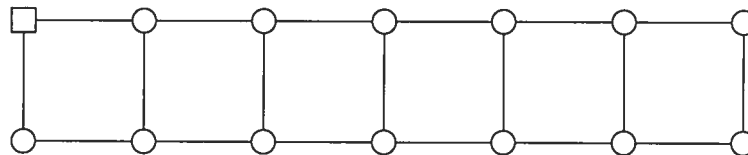


FIG. 6.1 – Une bande unitaire.

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.

Avant de démontrer le résultat, à savoir que le PPCC dans le cas de la bande unitaire est résoluble en temps polynomial, redéfinissons le concept de *chemin simple*. Rappelons qu'un *chemin simple* est un chemin qui ne repasse pas par une de ses arêtes (nomenclature de Berge [4]). Pour nous il s'agira d'un chemin partant du dépôt qui ne repasse pas par une de ses arêtes. Le *plus long chemin simple* est un chemin simple de longueur maximale. La figure 6.2 présente les deux plus longs chemins simples correspondants à la figure 6.1.

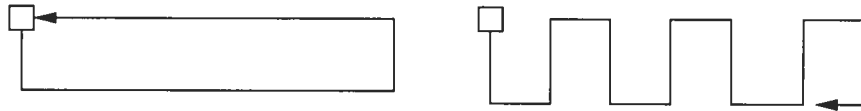


FIG. 6.2 – Deux plus longs chemins simples pour la bande unitaire.

Deux plus longs chemins simples pour la bande unitaire $s = 6$. Leur longueur est de 14.

Pour la bande unitaire $s = k$, la longueur d'un tel chemin vaut toujours $2k + 2$. Pour s'en convaincre, il suffit de partir du dépôt et d'envisager les deux (ou le seul cas possible dans les coins différents du dépôt) cas possibles à chaque bifurcation possible. La figure 6.3 présente le début de l'arbre de toutes les possibilités. Nous voyons que le nœud encadré mène à une impasse. En continuant ainsi l'arbre et en tenant compte toutes les possibilités, on trouve qu'il n'y a que les deux plus longs chemins illustrés sur la figure 6.2.

Passons maintenant au résultat et démontrons-le.

Théorème 6.1.1 *Le PPCC dans le cas de la bande unitaire est polynomial en temps $O(n)$*

Démonstration

La preuve se fait en exhibant la structure de la solution optimale et en comparant son coût à toutes les autres solutions possibles. Plus exactement en bornant inférieurement le coût de toutes les autres solutions et en montrant que cette borne inférieure est plus grande que le coût de nos deux candidats à l'optimalité. Les deux types de solutions optimales

sont proposées sur la figure 6.4 pour la bande unitaire de la figure 6.1.

Les solutions optimales décrites sur la figure 6.4 ont un coût de $5k^2 + 4k + 1$. Il s'agit en fait des solutions construites à partir des plus longs chemins simples. Ces solutions sont obtenues non seulement en prenant les plus longs chemins mais aussi en utilisant l'augmentation minimale (illustrée sur la figure 6.5) pour rendre tous les sommets de degré pair sauf le sommet de départ (le dépôt) et le sommet d'arrivée. Ces deux sommets ne doivent pas être de degré pair puisque nous cherchons un chemin et non un circuit.

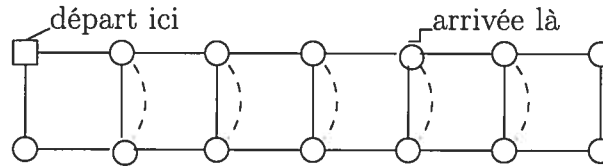


FIG. 6.5 – Augmentation minimale pour rendre les sommets de degré pair.

L'augmentation minimale pour rendre tous les sommets (sauf le dépôt et le sommet d'arrivée) de degré pair. Quel que soit le sommet d'arrivée, l'augmentation minimale consiste en la duplication des $k - 1$ arêtes centrales.

N'importe quelle solution doit commencer par un chemin simple. Si on construit la solution en commençant par le plus long chemin simple, les solutions les moins coûteuses sont celles décrites sur la figure 6.4 puisqu'elles utilisent une augmentation minimale.

Envisageons toutes les autres solutions. Une solution commencera nécessairement par un chemin simple de longueur $1 \leq l \leq 2k + 1$. Nous savons aussi qu'au moins $k - 1$ arêtes ont un coût relatif de 2 et que la dernière arête parcourue a un coût relatif de 1 (elle ne doit être parcourue qu'une seule fois) et ceci pour n'importe quelle solution.

Soit une telle solution. Bornons inférieurement son coût en constatant la chose suivante :

$$\underbrace{\frac{l(l+1)}{2}}_I + \underbrace{l+2}_{II} + \underbrace{\sum_{i=1}^{2k-l+1} (l+2) + i}_{III} + \underbrace{\sum_{j=1}^{k-2} (2k+3) + 2j}_{IV} + \underbrace{4k}_V \leq \text{coût d'une solution}$$

commençant par un chemin simple de longueur $1 \leq l \leq 2k + 1$.

