



HAL
open science

Optimisation de réseaux de télécommunications avec sécurisation

Raja Rebai

► **To cite this version:**

Raja Rebai. Optimisation de réseaux de télécommunications avec sécurisation. Mathématiques [math].
Université Paris Dauphine - Paris IX, 2000. Français. NNT : . tel-00010841

HAL Id: tel-00010841

<https://theses.hal.science/tel-00010841>

Submitted on 31 Oct 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

présentée à

L'UNIVERSITE PARIS-IX DAUPHINE
U.F.R. Mathématiques de la décision

pour obtenir le titre de

DOCTEUR EN SCIENCES

Spécialité

MATHÉMATIQUES APPLIQUÉES

par

Raja RÉBAÏ

OPTIMISATION DE RÉSEAUX DE
TÉLÉCOMMUNICATIONS AVEC
SÉCURISATION

Soutenue le 10 Février 2000 devant le jury composé de :

Frédéric BONNANS	Directeur de thèse
Ivar EKELAND	Président
Jacek GONDZIO	Rapporteur
Mounir HADDOU	Examineur
Abdel LISSER	Examineur
Philippe MAHEY	Rapporteur
Claudia SAGASTIZÁBAL	Examineur
Jean-Philippe VIAL	Rapporteur

À la mémoire de mes très chers frères

Yacine et Hassène.

*“Toutes les déceptions de pensée et d’espérance, tout cela est secondaire.
Le seul malheur irréparable, c’est la mort de ceux qu’on aime.”*

Romain Rolland

Remerciements

Je voudrais tout d'abord exprimer ma profonde gratitude à mon directeur de thèse, Frédéric BONNANS, Directeur de recherche à l'INRIA, pour les moyens qu'il a mis à ma disposition tout au long de ma thèse ainsi que pour son soutien permanent. Sa confiance et son encouragement m'ont permis de mener à bien ce travail.

Ce travail s'est déroulé à l'INRIA-Rocquencourt dans le cadre d'un contrat de recherche entre le CNET et l'INRIA. Je tiens à exprimer mes vifs remerciements à Abdel LISSER, Chercheur au CNET, pour sa collaboration et sa confiance. Je lui suis extrêmement reconnaissante de m'avoir offert l'opportunité de découvrir le domaine des télécommunications.

Je suis très honorée par la présence au jury d'Ivar EKELAND, Professeur à l'Université Paris Dauphine. Je le remercie très sincèrement d'avoir accepté d'être président du jury.

Philippe MAHEY, Professeur à l'Université Blaise Pascal et Jean-Philippe VIAL, Professeur à l'Université de Genève, m'ont fait l'honneur de rapporter cette thèse. Je leur en remercie vivement. Je tiens à leur exprimer toute ma gratitude pour l'intérêt qu'ils ont bien voulu porter à ce travail.

Je voudrais remercier tout particulièrement Jacek GONDZIO, Professeur à l'Université d'Édimbourg, pour les discussions fructueuses que nous avons eues et ses conseils éclairés. Il a également accepté d'être rapporteur de ce travail. Je voudrais qu'il trouve ici le témoignage de ma reconnaissance.

J'ai le plaisir de remercier chaleureusement Claudia SAGASTIZABAL, Chercheur à l'INRIA, pour l'amitié qu'elle m'a toujours témoignée et pour sa relecture critique de mon manuscrit. Je souhaite que ce mémoire soit le témoignage de ma profonde gratitude.

Je tiens à exprimer ma plus vive reconnaissance à Mounir HADDOU, Maître de

conférence à l'Université d'Orléans, qui m'a souvent motivée et encouragée dans les moments de doute. Son optimisme et son enthousiasme ont été pour moi d'une valeur inestimable.

Le chapitre 3 de ce mémoire est un travail effectué en collaboration avec Cecilia POLA, Professeur à l'Université de Cantabrie. Je suis très heureuse de lui témoigner toute mon amitié.

C'est plus que des remerciements que j'adresse à toutes les personnes formidables que j'ai eu la chance de rencontrer à l'INRIA et plus particulièrement aux Bâtiments 12 et 13. Chacun saura reconnaître la part d'amitié que je lui porte.

Je ne saurais terminer sans mentionner tout ce que je dois à mes parents et à ma soeur qui m'ont toujours soutenue et supportée.

Mes pensées vont aussi à celui qui m'a particulièrement aidée à passer les périodes les plus difficiles. Sans lui, je n'aurais jamais pu aller jusqu'au bout.

Résumé

La première partie de cette thèse, concerne une étude de robustesse des algorithmes de points intérieurs prédicteurs correcteurs, ainsi qu'une approche par décomposition de cette méthode pour la résolution de problèmes de multiflot.

Dans la deuxième partie, nous nous intéressons au Problème de Sécurisation Globale dont l'objectif est de déterminer un multiflot (qui transporte toute demande de son noeud origine à son noeud destination en respectant la loi de Kirchhoff) et l'investissement de moindre coût en capacités nominale et de réserve qui assure le routage nominal et garantit sa survie par reroutage global.

Dans notre modèle les routages et les capacités peuvent être fractionnés. PSG se formule alors comme un problème linéaire de grande taille avec plusieurs niveaux de couplage. Sa structure particulière appelle à l'emploi d'algorithmes de décompositions.

Nous proposons quatre méthodes utilisant la technique de génération de colonnes. Les deux premières sont basées sur les techniques proximales. Leur tâche principale consiste en la résolution de sous problèmes quadratiques indépendants.

Le troisième algorithme s'inspire de l'approche de points intérieurs décrite à la première partie.

Pour finir, nous intégrons une procédure d'élimination de chemins dans une adaptation d'un solveur de points intérieurs.

Nous reportons des résultats numériques obtenus en testant ces algorithmes sur des données réelles fournies par le CNET.

Mots-clés :

Programmation linéaire de grande taille, méthodes de points intérieurs, décomposition, problèmes de multiflots, algorithme prédicteur correcteur, méthode de décomposition proximale, conception de réseaux de télécommunications, sécurisation de réseaux.

Abstract

The first part of this thesis, is concerned with the study of the robustness of the predictor corrector interior point algorithm and with a decomposition approach of this method for solving multicommodity network-flow problems.

In the second part, we are concerned with the Global Survivability in Telecommunication Networks (PSG). The objective consists in finding the optimal routing and the least cost investment in base and reserve capacities. The routing and the base capacity insure base traffic and the reserve capacity guarantees survivability of the traffic against any arc failure (using a global rerouting strategy).

In our model we consider that routings and capacities can be fractionated. So PSG can be formulated as a large-scale linear program. Its special structure favors the use of decomposition algorithms.

We propose four methods using columns generation technics.

The first two are based on proximal decomposition methods. The main task of these algorithms consist in solving independent quadratic subproblems.

The third algorithm is inspired by the interior point decomposition approach described in the first part.

Finally, we incorporate path elimination procedure into an adaptation of ready-to-use interior points code.

We report some numerical results obtained by testing these algorithms with data from the France-Telecom Paris district transmission network.

Key-words:

Large-scale linear programming, interior point methods, decomposition, multicommodity network flow models, predictor corrector algorithm, proximal decomposition method, network design problems, survivability.

Table des matières

Remerciements	iii
Résumé	v
Abstract	vi
1 Introduction	0
1.1 Objectifs généraux	0
1.2 Cadre général	0
1.2.1 Problèmes linéaires de multiflot	2
1.2.2 Méthodes de points intérieurs	3
1.2.3 Méthodes proximales	4
1.3 Contenu de la thèse	5
2 Quelques notions de la théorie des graphes	7
2.1 Généralités sur les graphes	7
2.2 Connexité	8
2.2.1 Point d'articulation et Isthme	8
2.2.2 La k-connexité	9
2.2.3 La k-arête-connexité	9
2.2.4 Définition d'un flot	10
2.2.5 Définition algébrique des flots	10
2.3 Multiflots	10
2.3.1 Définition d'un multiflot	10
2.3.2 Exemples de problèmes de multiflots	11
2.3.3 Définition d'un routage	11
2.4 Méthode de génération de colonnes	12
3 Algorithmes de suivi de chemin perturbé	15
3.1 Introduction	17

3.2	Main results	19
3.3	Proof of main results	24
3.3.1	Preliminary results	24
3.3.2	Asymptotic analysis	26
3.3.3	The perturbed predictor corrector algorithm in small neighborhoods	30
3.3.4	The perturbed predictor corrector algorithm using large neighborhoods	35
3.4	Numerical experiments	39
4	Une approche par décomposition des méthodes de points intérieurs pour les problèmes de multiflots	43
4.1	Introduction	48
4.2	Linear Optimization problems with coupling constraints	50
4.2.1	Problem formulation	51
4.2.2	solving linear optimization problems with coupling constraints using a predictor-corrector algorithm	52
4.2.3	The Newton Step	55
4.2.4	Resolution of the reduced system	55
4.3	Linear multicommodity network-flow problem	59
4.3.1	Framework	60
4.3.2	Node-Arc Formulation	61
4.3.3	Arc-path Formulation	62
4.4	Numerical Results	66
4.4.1	Randomly generated Problems	66
4.4.2	Tests With Real Data	67
4.5	Conclusion	69
5	Conception de réseaux de télécommunications	70
5.1	Motivation	70
5.2	Introduction	71
5.3	Sécurisation topologique	72
5.4	Dimensionnement d'un réseau et Routage	75
5.5	Sécurisation globale des réseaux de transmission	78

6	Modélisation du problème de sécurisation globale de réseaux	81
6.1	Introduction	81
6.2	Description du problème	82
6.3	Génération de chemins	90
6.4	Calcul d'une estimation inférieure	92
6.4.1	Choix de l'estimation inférieure	92
6.4.2	Méthode de perturbation des multiplicateurs	94
6.5	Calcul d'une estimation supérieure	95
6.6	Formulation sommets-arcs de PSG	95
6.7	Exemple de problèmes de sécurisation	98
6.8	Conclusion	101
7	Sécurisation par la méthode de décomposition proximale	102
7.1	Méthode de décomposition proximale en optimisation convexe	103
7.1.1	Motivation	103
7.1.2	Opérateur maximal monotone	104
7.1.3	L'algorithme du point proximal	105
7.1.4	L'algorithme de décomposition proximale	106
7.1.5	Optimisation convexe avec ADP	108
7.2	Première stratégie de décomposition (DP1)	109
7.2.1	Résolution du problème de sécurisation globale par la méthode de décomposition proximale	112
7.2.2	Étape proximale (a)	117
7.2.3	Étape proximale (b)	118
7.2.4	Réduction du nombre de copies	120
7.2.5	Conclusion	121
7.3	Deuxième stratégie de décomposition (DP2)	121
7.3.1	Formulation du problème avec copies	123
7.3.2	Étape proximale	125
7.3.3	Étape de projection	128
7.3.4	Avantages et inconvénients de la deuxième stratégie de décom- position	129
7.4	Génération de chemins et garanties d'optimalité	130
7.4.1	Calcul des multiplicateurs	130

7.4.2	Garanties d'optimalité	132
7.5	Validation de DP1 et DP2	134
7.5.1	Influence du nombre maximal d'itérations mineures	134
7.5.2	Influence du paramètre de la méthode de décomposition proximale	136
7.5.3	Influence du facteur de relaxation	138
7.5.4	Comparaison entre DP1 et DP2	139
7.5.5	Conclusion	140
8	Sécurisation par une méthode de points intérieurs	141
8.1	Introduction	141
8.2	Une approche par décomposition d'une méthode de points intérieurs .	142
8.2.1	Schéma de décomposition du système de déplacement de Newton	144
8.2.2	Système réduit	149
8.3	Étude de complexité	150
8.3.1	Complexité de la construction de la matrice M	151
8.3.2	Comparaison avec l'approche directe	152
8.4	Réduction de la taille du système réduit	154
8.5	Conclusion	158
9	Sécurisation par HOPDM	159
9.1	Motivation	159
9.2	Techniques de pré-résolution	160
9.3	Utilisation de HOPDM	161
9.3.1	Gestion de chemins	162
9.3.2	Techniques d'élimination de colonnes	163
9.4	Essais Numériques	164
9.4.1	Données générées	164
9.4.2	Données du CNET	165
9.5	Conclusion	170
10	Analyse numérique	171
10.1	Précision de la solution	171
10.2	Génération de chemins	173
10.3	Itérations	174
10.4	Temps de calcul	175

10.5 Conclusion	178
11 Annexe	179
11.1 Introduction	179
11.2 Structure de données	180
11.2.1 Codage des chemins	180
11.2.2 Classement des chemins par flot	182
11.2.3 Classement des chemins par panne	182
11.2.4 Multiplicateurs de Lagrange	183
11.3 Décomposition proximale	184
11.4 Méthode du gradient réduit	186
11.4.1 Principe de la méthode	186
11.4.2 Algorithme du gradient réduit	187
11.5 Initialisation du gradient réduit	188
12 Conclusion	190
Références	192

Liste des Tables

3.4.1 Mean value of the number of iterations.	42
3.4.2 Number of iterations in the worst case.	42
3.4.3 Mean value of the number of iterations.	42
3.4.4 Number of iterations in the worst case.	42
4.4.1 Performance of the algorithm NAF on Networks with 30 nodes and 60 arcs	67
4.4.2 Comparison between NAF and APF performances	67
4.4.3 Size of problems solved	68
4.4.4 Performance of the algorithm NAF	68
4.4.5 Performance of the algorithm APF	68
4.5.6 size of aggregated dates	69
8.3.1 Comparaison des deux approches	154
9.4.1 Taille du dernier programme maître	164
10.1.1 Précision	172
10.1.2 Précision de la solution	173
10.2.1 Nombre de chemins générés : mip	173
10.2.4 Taille du tableau des copies des chemins : mipp	174
10.2.5 Taille du tableau des copies des demandes : mibfp	174
10.3.1 Nombre d'itérations externes	175
10.3.2 Nombre total d'itérations internes	175
10.4.1 Temps de calcul en secondes	175
10.4.2 Comparaison des performances de DIP et DP2	177
11.2.1 $\text{fp}(\text{mcod}+1, \text{mip})$, path	181
11.2.2 $\text{costp1}(\text{mipt})$, coût nominal	181
11.2.3 $\text{costp2}(\text{mipt})$, coût de réserve	181
11.2.4 $\text{pp}(2, \text{mipp})$: path-path	181
11.2.5 $\text{fp}(4, \text{nk})$: flow-path	182

11.2.6	<code>bf(2,n)</code> : break-flow	182
11.2.7	<code>bf(5,mibfp)</code> : break-flow-path	183
11.2.8	<code>zlam(nk+1)</code> , pointeur sur les tableaux des multiplicateurs	183
11.2.9	<code>zlam(mibft)</code> , lie les pannes aux flots	183
11.2.10	<code>zlam2(mibft)</code> , lie les contraintes aux flots	184
11.2.11	<code>zlam(mibft)</code> : lie les multiplicateurs aux flots	184
11.3.12	<code>varx(2,mipt)</code> , projection du routage sur \mathcal{A}	185
11.3.13	<code>vary(1,mippt)</code> , projection du routage sur \mathcal{A}^-	185
11.3.14	<code>varz(nb+2,n)</code> : tableau des capacités de réserve	186

Liste des Figures

2.3.1 Exemple de routage	12
5.3.1 Exemple de réseaux avec la même connexité	74
5.4.2 Routage des demandes	76
5.4.3 Reroutage local	77
5.4.4 Reroutage global	77
6.7.1 Routage du flot 1	99
6.7.2 Reroutage du flot 1	99
6.7.3 Reroutage du flot 1	100
6.7.4 Reroutage du flot 1	101
7.3.1 Matrice des contraintes d'un problème à deux pannes	122
7.3.2 Matrice des contraintes du problème avec copies	122
7.3.3 Structure de la hessienne	127
7.5.4 $p = 6, n = 15 K = 15$ et $maxiter = 100$	135
7.5.5 $p = 6, n = 15 K = 15$ et $maxiter = 10$	135
7.5.6 $p = 6, n = 15 K = 15, maxiter \in \{10, 100\}$	136
7.5.7 $p = 30, n = 30 K = 30$ et $\Lambda \in \{1, 20\}$	137
7.5.8 $p = 6, n = 15 K = 15$ et $\Lambda \in \{10^{-3}, \dots, 10^3\}$	138
7.5.9 $p = 6, n = 15 K = 15, maxiter = 100, fac = 10$ et $\rho \in \{0.75, 1, 1.5\}$	138
7.5.10 $p = 4, n = 6 K = 6, \lambda = 1$ et $\rho = 1$	139
7.5.11 $p = 5, n = 10 K = 10, \lambda = 1$ et $\rho = 1$	140
7.5.12 $p = 6, n = 15 K = 15, \lambda = 1$ et $\rho = 1$	140
9.4.1 La précision obtenue avec DIP sans élimination de chemins	165
9.4.2 Variation de la valeur optimale du programme maître	166
9.4.3 Variation du nombre de colonnes de la matrice des contraintes	166
9.4.4 Variation du nombre de lignes de la matrice des contraintes	167
9.4.5 Variation de la taille du problème maître	167
9.4.6 Variation du nombre d'itérations internes	168

9.4.7	Comparaison de la variation du coût	168
9.4.8	Comparaison de la variation du nombre de chemins	169
9.4.9	Comparaison de la variation du nombre de contraintes	169
9.4.10	Comparaison de la variation du nombre d'itérations internes	170
10.0.	Organigramme	171
10.1.	Variation de la précision en fonction de $\log(\theta)$	172
10.4.3	Variation de la vitesse de convergence de DP2 en fonction de la taille du problème	176
10.4.4	Variation de la vitesse de convergence de DIP en fonction de la taille du problème	176
10.4.5	Variation de la vitesse de convergence de DIP avec élimination de chemins	177

1. Introduction

1.1 Objectifs généraux

Cette étude a pour objet l'application des algorithmes rapides basés sur la notion de points intérieurs au problème dit de routage-reroutage de communications.

Les algorithmes de points intérieurs ont été la principale innovation dans le domaine des algorithmes d'optimisation dans ces dernières années. La rapidité et la robustesse de ces méthodes leur ont même permis de supplanter l'algorithme du simplexe, qui était resté le principal outil de résolution de problèmes linéaires depuis environ 50 ans. Dans certains domaines spécifiques, dont le plus important est celui des problèmes de flot, les algorithmes spécialisés (par exemple la version du simplexe adaptée au flot) sont néanmoins capables de mettre en échec les outils généraux.

Le problème de routage-reroutage constitue un cas intermédiaire puisqu'il se modélise comme un problème de multiflots. Malgré son analogie avec les problèmes de flot, la structure multiflots est loin d'avoir des propriétés aussi porteuses sur le plan numérique : les solutions ne sont pas forcément entières (en présence de données entières).

La complexité du problème pratique est néanmoins due en grande partie à la question du reroutage. Il s'agit de concevoir comment l'acheminement des messages sera modifié en cas de panne. Ce problème rentre dans le cadre général de l'optimisation (linéaire) avec recours, structure qui appelle l'emploi d'algorithmes de décomposition.

1.2 Cadre général

Les réseaux de télécommunications sont des systèmes de transmissions complexes destinés à transmettre des informations entre des paires de nœuds. Ils sont sujets à des pannes.

Quand une panne se produit, l'opérateur chargé de gérer le réseau en temps réel doit rediriger toutes les demandes portées par l'élément défaillant aussi rapidement que possible. Même si le reste du réseau a un excès de capacité, le reroutage peut être

impossible à réaliser.