Le tableau I reprend les différents termes de la borne inférieure et les décrit en détails.

I	:	$\frac{l(l+1)}{2}$:	La somme au sens cumulatif des l premières arêtes. Elles ont toutes un coût relatif égal à 1 et elles comptent pour $1 + 2 + 3 + \dots + l = \frac{l(l+1)}{2}$,
II	:	$l + 2$:	La $(l + 1)^{\text{ème}}$ arête au coût relatif égal à 2 (sinon le chemin simple serait de longueur supérieure à l) qui donne un coût de $l + 2$,
III	:	$\sum_{i=1}^{2k-l+1} (l + 2) + i$:	Au mieux, les $k - 2$ arêtes au coût relatif égal à 2 restantes peuvent être explorées en dernier dans le parcours (ce que la structure du graphe ne nous permet pas, mais nous cherchons une borne inférieure et non une solution réalisable). Il reste donc $3k + 1 - l - 1 - (k - 2) - 1 = 2k - l + 1$ arêtes de coût relatif égal à un,
IV	:	$\sum_{j=1}^{k-2} (2k + 3) + 2j$:	Les $k - 2$ arêtes de coût relatif égal à 2 qui sont placées à la fin,
V	:	$4k$:	La dernière arête qui doit avoir un coût relatif égal à un. Son coût est de $2k + 3 + 2(k - 2) + 1 = 4k$.

TAB. I – Description des termes de la borne inférieure pour la bande unitaire.

Après de brefs calculs, la borne inférieure devient

$$\underbrace{\frac{l(l+1)}{2}}_I + \underbrace{l+2}_{II} + \underbrace{\frac{2k-l+1}{2}(2k+l+6)}_{III} + \underbrace{(k-2)(3k+2)}_{IV} + \underbrace{4k}_V = 5k^2 + 7k - l + 1.$$

Or $1 \leq l \leq 2k + 1$. Cette borne inférieure est la plus petite quand $l = 2k + 1$ et vaut alors $5k^2 + 5k$. Comparons-là à la valeur de la solution optimale. Nos candidats ont pour valeur $5k^2 + 4k + 1$ qui est une valeur plus petite que la borne inférieure en autant que $k \geq 1$. Nos deux candidats sont donc des solutions optimales.

□

6.2 La grille unitaire

Définissons la grille unitaire (k, l) comme suit : il s'agit d'une grille dont toutes les arêtes ont un coût de un, elle possède k arêtes verticalement et l arêtes horizontalement. Le dépôt se situe, comme pour la bande unitaire, dans un coin extrême que nous situons en haut à gauche. Pour des raisons de symétrie évidente, nous ne nous occupons que des grilles unitaires (k, l) avec $k \leq l$ et $k \geq 2$.

La figure 6.6 reprend une grille unitaire (2,4).

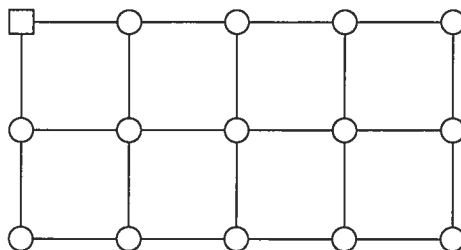


FIG. 6.6 – Une grille unitaire.

Une grille unitaire (2,4). Le dépôt est situé en haut à gauche.

La technique qui a prévalu pour la bande unitaire, à savoir la construction d'une solution optimale à partir du plus long chemin unitaire, n'est pas d'application ici comme le montre la figure 6.7 pour la grille unitaire 3×3 .

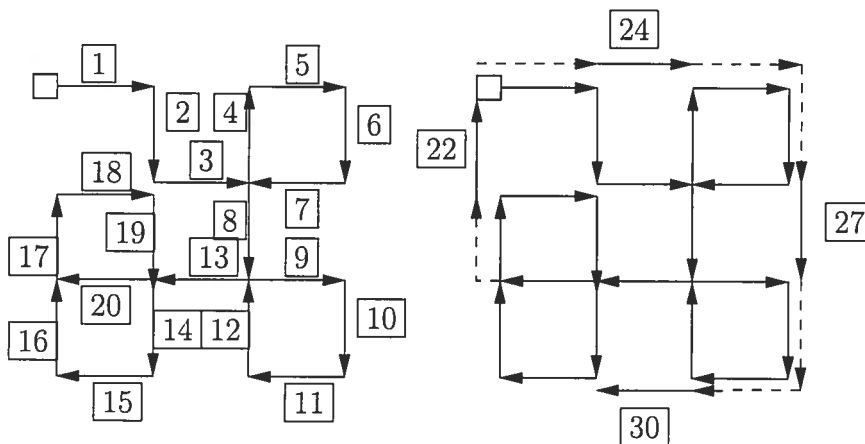


FIG. 6.7 – Un exemple de grille unitaire dont la solution optimale ne commence pas par un chemin simple le plus long.