Dans le modèle étudié dans cette thèse, pour assurer l'existence d'une capacité suffisante pour le reroutage, nous considérons un second réseau appelé le *réseau de réserve*. Ce réseau est installé en parallèle avec le réseau nominal, est de même type que ce dernier et possède une topologie identique.

Notre problème de sécurisation consiste à trouver le routage et l'investissement de moindre coût en capacités nominal et de réserve pour un certain jeu de demandes.

Dans notre étude nous écartons la possibilité d'avoir simultanément plusieurs pannes. Nous nous limitons au cas des pannes simples d'arcs pour des raisons de complexité numérique.

Il existe deux types de reroutage : local et global. La stratégie de reroutage local est de considérer que la rupture d'un arc crée entre ses extrémités une demande égale à la somme des flots qui le traversaient. Cette demande doit être reroutée dans le réseau de réserve. Dans ce cas, le reroutage se ramène à un problème de flot simple. Dans le cas du reroutage global, la demande créée par la panne est analysée pour trouver les différentes paires (origine-destination) affectées. La demande générée par la panne est considérée, alors, comme un multiflot. Rerouter cette demande revient à résoudre un problème de multiflot dans le réseau de réserve.

L'interruption de ces flots libère de la capacité dans le réseau nominal. Cette capacité libérée peut être utilisée pour le reroutage. Dans notre modèle nous utilisons uniquement la capacité de réserve pour assurer le routage d'un certain jeu de demandes.

L'approche globale est plus difficile à réaliser que la locale mais elle est économiquement préférable.

Le problème de sécurisation avec reroutage local a été déjà étudié par Lissier, Sarkissian et Vial [87]. Il comporte deux volets liés : celui du dimensionnement du réseau de réserve et celui du routage du trafic. Lissier et al formulent le problème comme un programme linéaire de très grande taille qu'ils résolvent à l'aide d'un algorithme de plans sécants basé sur le concept de centre analytique (ACCPM). La méthode permet ainsi de sécuriser avec l'approche locale des réseaux d'une taille allant jusqu'à 60 nœuds et 120 arcs, ce qui correspond à une modélisation réaliste du réseau de transmission d'interconnexion de France-Télécom.

1.2.1 Problèmes linéaires de multiflot

Un grand nombre de problèmes dans différents domaines (comme le transport, les télécommunications, la finance, etc ...) peuvent être modélisés en terme de flots. Dans ces problèmes on cherche à trouver le moyen le plus optimal pour transporter un flot (d'information, de bus, de courant électrique) dans un réseau (de transmissions, routier, électrique, etc ...).

Parmi ces problèmes d'optimisation, plusieurs représentent une classe spéciale de problèmes linéaires.

Dans un problème de flot minimal, l'objectif est de déterminer un flot qui transporte la demande de son nœud origine à son nœud destination de la manière la plus efficace tout en respectant la loi de Kirchhoff (la loi de conservation de flot [46]) et les limitations sur les arcs (les contraintes de capacité).

La complexité d'un algorithme de résolution de problème de flot dépend fortement de la représentation du réseau (représentation avec matrice d'adjacence ou matrice d'incidence ou liste d'adjacence, etc ...) et des structures utilisées pour maintenir et mettre à jour les calculs intermédiaires [120]. Au cours de notre étude nous constaterons l'impact de la formulation (sommets-arcs ou arcs-chemins) sur la vitesse de convergence de l'algorithme.

Pour trouver un flot réalisable (vérifiant la loi de Kirchhoff et les limites sur la capacité sur les arcs) on pourrait résoudre un problème de flot maximal [5].

Dans beaucoup de cas réels, plusieurs flots partageant un même réseau doivent être optimisés. La formulation de ces derniers s'exprime en terme de multiflot. Un cas particulier des problèmes de multiflot est celui de flot de coût minimal.

L'intérêt aux problèmes de multiflot dans les réseaux remonte aux années 40, où Hitchcock [67] a proposé un algorithme pour résoudre un problème de transport. Dantzig [31] a ensuite développé la méthode de simplexe dans le cadre de la programmation linéaire.

Pendant les années 50, Kruskal s'est intéressé au problème d'arbre de poids minimum et a proposé un algorithme pour sa résolution [83]. Trouver un plus court chemin est l'un des plus importants problèmes de flot simple. En 1957, Prim a proposé un algorithme pour le résoudre [118].

Le premier livre sur les problèmes de flots était publié par Ford et Fulkerson [37].

Depuis, avec la commercialisation intense des grands ordinateurs, des recherches actives ont produit une variété d'algorithmes et de codes pour résoudre ces problèmes. Pour une introduction de problèmes de multiflots et de leurs applications nous renvoyons le lecteur aux livres [5, 113, 37, 77].

Un résumé de l'histoire des différentes approches proposées à la résolution du problème de multiflot linéaire jusqu'à l'année 1977 se trouve dans [18]. Les études numériques menées pendant les deux dernières décennies (voir par exemple [40, 55, 77, 108]) élisent les spécialisations de l'algorithme de simplexe comme les algorithmes les plus rapides pour la résolution de ces problèmes.

Pourtant en 1970 Klee et Minty ont montré dans [80] que l'algorithme du simplexe de Dantzig [32] n'est pas polynômial.

1.2.2 Méthodes de points intérieurs

La possibilité de résoudre un problème linéaire en temps polynômial a été prouvé par Khachiyan en 1979 [79]. Cet algorithme a donné naissance à une famille de méthodes connues sous le nom de méthodes de points intérieurs (MIP). Ces dernières ont été utilisées pour résoudre efficacement des problèmes de multiflots. En effet, la meilleure complexité connue pour la résolution de problèmes de multiflots a été fournie par une de ces méthodes [74].

Cependant les premières implémentations n'étaient pas avantageuses pour les méthodes de points intérieurs. L'une des premières implémentations est décrite dans [4]. La tâche principale dans un algorithme de points intérieurs est la résolution des systèmes linéaires nécessaires au calcul des directions de déplacement. En 1988, Karmarkar et Ramakrishnan [75] introduisent une méthode de points intérieurs (*approximate dual projective*) qui utilise un algorithme de gradient conjugué préconditionné pour résoudre ces systèmes linéaires. Plusieurs auteurs ont proposé différents préconditionnements pour résoudre des problèmes de multiflots [114, 75, 116, 73, 95]. Pour une étude plus complète des méthodes de points intérieurs pour la résolution de problèmes de réseau nous nous référons à [120]. Toutes ces approches ont tenté d'exploiter la structure du problème linéaire de multiflot. Cependant les préconditionnements qu'elles utilisent restent indépendants de la structure du problème.

Récemment, Castro [20] a proposé un préconditionnement qui exploite la structure

du problème de multiflot linéaire de coût minimum (PMLCM).

Au cours de cette thèse nous nous sommes également intéressés à ce problème. Nous avons trouvé des similarités entre notre algorithme de points intérieurs et celui proposé par Castro [20] en terme de schéma de décomposition. L'intérêt de notre algorithme par rapport à celui de Castro réside dans l'utilisation de la formulation arcs-chemins. En effet la matrice de préconditionnement proposée par Castro [20] repose sur la formulation sommets-arcs et ne peut pas être étendue à la formulation arcs-chemins. Nous verrons dans la suite que cette dernière est numériquement plus performante que la formulation sommets-arcs pour la résolution des problèmes traités dans cette thèse.

Dans notre étude nous nous limitons aux problèmes linéaires de multiflot. Pour les problèmes non-linéaires de multiflot nous renvoyons le lecteur aux références suivantes [99, 82, 112, 62, 61, 41].

Nous nous sommes intéressés plus particulièrement à une méthode appartenant à la famille des méthodes de points intérieurs primales duales. Elle a été introduite par Mizuno, Todd and Ye dans [102] sous le nom de méthode *prédicteur-correcteur*. Les méthodes de points intérieurs primaux duaux (appelées encore méthodes de suivi de trajectoire centrale) possèdent à la fois la meilleure estimation de complexité connue à ce jour ($O(\sqrt{n}L)$ itérations) et une convergence quadratique [13, 52].

1.2.3 Méthodes proximales

La difficulté d'un problème de multiflot provient non seulement de sa grande taille mais surtout de la gestion des conflits sur les arcs, occasionnés par la limitation des capacités.

Le problème de sécurisation globale admet en plus de ces limitations, un autre niveau de couplage entre les pannes du réseau. Ce qui rend ce problème particulier de multiflot encore plus compliqué.

Dans [110], Ouorou s'intéresse au cas non linéaire convexe des problèmes de multiflot et propose une approche de décomposition utilisant les techniques proximales. Il applique plus précisément la méthode de décomposition proximale introduite dans [91] par Mahey, Oualibouch et Tao. L'algorithme de décomposition proximale appliqué au problème de multiflot strictement convexe de [111], donne des résultats satisfaisants

en comparant avec ACCPM [42].

L'idée générale de cette approche est de représenter le couplage entre des sous-systèmes par des sous-espaces de produit d'espaces des copies des variables primales et duales. La structure du problème transformé serait adéquate à l'application des techniques proximales. L'algorithme effectue à chaque itération deux pas distincts : un pas proximal qui régularise la fonction objective en ajoutant un terme quadratique dépendant des solutions primale et duale de l'itéré précédent et un pas de projection sur les sous-espaces correspondants.

L'efficacité de l'algorithme de décomposition proximale dépend du choix de ces sous-espaces.

Dans notre étude nous proposons deux stratégies de décomposition du problème de sécurisation globale. La projection sur les sous-espaces correspondant à ces stratégies revient à un simple calcul de moyenne. La tâche principale des deux algorithmes associés à ces stratégies est l'optimisation de sous-problèmes décomposés de multiflots quadratiques avec des contraintes linéaires.

1.3 Contenu de la thèse

Ce mémoire est organisé en 10 chapitres, dont le premier est cette introduction.

Le deuxième chapitre introduit des notions de la théorie des graphes que nous utilisons dans la suite.

Une étude de la robustesse des algorithmes prédicteurs-correcteurs est présentée dans le troisième chapitre. Nous nous intéressons au cas où la direction de Newton est calculée de manière approchée.