Pour la grille unitaire 3×3 , il existe un chemin de longueur 20 ne se recoupant pas (chemin simple) mais la solution élaborée à partir de ce chemin n'est pas optimale. Elle coûte 313.

La solution optimale de la grille unitaire 3×3 est montrée sur la figure 6.8.

Néanmoins, l'idée du remplissage en zigzag comme sur la figure 6.9 est valable pour la grille unitaire (k,l) .

Ce remplissage est optimal pour les arêtes intérieures et si nous remplissons la grille de cette façon en minimisant le service des arêtes extérieures qui n'ont pas

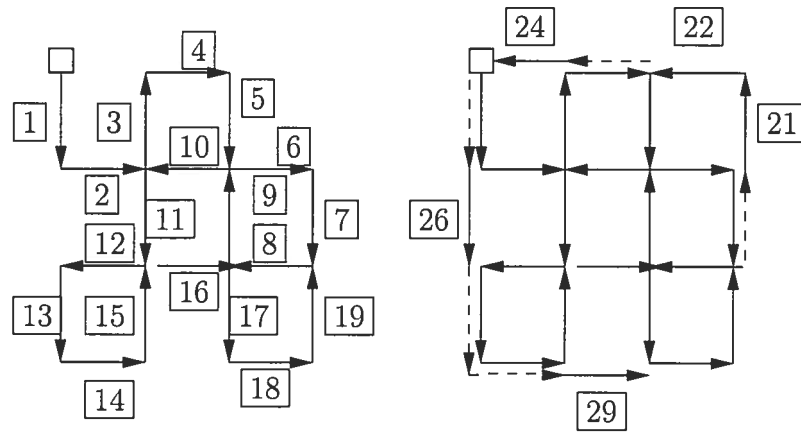


FIG. 6.8 – La solution optimale pour la grille 3×3 .

Pour la grille unitaire 3×3 , la solution optimale débute par un chemin de longueur 19 ne se recoupant pas. Elle coûte 312.

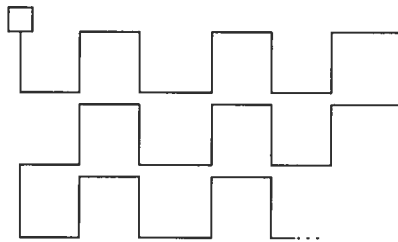


FIG. 6.9 – Comment remplir optimalement l'intérieur d'une grille.

Pour remplir optimalement l'intérieur d'une grille unitaire (k,l) , il faut parcourir cet intérieur en zigzagant.

encore été desservies, nous conjecturons que nous obtenons une solution optimale.

Conjecture 6.2.1 *Le PPCC dans le cas de la grille unitaire est polynomial et résoluble en temps $O(n)$.*

Plusieurs cas sont à distinguer suivant que k et l soient pairs ou impairs ($k \leq l$) et nous devons exclure le cas particulier de la grille unitaire $(2,2)$. La figure 6.10 résume notre conjecture concernant la grille unitaire.

Le cas de la grille unitaire 2×2 est particulier et notre conjecture ne tient pas dans ce cas comme le montre la figure 6.11.

La particularité du cas de la grille unitaire $(2,2)$ tient à ce que l'intérieur de la grille $(2,2)$ est vide au sens des arêtes (toutes ses arêtes touchent la frontière). Le remplissage en zig-zag n'est efficace que s'il y a des arêtes intérieures. La grille $(2,2)$ est la seule dont l'intérieur soit vide parmi les grille (k,l) avec $k,l \geq 2$.

6.3 Importance de la localisation du dépôt

Le choix du dépôt en haut à gauche n'est pas anodin. C'est le seul cas que nous avons réussi à traiter dans le cas général. Pour la bande unitaire, comme nous le verrons, le choix de la localisation du dépôt n'est pas très important, car les problèmes restent assez similaires quel que soit le sommet de départ de la tournée.

6.3.1 La bande unitaire

Pour des raisons de symétrie, nous nous ne occupons que des dépôts localisés en haut à gauche près du bord gauche de telle sorte qu'il y a moins d'arêtes à la gauche du dépôt qu'à sa droite.

Si le dépôt se trouve en deuxième position, la tournée optimale est très simple à obtenir comme le suggère la figure 6.12.

En effet, le carré sur la gauche peut être parcouru sans duplication, puis le reste du graphe peut être parcouru optimalement comme nous l'avons vu dans le cas de la bande unitaire avec le dépôt situé en haut à gauche. Il est à noter que l'augmentation minimale pour rendre les sommets autres que celui de départ et d'arrivée pairs comprend une arête de moins que lorsque le dépôt était situé dans le coin supérieur gauche. Dans ce cas-ci, la solution optimale nous permet d'utiliser exactement cette augmentation minimale.

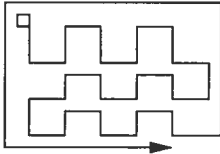
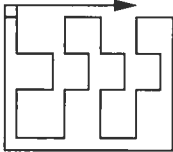
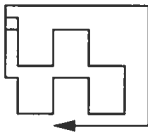
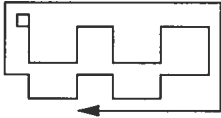
$k \backslash l$	Impair	Pair
Impair	 3×5	 3×4
Pair	 2×3	 2×4

FIG. 6.10 – Notre conjecture pour la grille unitaire.

Notre conjecture pour le cas général des grilles unitaires $k \times l$ consiste à parcourir l'intérieur des grilles en zigzaguant puis de terminer en desservant les arêtes de la frontière. Le cas 2×2 en est exclu.

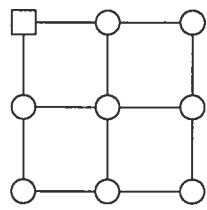
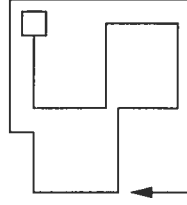
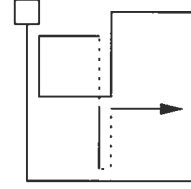
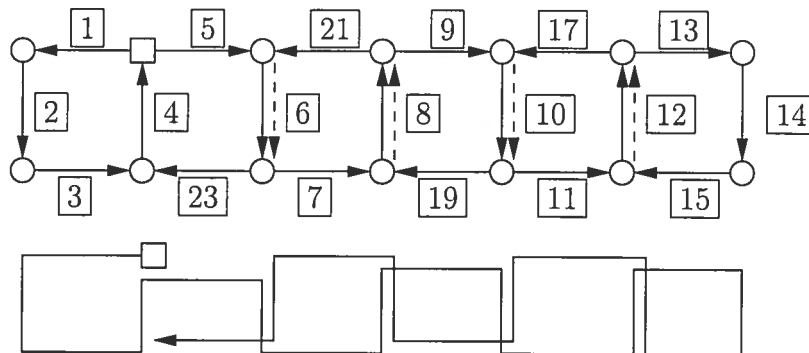
grille 2×2 .La solution suivant
la conjecture pour un
coût de 85La solution optimale
pour un coût de 84.FIG. 6.11 – *Le cas 2×2 .*

FIG. 6.12 – Une solution optimale quand le dépôt est au deuxième sommet.

La solution optimale utilise non seulement le plus long chemin simple mais aussi strictement l'augmentation minimale pour rendre tous les sommets de degré pair.

S'il y a plus d'un carré à la gauche du dépôt, une solution optimale consiste à d'abord parcourir les arêtes du bord de ces carrés, puis à zigzaguer pour couvrir les arêtes à droite du dépôt comme dans le cas de la bande unitaire avec le dépôt en haut à gauche pour finalement terminer par les arêtes *intérieures* des carrés à gauche du dépôt. La construction de cette solution optimale est illustrée sur la figure 6.13 dans le cas où il y a deux carrés à gauche du dépôt.

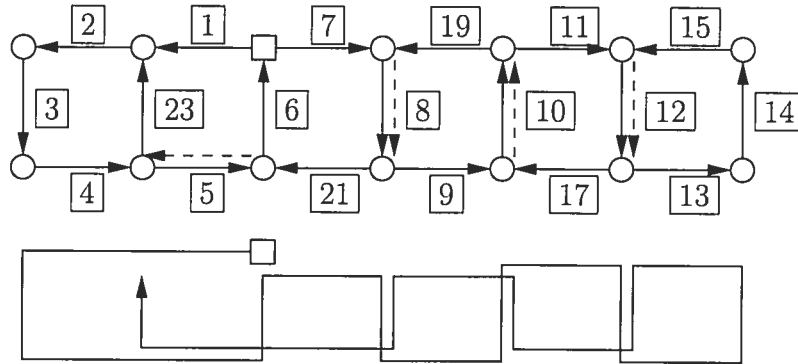


FIG. 6.13 – Une solution optimale quand le dépôt est au troisième sommet.

La tournée commence par parcourir le bord des deux carrés à gauche pour ensuite zigzaguer à droite comme dans le cas où le dépôt est en haut à gauche pour finalement parachever le service des arêtes non desservies des carrés à gauche du dépôt.

En effet, en parcourant le graphe comme indiqué, la solution optimale commence par le plus long chemin simple et utilise une augmentation minimale. Ceci nous donne une bonne assurance qu'il s'agit bien d'une solution optimale mais ne constitue pas une démonstration. Néanmoins, l'argument utilisé dans la démonstration du cas de la bande unitaire avec le dépôt en haut à gauche peut être repris ici.