Le quatrième chapitre est consacré à l'étude des problèmes linéaires de multiflot. Nous proposons la résolution de problèmes de multiflot de coût minimum par une approche de points intérieurs par décomposition. Les algorithmes utilisant cette approche, sont testés sur des données réelles fournies par le CNET.

La deuxième partie de cette thèse concerne la conception de réseaux de communications. Nous commençons cette partie par une description d'une série de modèles mathématiques de conception de réseaux de télécommunications qui ont été développés récemment (Chapitre 5).

Le problème de sécurisation des réseaux de transmission introduit au chapitre 5, est étudié dans le sixième chapitre. Nous discutons ses différentes formulations possibles.

Nous explicitons ensuite, une méthode de génération de chemins qui est intégrée à la formulation arcs-chemins du problème. Nous proposons également des garanties d'optimalité.

Les chapitres suivants présentent des méthodes de résolution du problème de sécurisation globale.

Dans le chapitre 7, nous proposons deux stratégies de décomposition basées sur la méthode de décomposition proximale et la génération de chemins.

Une troisième méthode de décomposition basée sur les points intérieurs est présentée dans le chapitre 8. Elle est inspirée des travaux que nous avons effectués dans le cadre du routage [15].

L'étude de complexité de l'algorithme précédent nous a amené à la résolution directe du programme maître par une méthode de points intérieurs. Ceci fait l'objet du chapitre 9, où nous décrivons l'utilisation du code HOPDM (Higher Order Primal-Dual Method) pour la résolution du problème de sécurisation globale. La croissance du nombre de colonnes peut représenter un handicap à cette méthode. Pour une meilleure utilisation de ce code nous avons étudié des stratégies de suppression de chemins.

A l'exception de la méthode de points intérieurs présentée au chapitre 8, toutes les autres méthodes ont été implémentées et étudiées numériquement. Le chapitre 10 est consacré à l'analyse numérique de ces méthodes.

Une annexe concernant les détails des implémentations des algorithmes étudiés est également fournie.

2. Quelques notions de la théorie des graphes

Dans ce chapitre nous introduisons quelques notions sommaires de la théorie des graphes qui vont nous être utiles dans les chapitres suivants. Pour plus de détails nous nous référons à [46] de Gondran et Minoux, ainsi qu'à l'ouvrage [5], plus récent, de Ahuja, Magnanti et Orlin.

2.1 Généralités sur les graphes

La théorie des graphes permet de représenter simplement la structure d'un grand nombre de situations. L'exemple le plus classique est la représentation d'un *réseau de communication* : réseau de routes représenté par une carte routière, réseau de chemin de fer, de téléphone, ou de relais de télévision, réseaux électriques, réseaux d'informations dans une organisation, etc ... [46].

Définition 1. Un graphe orienté $G(V, E)$ est déterminé par un ensemble V non vide dont les éléments sont appelés *sommets ou nœuds* et un ensemble E dont les éléments sont des couples ordonnés de sommets appelés *arcs*. Les sommets seront numérotés $1, \dots, p$ (on dit alors que G est d'ordre p), et les arcs $1, \dots, n$.

Définition 2. Pour $A \subset V$, nous désignons par $W^+(A)$ l'ensemble des arcs ayant leur extrémité initiale dans A et leur extrémité terminale dans $(V - A)$.

Nous avons $W^-(A) = W^+(V - A)$.

Définition 3. Le *degré* d'un sommet $i \in V$ est l'entier $d(i) = |W^+(i)| + |W^-(i)|$.

Définition 4. La *matrice d'incidence* sommets-arcs d'un graphe G est la matrice $A = (a_{ij})$ de taille $p \times n$, définie par

$$a_{ij} = \begin{cases} 1 & \text{si } i \text{ est le sommet initial de } j, \\ -1 & \text{si } i \text{ est le sommet terminal de } j, \\ 0 & \text{dans les autres cas.} \end{cases} \quad (2.1.1)$$

Définition 5. Une *arête* est un couple non ordonné de nœuds. Tandis que pour un *arc* l'ordre du couple de nœuds qui le définit est important $[(i, j) \neq (j, i)]$.

Définition 6. Une *chaîne* dans G est une suite d'arcs j_1, \dots, j_p telle que chaque arc j_l ($2 \leq l \leq p - 1$) a une extrémité commune avec l'arc j_{l+1} .

Une chaîne est *simple* si elle ne comporte pas plusieurs fois le même arc dans la suite qui la construit. On dit qu'une chaîne est *élémentaire* si tous les sommets sont au plus de degré 2.

Définition 7. Un *cycle* est une chaîne dont les extrémités coïncident.

Un cycle est dit *élémentaire* s'il ne contient strictement aucun autre cycle.

Définition 8. Un *chemin* est une chaîne dont tous les arcs sont orientés dans le même sens. Un chemin *élémentaire* est un chemin dont tous les sommets sont au plus de degré 2.

Définition 9. Un *circuit* est un chemin dont les extrémités coïncident. Un circuit *élémentaire* est un circuit ne contenant strictement aucun autre circuit.

Définition 10. Un graphe $G(V, E)$ est dit *complet* si, pour toute paire de nœuds (i, j) , il existe au moins un arc de la forme (i, j) ou (j, i) .

2.2 Connexité

Une notion importante pour la sécurisation des réseaux est la connexité. Ce paragraphe introduit quelques définitions liées à la connexité.

Définition 11. Un graphe est dit *connexe* si, pour tout couple de sommets i et j , il existe une chaîne joignant i et j .

Définition 12. Un graphe est dit *fortement connexe* si, étant donnés deux sommets quelconques i et j (dans cet ordre), il existe un chemin d'extrémité initiale i et d'extrémité terminale j .

2.2.1 Point d'articulation et Isthme

Définition 13. Un *point d'articulation* d'un graphe est un sommet dont la suppression augmente le nombre de composantes connexes.

Définition 14. Un *isthme* est une arête dont la suppression augmente le nombre de composantes connexes.

Définition 15. Un ensemble d'*articulations* $\mathcal{A} \in V$ d'un graphe G est un ensemble \mathcal{A} de sommets tel que le sous-graphe $G_{V-\mathcal{A}}$, déduit de G par suppression des sommets de \mathcal{A} , ne soit plus connexe.

2.2.2 La k -connexité

Définition 16. Un graphe est dit *k -connexe* si et seulement si il est connexe d'ordre $p \geq k + 1$, et il n'admet pas d'ensemble d'articulations de cardinal $k - 1$.

2.2.3 La k -arête-connexité

Définition 17. Un graphe est dit *k -arête-connexe* si et seulement s'il est connexe avec n arêtes $n \geq k + 1$, et n'admet pas d'ensemble d'isthmes de cardinal $k - 1$.

Autrement dit, G est k -arête-connexe s'il ne peut être déconnecté par l'élimination de moins de k arêtes [46].

Définition 18. Un graphe fortement connexe est *k -arc-connexe* s'il reste fortement connexe après élimination de moins de k arcs.

On remarque que 1-arc-connex est équivalent à la forte connexité.

“Un des meilleurs exemples pour illustrer la notion de flot sur un graphe est certainement celui du courant électrique, et c'est d'ailleurs dans ce cadre qu'ont été d'abord étudiés les problèmes de flots” [46].

De nos jours, la notion de flot offre un modèle général qui couvre un large champs d'applications importantes. Elle s'applique à des problèmes concrets comme les problèmes de transport (routier, aérien, ferroviaire), la structuration et dimensionnement optimaux des réseaux de télécommunications, les problèmes de gestion des stocks, d'ordonnancement et d'affectation sans oublier sa grande importance dans les mathématiques combinatoires.

2.2.4 Définition d'un flot

Soit $G(V, E)$ un graphe connexe dont les arcs sont numérotés $e = 1, 2, \dots, n$.

Définition 19. Un flot est un vecteur à n composantes $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, tel qu'en tout sommet $i \in V$ de G , la première loi de Kirchhoff soit vérifiée, c'est à dire

$$\sum_{u \in W^+(i)} x_u = \sum_{u \in W^-(i)} x_u. \quad (2.2.1)$$

Pour $e \in E$, la composante e du vecteur x est appelée *quantité de flot* ou *flux* sur l'arc e . La relation (2.2.1) exprime simplement que la somme des flux entrant en un sommet, est égale à la somme des flux sortant (loi de conservation aux nœuds).

2.2.5 Définition algébrique des flots

Soit $A = (a_e^i)$, $i = 1, \dots, p$, $e = 1, \dots, n$, la matrice d'incidence sommets-arcs du graphe G . A chaque sommet i de V correspond la ligne i de la matrice A , et l'on a :

$$W^+(i) = \{e \in E / a_e^i = +1\} \text{ et } W^-(i) = \{e \in E / a_e^i = -1\}.$$

La loi de conservation aux nœuds (2.2.1) peut donc se mettre sous la forme matricielle équivalente: $Ax = \mathbf{0}$.

2.3 Multiflots

Les problèmes de *multiflots* apparaissent lorsque plusieurs produits partagent un même réseau. La planification, le management, les transports, les télécommunications sont quelques-uns des domaines où un grand nombre de situations se modèlisent en problèmes de multiflots.

L'une des raisons de l'importance des multiflots est le fait qu'ils constituent un champ de tests naturel de nouvelles méthodes de décomposition.

2.3.1 Définition d'un multiflot

Soit un graphe $G(V, E)$ orienté ($|V| = p$ et $|E| = n$) sur lequel circulent K flots simples x^1, \dots, x^K . Pour $k = 1, \dots, K$, x^k est un flot simple entre le nœud source o^k

et le nœud puits p^k de valeur r^k s'il vérifie :

$$Ax^k = b^k,$$

où A est la matrice d'incidence sommets-arcs du graphe et $b^k \in \mathbb{R}^p$ est un vecteur à composantes toutes nulles sauf $b^k_{o^k}$ et $b^k_{p^k}$, qui valent respectivement $+r^k$ et $-r^k$.
Considérons le vecteur $x^{mf} = (x^{mf})_{e \in E}$ défini par

$$x^{mf} = \sum_{k=1}^K x^k.$$

Le vecteur x^{mf} est un vecteur *multiflot* sur G de valeurs r^1, r^2, \dots, r^K .