La tournée la plus coûteuse au sens cumulatif part du sommet en haut à gauche. En effet, c'est le seul cas où l'augmentation minimale a une arête de plus que les autres cas.

6.3.2 La grille unitaire

Comme annoncé dans le préambule de cette section, nous n'avons pas de résultat. Si nous situons le dépôt en un sommet intérieur nous compliquons le problème. En effet, à partir d'un tel sommet la tournée peut partir dans trois (si le sommet est situé sur le bord) ou quatre (si le sommet est un sommet intérieur) directions différentes augmentant ainsi le nombre de possibilités. Bien que la recherche d'une solution optimale nous semble plus difficile dans ce cas, nous conjecturons que le PPCC dans ce cas reste résolvable en temps polynomial.

Conclusions

L'objectif principal de ce mémoire était de proposer un nouveau problème ainsi que de tenter une première étude de ce problème. Notre motivation pour l'introduction de ce problème était un souci de prendre en considération l'attente des clients dans le PPC. Ce désir s'est traduit par un caractère cumulatif dans la fonction objectif et a donné naissance au PPCC.

Notre définition du problème est en rupture avec la tradition du PPC mais comme nous l'avons vu, notre définition n'est en rien restrictive. Au contraire, elle nous permet d'englober un problème plus vaste et si nous voulons nous restreindre à une tournée qui part et finit au dépôt, une légère adaptation du problème nous permet de le faire.

Tout comme la version cumulative du TSP est bien plus complexe que le TSP, la version cumulative du PPC est bien plus complexe que le PPC. En fait, le paradigme du PPC ne nous est pas d'une grande aide pour résoudre le PPCC. Il faut développer de nouveaux outils pour aborder ce problème. Nous avons lancé quelques nouvelles pistes dans ce mémoire.

Dans l'introduction nous avons repris les quatre questions essentielles auxquelles nous désirions apporter une réponse ou un début de réponse dans ce mémoire. Elles sont tellement importantes et centrales dans notre travail que nous les formulons à nouveau mais cette fois-ci en les commentant avec les résultats présentés dans ce mémoire.

Quelle est la complexité du problème?

Nous avons démontré que le PPCC est NP-difficile. Mieux, notre démonstration prouve que ce problème est NP-difficile sur les arbres. Mieux encore, notre démonstration permet de déduire que cette complexité est structurelle. Le PPCC est fortement NP-difficile.

Que le problème soit fortement NP-difficile ne devrait pas tellement surprendre. En effet, le caractère cumulatif de la fonction objectif impose une solution globale dans le sens que le coût d'une arête dépend du parcours précédent celle-ci et donc des autres arêtes. Ceci implique qu'un petit changement local dans la structure du graphe peut avoir de grandes répercussions sur le type de solution. Il serait intéressant de pousser plus loin l'étude de la complexité de ce problème.

Comment pouvons-nous modéliser le PPCC?

Nous avons présenté un modèle linéaire en nombres entiers. Certes, il a le mérite d'exister, mais il est tellement complexe qu'il semble illusoire de pouvoir l'utiliser pour résoudre des problèmes concrets. Bien qu'il n'ait pas été étudié en détails, la recherche d'un autre modèle plus simple semble souhaitable. Toutefois, il permet d'obtenir une borne inférieure qui tient compte de la structure du graphe en un temps raisonnable. Il serait aussi intéressant de voir ce que devient ce modèle sur des types de graphes particuliers comme par exemple les arbres. Il y a peut-être moyen de simplifier ce modèle sur ce type de graphes.

Peut-on développer un algorithme exact pour le cas général?

Nous en avons présenté deux. Le premier, un algorithme d'énumération implicite, profite grandement du caractère cumulatif de la fonction objectif. Plus les coûts des arêtes sont différents, moins de solutions devront être explicitement envisagées. Le second, une méthode de séparation et évaluation progressive, nous semble beaucoup plus prometteur. Une étude plus poussée du modèle et surtout une meilleure borne supérieure permettront d'améliorer cette approche mais comme le problème est fortement NP-difficile, on peut se demander si le jeu en vaut la chandelle. Sans doute est-il préférable de se tourner vers les heuristiques pour solutionner le problème.

Peut-on identifier des cas polynomiaux?

Nous en avons identifié essentiellement quatre : la ligne droite, le cercle, l'arbre unitaire et possiblement la grille unitaire. Les trois premiers cas sont des adaptations d'algorithmes existants pour le TSP cumulatif. Le PPCC sur la bande unitaire est résolvable en temps polynomial comme nous l'avons démontré et nous conjecturons que c'est aussi le cas pour la grille unitaire. Des recherches plus approfondies sur la grille montreraient sans doute que le PPCC sur la grille générale est NP-difficile.

Il est une cinquième question importante pour tout problème NP-difficile :

Existe-t-il une heuristique avec une borne de pire cas?

Nous n'avons pas encore de réponse pour l'instant mais il serait intéressant de voir dans quelle mesure l'algorithme d'approximation avec borne de pire cas pour le TSP cumulatif peut être adapté au PPCC.

Notre recherche nous a amené à développer trois outils supplémentaires : deux bornes inférieures et une heuristique.

La borne algébrique est extrêmement simple mais très rapide à évaluer. De plus, nous avons vu qu'elle peut donner de bons résultats puisque sa valeur coïncidait avec celle de la deuxième borne obtenue par relaxation du modèle dans l'exemple choisi. Cette dernière borne est meilleure puisqu'elle fait appel au modèle et donc à la structure du graphe contrairement à la borne algébrique. Cette deuxième borne semble être calculable assez rapidement. Une étude plus détaillée du modèle amènera certainement des améliorations à celle-ci.

L'heuristique proposée a le grand avantage d'être très rapide et facilement implémentable. Il va de soi qu'il faudra rechercher des heuristiques plus fines mais sans doute seront-elles beaucoup plus lourdes.

Passons maintenant en revue trois grands axes de recherche qu'il serait intéressant d'explorer dans un futur proche.

Comme en général pour les problèmes NP-difficiles et à fortiori les problèmes fortement NP-difficiles, il faut se tourner vers les heuristiques pour trouver des méthodes de résolution pratiques.

Une autre grande avenue de recherche est la comparaison entre le PPCC et les différentes versions du PPC. Quelles sont les qualités et avantages des solutions fournies par le PPCC par rapport aux solutions des autres versions du PPC?

Finalement, une troisième possibilité est de généraliser le PPCC. Plusieurs possibilités sont envisageables. Tout d'abord étudier la version plus réaliste où l'on distingue deux (voire trois) coûts par arêtes. Un coût de traitement de l'arête et un autre de passage (quand l'arête est déjà traitée ou non). Ensuite, on peut ajouter les contraintes des PPC classiques au PPCC. Ainsi, on pourrait étudier une version du PPCA à la sauce cumulative. La version dirigée, mixte et rurale vaudraient, elles aussi, le détour. Enfin, il serait sans doute intéressant de voir le PPCC dans une perspective plus globale. Comme tel, le PPCC fait partie d'une famille de problèmes

dépendant du temps. En effet, le coût des arêtes dépend en fait du parcours suivi pour la tournée, c'est-à-dire du moment où l'arête est desservie. Le coût cumulatif de chaque arête est essentiellement le moment où elle est desservie multiplié par son coût si on ne tient pas compte des diamètres. Il pourrait être intéressant d'étudier plus généralement les problèmes où le coût final d'une arête dépendrait linéairement du temps ainsi que de son coût original. Peut-être y aurait-il moyen de trouver des grandes familles de problèmes qui se distinguent les unes des autres structurellement ?

Bibliographie

- [1] AFRATI, F., S. COSMADAKIS, C. PAPADIMITRIOU, G. PAPAGEORGIOU et N. PAKOSTANTINO, «The complexity of the traveling repairman problem», *Revue d'Automatique, d'Informatique et de Recherche Opérationnelle, Informatique Théorique et Applications*, vol. 20, 1986, pp. 79–87.
- [2] ARCHER, A. et D. P. WILLIAMSON, «Faster approximation algorithms for the minimum latency problem», citeseer.nj.nec.com/archer03faster.html, 2003.
- [3] ARORA, S. et G. KARAKOSTAS. «A $2+\epsilon$ approximation algorithm for the k -MST problem». In *Symposium on Discrete Algorithms* (2000), pp. 754–759.
- [4] BERGE, *Graphes et hypergraphes*. Dunod, Paris, 1970.
- [5] BIANCO, L., A. MINGOZZI et S. RICCIARDELLI, «The traveling salesman problem with cumulative costs», *Networks*, vol. 23, no. 2, 1993, pp. 81–91.
- [6] BLUM, A., P. CHALASANI, D. COPPERSMITH, B. PULLEYBLANK, P. RAGHAVAN et M. SUDAN, «The minimum latency problem», *Proc. 26th Association on Computing Machinery Symposium on the Theory of Computing*, 1994, pp. 163–171.
- [7] CHRISTOFIDES, N., *Graph Theory. An Algorithmic Approach*. Academic Press, London, 1975.
- [8] EDMONDS, J. et L. JOHNSON, «Matching, euler tours and the chinese postman problem», *Mathematical Programming.*, vol. 5, 1973, pp. 88–124.
- [9] EISELT, H. A., M. GENDREAU et G. LAPORTE, «Arc routing problems, part I: The chinese postman problem», *Operations Research*, vol. 43, no. 2, 1995, pp. 231–242.
- [10] FISCHETTI, M., G. LAPORTE et S. MARTELLO, «The delivery man problem and cumulative matroids», *Operations Research*, vol. 41, no. 6, 1993, pp. 1055–1064.
- [11] FREDERICKSON, G. N., «Approximation algorithms for some postman problems», *Journal of the Association for Computing Machinery*, vol. 26, no. 3, 1979, pp. 538–554.
- [12] GARCÍA, A., P. JODRÁ et J. TEJEL, «A note on the travelling repairman problem», *Pre-Publicaciones Del Seminario Matematico Garcia De Galdeano*, no. 3, 2001, pp. 1–16.
- [13] GARG, N., «A 3-approximation for the minimum tree spanning k vertices», *Proc. 37th Institute of Electrical and Electronics Engineers Symposium on Foundations of Computer Science*, 1996.