La *valeur totale* de x^{mf} est $r = r^1 + r^2 + \dots + r^K$.

2.3.2 Exemples de problèmes de multiflots

Considérons un réseau de communication constitué par un ensemble de centres V , et un ensemble de voies de communication E .

Ce réseau a pour fonction d'écouler, entre certains couples de centres (i, j) une certaine quantité de trafic t_{ij} . Comme le trafic entre i et j ne peut pas se mélanger à du trafic entre k et l , les problèmes d'écoulement de trafic dans de tels réseaux se formulent comme des problèmes de multiflots.

- Si on s'intéresse à la possibilité d'écoulement d'une certaine demande $T = (t_{ij})$ à travers un réseau donné, dont chaque arc (arête) $e \in E$ est muni d'une capacité connue $U_e \geq 0$, il s'agit d'un problème de *multiflot compatible*.
- Si on munit chaque arc $e \in E$ d'un coût $\Phi_e(U_e)$ d'installation de U_e unité de capacité, et que l'on cherche à écouler une certaine demande $T = (t_{ij})$ au moindre coût, on a à résoudre un problème de *multiflot de coût minimum* (pour plus de détails voir [46]).

2.3.3 Définition d'un routage

On dit que l'on a effectué un *routage* des flots x^1, \dots, x^K lorsqu'on a affecté à chaque flot k des chemins particuliers pris parmi l'ensemble des chemins entre o^k et p^k .

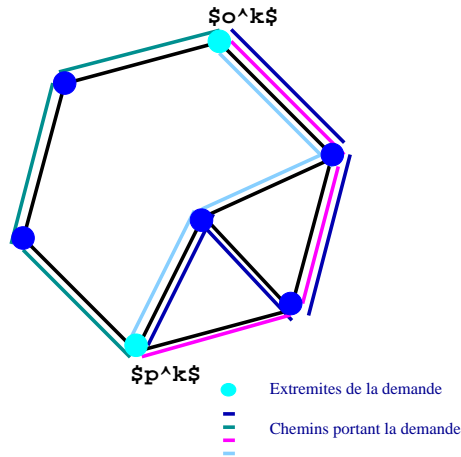


Figure 2.3.1: Exemple de routage

Remarque 2.3.1. On peut définir un multiflot x^{mf} sans imposer aux flots d'être positifs.

Dans ce cas la composante x_j^{mf} du multiflot est la somme des valeurs absolues des composantes x_j^k des différents flots.

Sur un graphe non orienté N , en remplaçant chaque arête $j = [i, t]$ par deux arcs $j^+ = (i, t)$ et $j^- = (t, i)$, on définit un graphe orienté G . Soient alors K flots x^1, \dots, x^K à composantes positives sur G . Le vecteur $x^{mf} = \sum_{k=1}^K x^k$ défini par

$$x^{mf} = \sum_{k=1}^K x^{k+} + \sum_{k=1}^K x^{k-}$$

est appelé multiflot sur le graphe non orienté N .

2.4 Méthode de génération de colonnes

Dans ce paragraphe nous présentons le principe général de la génération de colonnes [85] (XII.4.2).

Nous nous intéressons à la résolution d'un problème linéaire

$$\begin{cases} \text{Min}_x c^T x \\ \text{(i) } Ax = b, \\ \text{(ii) } x \in X, \end{cases} \quad (2.4.1)$$

où $X \subset \mathbb{R}^n$ est un ensemble polyédral, $A \in \mathbb{R}^{p \times n}$, $c, x \in \mathbb{R}^n$ et $b \in \mathbb{R}^p$. La contrainte (2.4.1.i) rend le problème (2.4.1) plus difficile. Nous supposons que sans

la contrainte (2.4.1.i), le problème pourrait exploiter pleinement la structure spéciale de l'ensemble X et être facilement résolu.

Tout point de l'ensemble polyédral peut être représenté par une combinaison linéaire de ses points et rayons extrémaux. Soient p_1, \dots, p_k et r_1, \dots, r_l respectivement les points et les rayons extrémaux de X . Pour tout $x \in X$ il existe des coefficients barycentriques $\alpha_1, \dots, \alpha_k$ et β_1, \dots, β_l tels que :

$$\left\{ \begin{array}{l} x = \sum_{i=1}^k \alpha_i p_i + \sum_{j=1}^l \beta_j r_j, \\ \alpha_i \geq 0, \quad i = 1, \dots, k, \\ \beta_j \geq 0, \quad j = 1, \dots, l, \\ \sum_{i=1}^k \alpha_i = 1. \end{array} \right. \quad (2.4.2)$$

En utilisant (2.4.2) nous formulons le problème (2.4.1) comme suit :

$$\left\{ \begin{array}{l} \text{Min}_{\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_l} \sum_{i=1}^k \alpha_i c^T p_i + \sum_{j=1}^l \beta_j c^T r_j \\ \text{(i)} \quad \sum_{i=1}^k \alpha_i A p_i + \sum_{j=1}^l \beta_j A r_j = b, \\ \text{(ii)} \quad \sum_{i=1}^k \alpha_i = 1, \\ \text{(iii)} \quad \alpha_i \geq 0, \quad i = 1, \dots, k, \\ \text{(iv)} \quad \beta_j \geq 0, \quad j = 1, \dots, l. \end{array} \right. \quad (2.4.3)$$

Le problème (2.4.3) a un très grand nombre de variables. Il est souvent impossible de le formuler explicitement (2.4.3) et de le résoudre directement. Par contre nous pouvons générer les points p_i et les rayons r_i uniquement s'ils sont nécessaires. Nous considérons respectivement des sous ensembles de points extrémaux et de rayons extrémaux de X : $\{p_i, i \in I\}$ et $\{r_j, j \in J\}$. En restreignant le problème (2.4.3) à ces sous ensembles, nous obtenons un problème maître réduit. La solution primale de ce dernier fournit une estimation supérieure de la valeur optimale du problème (2.4.3). Pour avoir une estimation inférieure de cette valeur, nous utilisons la solution duale du problème maître réduit. Plus précisément, nous utiliserons le multiplicateur associé à la contrainte (2.4.1.i), que nous notons y . En dualisant la contrainte (2.4.1.i), le Lagrangien associé au problème (2.4.1) peut s'écrire comme suit $\mathcal{L}(x, y) = c^T x + y^T (Ax - b)$, $x \in X$. Une estimation inférieure de la valeur optimale du problème (2.4.1)

est donnée par $\mathcal{L}(y) = \inf_{x \in X} \mathcal{L}(x, y)$. La technique de génération de colonnes fournit des garanties de la qualité de la solution. L'algorithme est arrêté lorsque la différence entre les estimations supérieure et inférieure est plus petite que la précision désirée. Si la solution courante ne vérifie pas le critère d'arrêt précédent, nous pouvons déduire que l'ensemble X n'est pas assez caractérisé et qu'une nouvelle colonne Ap_i ou Ar_i devrait être introduite au programme maître réduit. Suivant la valeur de $\mathcal{L}(y)$, l'algorithme introduit une nouvelle colonne Ap_i ou alors Ar_i au programme maître réduit. Si $\mathcal{L}(y)$ est fini alors il existe un point extrémal p de X tel que $\mathcal{L}(y) = c^T p + y^T (Ap - b)$. Et si $\mathcal{L}(y)$ est infini, il existe alors un rayon r de X , tel que $c^T r + y^T (Ar - b) < 0$. Dans le schéma classique de la génération de colonnes [76], tout programme maître réduit est résolu jusqu'à l'optimalité. Gondzio et Sarkissian proposent dans [51] une méthode de génération de colonnes qui s'inspire de la méthode ACCPM (analytic center cutting plane method) [42] et utilise les prix centraux. Ils utilisent pour la résolution du programme maître réduit, un algorithme de points intérieurs primal dual et non réalisable qui emploie la notion de μ -centre afin de contrôler la distance à l'optimalité.

3. Algorithmes de suivi de chemin perturbé

Les méthodes de points intérieurs ont apporté une véritable révolution [52, 81]. En particulier la famille des algorithmes de points intérieurs primaux duaux qui donne les meilleurs résultats. Ces algorithmes possèdent à la fois une complexité polynômiale et une convergence quadratique. Malgré les progrès effectués dans l'analyse théorique des algorithmes, il y a un écart important entre les vitesses théoriques de convergence, assez lentes, et les performances pratiques. Nous nous intéressons plus spécifiquement aux algorithmes prédicteur-correcteur. Ce sont des algorithmes de suivi de trajectoire centrale qui opèrent à l'intérieur d'un voisinage \mathcal{R} .

Ils consistent à alterner:

- les pas de centralisation, de façon à se rapprocher de la trajectoire centrale.
- les pas affines dans lesquels l'algorithme calcule le plus grand pas permettant de rester dans \mathcal{R} .

Ces algorithmes de suivi de chemin opèrent en prenant comme direction de déplacement la direction de Newton associée à l'équation de la trajectoire centrale, tout en contrôlant le centrage grâce à une mesure de proximité [52]. Ils utilisent un paramètre μ , qui mesure le respect des conditions de complémentarité. On évalue la complexité de l'algorithme par le nombre d'opérations permettant de calculer un point associé à la mesure visée μ_∞ . On pose $\bar{L} = \log(\mu_0/\mu_\infty)$ où μ_0 est la mesure du point initial.