- [14] GOEMANS, M. et J. KLEINBERG, «An improved approximation ratio for the minimum latency problem», *Mathematical Programming*, vol. 82, 1998, pp. 111–124.
- [15] GOEMANS, M. et D. WILLIAMSON, «A general approximation technique for constrained forest problems», *Society for Industrial and Applied Mathematics Journal on Computing*, vol. 24, 1995, pp. 296–317.
- [16] LUCENA, A., «Time-dependent traveling salesman problem - the deliveryman case», *Networks*, vol. 20, no. 6, October 1990, pp. 753–763.
- [17] DROR (EDITOR), M. et AL., *Arc Routing: Theory, Solutions and Applications*. Kluwer Academic Publishers, 2000.
- [18] MINIEKA, E., «The delivery man problem on a tree network», *Annals of Operations Research*, vol. 18, no. 1-4, february 1989, pp. 261–266.
- [19] SAHNI, S. et T. GONZALEZ, «P-complete approximation problems», *Journal of the Association for Computing Machinery*, vol. 23, 1976, pp. 555–565.
- [20] SITTEERS, R., «The minimum latency problem is NP-hard for weighted trees», *9th conference of Integer Programming and Combinatorial Optimization*, 2002, pp. 230–239.
- [21] VAN AARDENNE-EHRENFEST, T. et N. G. DE BRUIJN, «Circuits and trees in oriented linear graphs», *Simon Stevin*, vol. 28, 1951, pp. 203–217.

Annexes

Modèle linéaire du PPCC de l'exemple simple

Voici le modèle pour l'exemple de la section 1.3. Un programme établit automatiquement toutes les contraintes à partir d'une liste d'arêtes donnés sous la forme $A B 1$ où A et B sont les sommets de l'arête et 1 son coût. Une arête est alors automatiquement désignée par la concaténation des deux noms de ses sommets. Le tableau II donne l'équivalence entre les noms des arêtes de cet exemple.

Noms pratiques		Noms théoriques
AB	=	a
BC	=	b
AC	=	c

TAB. II – La conversion entre les noms des arêtes.

```
\Cumulative Chinese Postman Problem linear model for the graph
\written in "graph.txt".
\This file was automatically generated by CplexGenerator.
\Generating 39 variables and 60 constraints.
```

Minimize

```
obj: 3 d1AC + 2 d2AC + d3AC +
150 d1BC + 100 d2BC + 50 d3BC +
3 d1AB + 2 d2AB + d3AB
```

Subject To

```
c1.1: v1AC + v2AC + v3AC = 1
c1.2: v1BC + v2BC + v3BC = 1
c1.3: v1AB + v2AB + v3AB = 1
```

```
c2.1: v1AC + v1BC + v1AB = 1
c2.2: v2AC + v2BC + v2AB = 1
c2.3: v3AC + v3BC + v3AB = 1
```