Les algorithmes de suivi de chemin de type prédicteur-correcteur se sont avérés très efficaces pour résoudre des problèmes d'optimisation linéaire. Cependant, l'hypothèse du calcul exact de la direction de *Newton* (correspondant à un pas affine ou de centrage) est peu réaliste. En effet, dans le cas de problèmes de grande taille, il peut être nécessaire d'utiliser des algorithmes itératifs. L'article [16] présente une étude de ces algorithmes prenant en compte une erreur dans le membre de droite. On donne des estimations précises et explicites de l'erreur permettant de préserver la complexité algorithmique du cas non perturbé. On calcule aussi des estimations explicites

permettant de garantir un taux de convergence linéaire donné. Les résultats sont obtenus dans le cadre de problèmes de complémentarités monotones. Ils s'appliquent donc aussi à l'optimisation quadratique convexe.

Rapport INRIA RR-2745

Accepted in “*Optimization Methods and Software*”

J.F. Bonnans and C. Pola and R. Rébaï

**PERTURBED PATH FOLLOWING
PREDICTOR-CORRECTOR
INTERIOR POINT ALGORITHMS Abstract**

The path following algorithms of predictor corrector type have proved to be very effective for solving linear optimization problems. However, the assumption that the Newton direction (corresponding to a centering or affine step) is computed exactly is unrealistic. Indeed, for large scale problems, one may need to use iterative algorithms for computing the Newton step.

In this paper, we study algorithms in which the computed direction is the solution of the usual linear system with an error in the right-hand-side. We give precise and explicit estimates of the error under which the computational complexity is the same as for the standard case. We also give explicit estimates that guarantee an asymptotic linear convergence at an arbitrary rate. Finally, we present some encouraging numerical results.

Because our results are in the framework of monotone linear complementarity problems, our results apply to convex quadratic optimization as well.

3.1 Introduction

In the last decade, a new generation of polynomial algorithms, based on the idea of computing a sequence of interior points, brought a revolution in the field of linear optimization [52, 81]. In the past few years, some algorithms that follow the central path, using Newton directions on the equation of the central path, focused the attention of the community because they both reach the optimal complexity known until now, while converging quadratically [102, 104, 117]. Most available implementations of interior point methods are actually of this type.

For very large scale problems it may be useful to compute the Newton step by an iterative algorithm. It is often observed that iterative algorithms compute at a small cost a rough approximation of the solution, while it may be much more expensive to obtain a precise value of the solution. Therefore a question arises: will the good convergence properties of path following algorithms remain if the Newton direction is computed approximately? This question is meaningful in the general framework of linear complementarity problems, in which linear optimization problems can be embedded (this also allows us to embed quadratic programming). Our concern is when the linearization of the complementarity condition is solved approximately, that is to say we consider the usual linear equations corresponding to the complementarity condition with a perturbation in the right-hand-side; we assume that the equations of the Newton step corresponding to the linear equations of the complementarity problem (i.e., in the case of linear optimization, the primal and dual linear feasibility constraints) themselves are not perturbed, but they can have a non-null right-hand-side if the starting point is infeasible.

In that framework, we are able to give explicit estimates on the precision with which the Newton step is computed, in order to keep complexity at the same order as for the non-perturbed case, or an asymptotic linear convergence rate at an arbitrary rate. We give such estimates for two algorithms of the predictor corrector type, in small and large neighborhoods, respectively.

An early reference where inexact computation of the Newton step is discussed is [82]. Let us mention the reference [105], where inexact Newton steps are also discussed in the framework of linear programming. The perturbation in that paper is in the right-hand-side of the linearization of the equations of the Newton step corresponding to the primal and dual feasibility conditions, and the obtained bounds are not so explicit as those obtained here. However, we note the following relation between the two kind of perturbations: if the perturbation is in the right-hand-side of the feasibility conditions by using projections it can be reduced to our framework.

Another discussion of inexact Newton steps, in the context of interior point algorithms for network flow problems, can be found in [115], where it is shown that this kind of method may be effective.

The paper is structured as follows. Section 3.2 presents the main results for the perturbed predictor corrector algorithm in small and large neighborhoods. The proofs are given in Section 3.3. Finally some numerical results are reported in the

last section.

Conventions Given a vector $x \in \mathbb{R}^n$ the relation $x > 0$ is equivalent to $x_i > 0$, $i = 1, 2, \dots, n$, while $x \geq 0$ means $x_i \geq 0$, $i = 1, 2, \dots, n$. We denote $\mathbb{R}_+^n = \{x \in \mathbb{R}^n : x \geq 0\}$ and $\mathbb{R}_{++}^n = \{x \in \mathbb{R}^n : x > 0\}$. We write $\|\cdot\|$ instead of $\|\cdot\|_2$. Whenever we use other norms like $\|\cdot\|_\infty$ we use the corresponding symbol.

Given a vector x , the corresponding upper case symbol denotes as usual the diagonal matrix X defined by the vector. The symbol $\mathbf{1}$ represents the vector of all ones, with dimension given by the context.

We denote component-wise operations on vectors by the usual notations for real numbers. Thus, given two vectors u, v of the same dimension, uv , u/v , etc. will denote the vectors with components $u_i v_i$, u_i/v_i , etc. We denote the null space and range space of a matrix M by $\mathcal{N}(M)$ and $\mathcal{R}(M)$ respectively.

The notation $x^k = O(\mu_k)$ means that there is a constant K (dependent on problem data) such that for every $k \in \mathbb{N}$, $\|x^k\| \leq K\mu_k$. Similarly, if $x^k > 0$, $x^k = \Omega(\mu_k)$ means that $(x^k)^{-1} = O(1/\mu_k)$. Finally, $x^k \approx \mu_k$ means that $x^k = O(\mu_k)$ and $x^k = \Omega(\mu_k)$.

We use the same notations for a point x in a set parameterized by μ , say \mathcal{E}_μ . We say that $x = O(\mu)$ (resp. $x = \Omega(\mu)$, $x \approx \mu$) whenever there is a constant K such that $\|x\| \leq K\mu$ (resp. $x^{-1} = O(1/\mu)$, $x = O(\mu)$ and $x = \Omega(\mu)$) for all $x \in \mathcal{E}_\mu$, and all small enough μ . In particular, $x \approx 1$ in \mathcal{E}_μ means that there are constants $K_2 > K_1 > 0$, such that any $x \in \mathcal{E}_\mu$ satisfies $K_1 \leq x_i \leq K_2$, $i = 1, \dots, n$.

Given two vector functions x and y , $x \approx y$ means that $x_i \approx y_i$ for $i = 1, \dots, n$, for small enough μ .

3.2 Main results

The monotone horizontal linear complementarity problem (LCP) is as follows: to find $(x, s) \in \mathbb{R}^n \times \mathbb{R}^n$ satisfying

$$\begin{aligned} xs &= 0, \\ Qx + Rs &= h, \\ x, s &\geq 0, \end{aligned} \tag{LCP}$$

where $h \in \mathbb{R}^n$, and $Q, R \in \mathbb{R}^{n \times n}$ are such that for any $u, v \in \mathbb{R}^n$,

$$Qu + Rv = 0 \text{ implies } u^T v \geq 0. \tag{3.2.1}$$

Note that the monotone horizontal linear complementarity problem is equivalent to the monotone linear complementarity problem in standard form (e.g. [14]), and LCP is more convenient for the asymptotic analysis, as we will see later.

We denote the feasible set, and the set of solutions of (LCP) as

$$\mathcal{F} := \{(x, s) \in \mathbb{R}_+^n \times \mathbb{R}_+^n; Qx + Rs = h\}, \quad (3.2.2)$$

$$\mathcal{S} := \{(x, s) \in \mathcal{F}; xs = 0\}. \quad (3.2.3)$$

Also, we denote the set of strictly complementary solutions by

$$\mathcal{S}^0 := \{(x, s) \in \mathcal{S}; x + s > 0\}. \quad (3.2.4)$$

It is well known that if (LCP) represents a linear programming problem, then if \mathcal{S} is nonempty so is \mathcal{S}^0 . This is not true in general, since it is easy to construct a quadratic program with nonempty \mathcal{S} and empty \mathcal{S}^0 . The existence of a strictly complementary solution will be an essential assumption in points 3.3.2, 3.3.3 and 3.3.4.

Let $(x^0, s^0, \mu_0) \in \mathbb{R}_+^n \times \mathbb{R}_+^n \times \mathbb{R}_{++}$ be given. Set

$$g = (h - Qx^0 - Rs^0)/\mu_0.$$

Then (x^0, s^0, μ_0) is an element of the set

$$\mathcal{F}^g := \{(x, s, \mu) \in \mathbb{R}_+^n \times \mathbb{R}_+^n \times \mathbb{R}_{++}; Qx + Rs = h - \mu g\}. \quad (3.2.5)$$

If in addition $x^0 s^0 = \mu_0 \mathbf{1}$, then (x^0, s^0, μ_0) belongs to the *infeasible central path pinned on g* , defined as

$$\begin{aligned} xs &= \mu \mathbf{1}, \\ Qx + Rs &= h - \mu g. \end{aligned} \quad (3.2.6)$$

We consider algorithms for solving (LCP) that follow approximately this infeasible central path, which whenever $g = 0$ (feasible case) coincide with the (usual) central path. We assume that the algorithms under consideration generate points (x, s, μ) belonging to a “small” neighborhood of the form

$$\mathcal{V}_\alpha := \{(x, s, \mu) \in \mathcal{F}^g; \|\frac{xs}{\mu} - \mathbf{1}\| \leq \alpha, \mu \leq \mu_0\},$$

where $\alpha > 0$ and $\mu_0 > 0$ are given constants, or to a “large” neighborhood

$$\mathcal{N}_\nu := \{(x, s, \mu) \in \mathcal{F}^g; \nu \mathbf{1} \leq \frac{xs}{\mu} \leq \nu^{-1} \mathbf{1}, \mu \leq \mu_0\}$$

where $0 < \nu < 1$ and $\mu_0 > 0$ are again given constants. It is easily seen that

$$\mathcal{V}_\alpha \subset \mathcal{N}_\nu \quad \text{for all } 0 < \nu \leq 1 - \alpha, \quad (3.2.7)$$

$$\mathcal{N}_\nu \subset \mathcal{V}_\alpha \quad \text{for all } \alpha \geq \sqrt{n} \left(\frac{1}{\nu} - 1 \right). \quad (3.2.8)$$