$$c3.1AC: v1AC - d1AC \leq 0$$

$$c3.2AC: v2AC - d2AC \leq 0$$

$$c3.3AC: v3AC - d3AC \leq 0$$

$$c3.1BC: v1BC - d1BC \leq 0$$

$$c3.2BC: v2BC - d2BC \leq 0$$

$$c3.3BC: v3BC - d3BC \leq 0$$

$$c3.1AB: v1AB - d1AB \leq 0$$

$$c3.2AB: v2AB - d2AB \leq 0$$

$$c3.3AB: v3AB - d3AB \leq 0$$

$$c4.AC2.1: v2AC + d1AC \leq 1$$

$$c4.AC3.1: v3AC + d1AC \leq 1$$

$$c4.AC3.2: v3AC + d2AC \leq 1$$

$$c4.BC2.1: v2BC + d1BC \leq 1$$

$$c4.BC3.1: v3BC + d1BC \leq 1$$

$$c4.BC3.2: v3BC + d2BC \leq 1$$

$$c4.AB2.1: v2AB + d1AB \leq 1$$

$$c4.AB3.1: v3AB + d1AB \leq 1$$

$$c4.AB3.2: v3AB + d2AB \leq 1$$

$$c5.1A: d1AC + d1AB - 2 D1A - V0A - V1A = 0$$

$$c5.2A: d2AC + d2AB - 2 D2A - V1A - V2A = 0$$

$$c5.3A: d3AC + d3AB - 2 D3A - V2A - V3A = 0$$

$$c5.1C: d1AC + d1BC - 2 D1C - V0C - V1C = 0$$

$$c5.2C: d2AC + d2BC - 2 D2C - V1C - V2C = 0$$

$$c5.3C: d3AC + d3BC - 2 D3C - V2C - V3C = 0$$

$$c5.1B: d1AB + d1BC - 2 D1B - V0B - V1B = 0$$

$$c5.2B: d2AB + d2BC - 2 D2B - V1B - V2B = 0$$

$$c5.3B: d3AB + d3BC - 2 D3B - V2B - V3B = 0$$

$$c6.1: d1AC + d1BC + d1AB - D1A - D1C - D1B = 1$$

$$c6.2: d2AC + d2BC + d2AB - D2A - D2C - D2B = 1$$

$$c6.3: d3AC + d3BC + d3AB - D3A - D3C - D3B = 1$$

$$c7.A1: D1A + V0A + V1A \leq 1$$

$$c7.A2: D2A + V1A + V2A \leq 1$$

$$c7.A3: D3A + V2A + V3A \leq 1$$

$$c7.C1: D1C + V0C + V1C \leq 1$$

$$c7.C2: D2C + V1C + V2C \leq 1$$

$$c7.C3: D3C + V2C + V3C \leq 1$$

$$c7.B1: D1B + V0B + V1B \leq 1$$

$$c7.B2: D2B + V1B + V2B \leq 1$$

$$c7.B3: D3B + V2B + V3B \leq 1$$

$$c8.1: V1A + V1C + V1B = 1$$

$$c8.2: V2A + V2C + V2B = 1$$

$$c8.3: V3A + V3C + V3B = 1$$

$$c9.1AC: v1AC - V1A - V1C \leq 0$$

$$c9.1BC: v1BC - V1C - V1B \leq 0$$

$$c9.1AB: v1AB - V1A - V1B \leq 0$$

$$c9.2AC: v2AC - V2A - V2C \leq 0$$

$$c9.2BC: v2BC - V2C - V2B \leq 0$$

$$c9.2AB: v2AB - V2A - V2B \leq 0$$

$$c9.3AC: v3AC - V3A - V3C \leq 0$$

$$c9.3BC: v3BC - V3C - V3B \leq 0$$

$$c9.3AB: v3AB - V3A - V3B \leq 0$$

$$c10A: V0A = 1$$

$$c11C: V0C = 0$$

$$c11B: V0B = 0$$

BINARIES

v1AC v1BC v1AB

v2AC v2BC v2AB

v3AC v3BC v3AB

d1AC d1BC d1AB

d2AC d2BC d2AB

d3AC d3BC d3AB

V0A V0C V0B
V1A V1C V1B
V2A V2C V2B
V3A V3C V3B

D1A D1C D1B
D2A D2C D2B
D3A D3C D3B

End

Solution fournie par CPLEX pour l'exemple simple

Voici la solution fournie par CPLEX pour l'exemple de la section 1.3. La remarque de la section 6.3.2 introduisant le modèle de cet exemple explique pourquoi le nom des arêtes est constitué des deux sommets de l'arête.

```
Problem 'graph.txt.lp' read.
Read time = 0.01 sec.
Tried aggregator 1 time.
MIP Presolve eliminated 7 rows and 5 columns.
MIP Presolve modified 8 coefficients.
Aggregator did 2 substitutions.
Reduced MIP has 51 rows, 32 columns, and 153 nonzeros.
Presolve time = 0.01 sec.
Clique table members: 63
MIP emphasis: optimality
Root relaxation solution time = 0.00 sec.
Objective is integral.
```

Nodes		Objective	IInf	Best Integer	Cuts/		ItCnt	Gap
Node	Left				Best Node	Fractcuts:		
0	0	56.0000	8		56.0000		20	
		56.3333	15		Fractcuts: 8		25	
*		57.0000	0	57.0000	Fractcuts: 15		26	

Gomory fractional cuts applied: 3

```
Integer optimal solution: Objective = 5.7000000000e+01
Solution time = 0.01 sec. Iterations = 26 Nodes = 0
```

Variable Name	Solution Value
d2AC	1.000000
d3BC	1.000000
d1AB	1.000000
d2AB	1.000000
v2AC	1.000000
v3BC	1.000000
v1AB	1.000000
V0A	1.000000
D2A	1.000000
V2C	1.000000
V1B	1.000000
V3B	1.000000

All other variables in the range 1-39 are zero.