The algorithms studied in this paper start each iteration from data $(x, s, \mu) \in \mathcal{V}_\alpha$ (or \mathcal{N}_ν) and then compute two steps in \mathcal{V}_α (or \mathcal{N}_ν), a corrector step “near” to the central path and a predictor step with a smaller value of μ . Consider (3.2.6) with μ replaced by $\gamma\mu$ where $\gamma \in \{0, 1\}$. An approximate solution of (3.2.6) can be found by making a Newton step. Fixed γ , our algorithms compute a Newton direction (u, v) that is solution of the following perturbed linear system:

$$\begin{aligned} su + xv &= -xs + \gamma\mu\mathbf{1} + \mu\eta, \\ Qu + Rv &= (1 - \gamma)\mu g, \end{aligned} \quad (3.2.9)$$

where $\eta \in \mathbb{R}^n$ is a perturbation taking into account the error in the computation of the Newton step. (As the unperturbed right-hand side is of order μ , we may expect to deal with perturbations of order μ , i.e. η of order 1.) Note that the second equation is not perturbed. We can do this assumption without any loss of generality as the perturbed Newton step may be projected at a low cost into the set of solutions of the second equations in order to reduce to our framework. Furthermore, in the case of considering explicit perturbations only in the second equation of (3.2.9), it is not so easy to obtain explicit error estimates that guarantee a polynomial complexity of the algorithm (see [105]).

Let us note that under the monotonicity assumption (3.2.1) the system (3.2.9) has a unique solution. We denote by (u^c, v^c) the solution of (3.2.9) with $\gamma = 1$, called centering direction, and by (u^a, v^a) the solution of (3.2.9) with $\gamma = 0$, called affine-scaling direction. The centering direction and the affine-scaling direction are computed in the corrector step and predictor step respectively.

Finally, both steps choose a step-length $\theta \in (0, 1]$ such that the new point

$$x^\sharp = x + \theta u, \quad s^\sharp = s + \theta v, \quad \mu_\sharp := (1 - \theta + \theta\gamma)\mu, \quad (3.2.10)$$

belongs to \mathcal{V}_α (or to \mathcal{N}_ν).

Now we are ready to state a generic perturbed predictor corrector algorithm (**GPC**), in which the set \mathcal{G} denotes \mathcal{V}_α or \mathcal{N}_ν depending on the algorithm:

Algorithm GPC Data: $\mu_\infty > 0$, $(x^0, s^0, \mu_0) \in \mathcal{G}$. $k := 0$

REPEAT

- $x := x^k, s := s^k, \mu := \mu_k$;
- Corrector step: Compute (u^c, v^c) solution of (3.2.9) with $\gamma = 1$.
 $x(\theta) := x + \theta u^c, s(\theta) := s + \theta v^c$.
 Compute $\theta^c \in]0, 1]$ such that $(x(\theta^c), s(\theta^c), \mu) \in \mathcal{G}$.
 $x := x(\theta^c), s := s(\theta^c)$.
- Predictor step: Set $\gamma = 0$. Compute (u^a, v^a) solution of (3.2.9) with $\gamma = 0$.
 $x(\theta) := x + \theta u^a, s(\theta) := s + \theta v^a, \mu(\theta) := (1 - \theta)\mu$.
 Compute θ^a , the largest value in $]0, 1[$ such that
 $(x(\theta), s(\theta), \mu(\theta)) \in \mathcal{G}, \forall \theta \leq \theta^a$.
 $x^{k+1} := x(\theta^a), s^{k+1} := s(\theta^a), \mu_{k+1} := (1 - \theta^a)\mu_k$.
- $k := k + 1$.

UNTIL $\mu_k < \mu_\infty$.

In the following we consider two particular algorithms of the above general scheme and we state our main results in both cases.

The *perturbed predictor corrector algorithm in small neighborhoods* is defined as follows:

Algorithm SPC

Fix $\epsilon_c > 0, \epsilon_a > 0, \alpha \in]0, 1/2]$ and $\mathcal{G} = \mathcal{V}_\alpha$.

Specialize Algorithm **GPC** to this case with $\|\eta\| \leq \epsilon\alpha$, where $\epsilon \leq \epsilon_c$ during the corrector step and $\epsilon \leq \epsilon_a$ during the predictor step, and $\theta^c = 1$ at each iteration.

Note that in **GPC** we require that the point obtained after a restoration step is in the neighborhood \mathcal{G} , whereas we fix here $\theta^c = 1$. We check below that when ϵ_c is small enough, then the point obtained in a corrector step with $\theta^c = 1$ belongs to $\mathcal{V}_{\alpha/2}^g$. In addition, we give an explicit estimate of the size of errors for which the complexity of $O(\sqrt{n}L)$ for the feasible case (resp. $O(nL)$ for the infeasible case) is preserved. Assuming the strict complementarity hypothesis, we also compute a tighter bound under which a given asymptotic linear rate may be observed.

We say that (x^0, s^0) *dominates* (x^*, s^*) if $x^0 \geq x^*$ and $s^0 \geq s^*$. Similarly, we say that (x^0, s^0) *dominates* a solution of (LCP) if there exists $(x^*, s^*) \in \mathcal{S}$ which is dominated by (x^0, s^0) .

Theorem 1. Let $L := \log_2(\mu_0/\mu_\infty)$. We assume that $\epsilon_c \leq 1/12$ ($\epsilon_c \leq 1/4$ whenever $\alpha \leq 1/4$), and that (x^0, s^0) dominates a solution of (LCP) whenever it is not feasible. Then Algorithm **SPC** has the following properties:

- (i) (Feasible case) Assume (x^0, s^0) to be feasible. If $\epsilon_a \leq 1/4$, then the algorithm finds a feasible point such that $\mu_k \leq \mu_\infty$ in at most $3\sqrt{n/\alpha}L$ iterations.
- (ii) (Infeasible case) If $\epsilon_a = O(n)$, then the algorithm finds a point such that $\mu_k \leq \mu_\infty$ in at most $O(nL)$ iterations.
- (iii) (Asymptotic rate of convergence) Assume $\mathcal{S}^0 \neq \emptyset$. Let $\beta \in (0, 1)$ and ϵ_a be such that

$$\epsilon_a \left(1 + (1 - \beta) \sqrt{4 + \frac{\epsilon_a^2}{2}} \right) < \frac{\beta}{2(1 - \beta)}. \quad (3.2.11)$$

Then $\limsup \mu_{k+1}/\mu_k \leq \beta$. In particular, if $\epsilon_a \leq 1/5$, then $\mu_{k+1} \leq \mu_k/2$ for k large enough.

We now state the *perturbed predictor corrector algorithm in large neighborhoods*.

Algorithm LPC

Fix $\epsilon_c > 0$, $\epsilon_a > 0$, $\nu \in]0, 1/2]$ and $\mathcal{G} = \mathcal{N}_\nu$.

Specialize Algorithm **GPC** to this case with $\|\eta\|_\infty \leq \epsilon_c$ when computing the centering direction and $\|\eta\|_\infty \leq \epsilon_a$ when computing the affine direction, and

$$\theta^c = \min \left\{ 1, \frac{\mu}{2\|u^c v^c\|_\infty} \left(\frac{1}{2} - \|\eta\|_\infty \right) \right\} \quad (3.2.12)$$

at each iteration.

This choice of θ^c implies, as we will see, that the point obtained after a corrector step belongs to the interior of the large neighborhood, so that the algorithm is well defined. The theorem below also gives an explicit estimate of the size of errors for which the known complexity of the algorithm with $\eta = 0$, which is now $O(nL)$ for the feasible case and $O(n\sqrt{n}L)$ for the infeasible case, is preserved. Assuming the strict complementarity hypothesis, we also compute a tighter bound under which a given asymptotic linear rate may be observed.

Theorem 2. Let $L = \log_2(\mu_0/\mu_\infty)$. Assume that $\epsilon_c \leq 1/4$ and that (x^0, s^0) dominates a solution of (LCP) whenever it is not feasible. Then Algorithm **LPC** has the following properties:

- (i) (Feasible case) Assume (x^0, s^0) to be feasible. If $\epsilon_a \leq 1/4$, the algorithm finds a feasible point satisfying $\mu_k \leq \mu_\infty$ in at most $16nL/\nu^3$ iterations.
- (ii) (Infeasible case) If $\epsilon_a = O(\sqrt{n})$, then the algorithm finds a point satisfying $\mu_k \leq \mu_\infty$ in $O(n\sqrt{n}L)$ iterations.
- (iii) (Asymptotic rate of convergence) Assume $\mathcal{S}^0 \neq \emptyset$. Let $\beta \in (0, 1/2)$ and ϵ_a be such that

$$\epsilon_a \leq \frac{\nu^4 \beta}{64(1-\beta)^2 n^{3/2}}.$$

Then $\limsup \mu_{k+1}/\mu_k \leq \beta$.

3.3 Proof of main results

3.3.1 Preliminary results

In this section we start by recalling some known technical results and deriving an easy consequence of them. These results will be used for the analysis of both algorithms.

Set

$$\phi := \sqrt{\frac{xs}{\mu}}; \quad d := \sqrt{\frac{\mu x}{s}}; \quad \bar{u} := d^{-1}u; \quad \bar{v} := \frac{dv}{\mu}; \quad \bar{Q} := QD; \quad \bar{R} := \mu RD^{-1}.$$

Multiplying the first equation in (3.2.9) by $1/\sqrt{\mu xs}$, we obtain

$$\sqrt{\frac{s}{\mu x}}u + \frac{1}{\mu}\sqrt{\frac{\mu x}{s}}v = -\sqrt{\frac{xs}{\mu}} + \gamma\sqrt{\frac{\mu}{xs}} + \sqrt{\frac{\mu}{xs}}\eta.$$

Therefore (\bar{u}, \bar{v}) is solution of the scaled equation

$$\begin{cases} \bar{u} + \bar{v} &= -\phi + \gamma\phi^{-1} + \phi^{-1}\eta, \\ \bar{Q}\bar{u} + \bar{R}\bar{v} &= (1-\gamma)\mu g. \end{cases} \quad (3.3.1)$$

From (3.2.9) and (3.2.10) it is easily seen that

$$\frac{x^\sharp s^\sharp}{\mu^\sharp} = \frac{1-\theta}{1-\theta+\theta\gamma} \left(\frac{xs}{\mu} - \mathbf{1} \right) + \mathbf{1} + \frac{\theta}{1-\theta+\theta\gamma}\eta + \frac{\theta^2}{1-\theta+\theta\gamma} \frac{uv}{\mu}. \quad (3.3.2)$$

The first statement of the lemma below is due to Mizuno ([103], Lemma 1), and the second statement is due to Mizuno, Jarre and Stoer ([104], Corollary 1).

Lemma 1. i. If $y, z \in \mathbb{R}^n$ satisfies $y^T z \geq 0$, then $\|yz\| \leq \frac{1}{\sqrt{8}}\|y + z\|^2$.

ii. Let (\hat{u}, \hat{v}) be solution of

$$\begin{aligned}\hat{u} + \hat{v} &= \hat{f}, \\ Q\hat{u} + R\hat{v} &= \hat{g},\end{aligned}$$

and \hat{x}, \hat{s} such that $Q\hat{x} + R\hat{s} = \hat{g}$. Then

$$\begin{cases} \|\hat{u}\| \leq \|\hat{f}\| + \|\hat{x}\| + \|\hat{s}\|, \\ \|\hat{v}\| \leq \|\hat{f}\| + \|\hat{x}\| + \|\hat{s}\|. \end{cases}$$

The following technical estimates will be useful in the sequel.

Lemma 2. Let (x^0, s^0, μ_0) and (x, s, μ) be two elements of \mathcal{N}_ν such that (x^0, s^0) dominates a solution (x^*, s^*) of (LCP) . Set $(\tilde{x}, \tilde{s}) = (x^* - x^0, s^* - s^0)/\mu^0$. Then

$$\begin{aligned}x^T s^0 + s^T x^0 &\leq 4\mu_0 \frac{n}{\nu}, \\ Q\tilde{x} + R\tilde{s} &= g, \\ \|d^{-1}\tilde{x}\| + \left\|\frac{d\tilde{s}}{\mu}\right\| &\leq \frac{4n}{\nu\sqrt{\nu}\mu}.\end{aligned}$$

Preuve. By Theorem 2.2 of [17], we have

$$x^T s^0 + s^T x^0 \leq 2\mu_0 \frac{n}{\nu} + \xi,$$

where

$$\xi \leq (x^0)^T s^* + (s^0)^T x^* \leq 2(x^0)^T s^0 \leq 2\mu_0 \frac{n}{\nu}.$$

The first relation follows. The second statement is obvious. Let us prove the last one.

We have

$$\|d^{-1}\tilde{x}\| = \left\|\frac{\phi^{-1}}{\mu}s\tilde{x}\right\| \leq \frac{\|\phi^{-1}\|_\infty}{\mu}\|s\tilde{x}\| \leq \frac{\|\phi^{-1}\|_\infty}{\mu}\|s\tilde{x}\|_1 = \frac{\|\phi^{-1}\|_\infty}{\mu}s^T|\tilde{x}|,$$

and similarly

$$\left\|\frac{d\tilde{s}}{\mu}\right\| = \left\|\frac{\phi^{-1}}{\mu}x\tilde{s}\right\| \leq \frac{\|\phi^{-1}\|_\infty}{\mu}\|x\tilde{s}\| \leq \frac{\|\phi^{-1}\|_\infty}{\mu}\|x\tilde{s}\|_1 = \frac{\|\phi^{-1}\|_\infty}{\mu}x^T|\tilde{s}|.$$

As (x^0, s^0) dominates (x^*, s^*) , we obtain

$$\|d^{-1}\tilde{x}\| + \left\|\frac{d\tilde{s}}{\mu}\right\| \leq \frac{\|\phi^{-1}\|_\infty}{\mu} \left[s^T \frac{(x^0 - x^*)}{\mu_0} + x^T \frac{(s^0 - s^*)}{\mu_0} \right] \leq \frac{\|\phi^{-1}\|_\infty}{\mu\mu_0} (s^T x^0 + x^T s^0).$$

Combining with $\|\phi^{-1}\|_\infty \leq 1/\sqrt{\nu}$ and the first statement, the result follows. \square

The next lemma gives upper bounds for the term $\|uv\|/\mu$ of (3.3.2). Part **i** will be used in the centering step and in the feasible affine-scaling step, while part **ii** will be useful in the infeasible affine-scaling step.

Lemma 3. Let $(x, s, \mu) \in \mathcal{N}_\nu$ and (u, v) be solution of (3.2.9). Then

i. If $u^T v \geq 0$, then

$$\frac{\|uv\|}{\mu} \leq \frac{1}{\nu\sqrt{8}} \|\phi^2 + \gamma\mathbf{1} + \eta\|^2. \quad (3.3.3)$$

ii. If (x^0, s^0) dominates a solution of (LCP) and $\mu \leq \mu_0$, then the affine-scaling direction satisfies

$$\frac{\|u^a v^a\|}{\mu} \leq \left(\frac{\|\eta\|}{\sqrt{\nu}} + \frac{5n}{\nu\sqrt{\nu}} \right)^2. \quad (3.3.4)$$

Preuve. **i.** Using Lemma 1i and (3.3.1) we get

$$\begin{aligned} \frac{\|uv\|}{\mu} &= \|\bar{u}\bar{v}\| \leq \frac{1}{\sqrt{8}} \|\bar{u} + \bar{v}\|^2 = \frac{1}{\sqrt{8}} \|\phi + \gamma\phi^{-1} + \phi^{-1}\eta\|^2, \\ &\leq \frac{\|\phi^{-1}\|_\infty^2}{\sqrt{8}} \|\phi^2 + \gamma\mathbf{1} + \eta\|^2. \end{aligned}$$

Since $\|\phi^{-1}\|_\infty^2 \leq \nu^{-1}$, the result follows.

ii. Applying Lemma 1ii and Lemma 2 to the scaled equation (3.3.1), and using $\|\phi^{-1}\|_\infty \leq 1/\sqrt{\nu}$ and $\|\phi\| \leq \sqrt{n/\nu}$, we have whenever $\gamma = 0$

$$\begin{aligned} \|\bar{u}\| &\leq \|\phi + \phi^{-1}\eta\| + \mu \left(\|d^{-1}\tilde{x}\| + \left\| \frac{d\tilde{s}}{\mu} \right\| \right), \\ &\leq \frac{\sqrt{n}}{\sqrt{\nu}} + \frac{\|\eta\|}{\sqrt{\nu}} + 4\frac{n}{\nu\sqrt{\nu}} \leq \frac{\|\eta\|}{\sqrt{\nu}} + \frac{5n}{\nu\sqrt{\nu}}. \end{aligned}$$

The same estimate holds for \bar{v} . Using $\|\bar{u}\bar{v}\| \leq \|\bar{u}\|\|\bar{v}\|$, the result follows. \square

3.3.2 Asymptotic analysis

We now prove a general result that will be used for the study of the rapid asymptotic linear convergence in Section 3.3.3 and 3.3.4. Here, as in those sections, we assume (LCP) has a strictly complementary solution, i.e. $\mathcal{S}^0 \neq \emptyset$. It is well known that in this case there is a unique partition

$$B \cup N = \{1, 2, \dots, n\}, \quad B \cap N = \emptyset,$$

such that for any $(x, s) \in \mathcal{S}^0$ we have $x_B > 0, s_B = 0, x_N = 0$ and $s_N > 0$.

Following Bonnans and Gonzaga [14] we rename the variables in the following way

$$x \leftarrow (x_B, s_N) \quad \text{and} \quad s \leftarrow (s_B, x_N).$$

Algorithm **GPC** is invariant with respect to the permutation, whose advantage is just to simplify the analysis by assuming that $N = \emptyset$. Therefore in points 3.3.3 and 3.3.4 we will always refer to x as the vector of large variables and to s as the vector of small variables. Of course this change of variables can only be done in the analysis, it cannot be used by algorithms since B and N are unknown.

The following lemma indicates what are the order of magnitudes in the vicinity of the infeasible central path.

Lemme 4. ([17], Lemma 4.1) If $(x, s, \mu) \in \mathcal{G}$, then $x \approx 1, s \approx \mu$ and $d \approx 1$.

Define the scaled variables

$$\bar{x} := d^{-1}x, \quad \bar{s} := \frac{d}{\mu}s.$$

This scaling transfers x and s to the same vector $\bar{x} = d^{-1}x = \phi = ds/\mu = \bar{s}$. This scaling allows us to represent the solution of (3.2.9) as $O(\mu)$ perturbations of quantities that are easily analyzed. In order to do so we represent the solution of the scaled equations (3.3.1) in terms of orthogonal projections. Given $M \in \mathbb{R}^{m \times n}$, $q \in \mathbb{R}^m$ and the affine space defined by $Mx = q$, we define the projections operators $P_{M,q}$ and P_M by

$$x \rightarrow P_{M,q}x = \operatorname{argmin}\{\|w - x\| : Mw + q = 0\},$$

and $P_M := P_{M,0}$. Similarly, we denote $\tilde{P}_M = I - P_M$ the orthogonal projection on $\mathcal{R}(M^T)$. It is easily checked that $P_{M,q}x = P_Mx + P_{M,q}0$ for any $x \in \mathbb{R}^n$ and $q \in \mathbb{R}^m$.

Lemme 5. Let $(x, s, \mu) \in \mathcal{G}$.

i. (Feasible case) If (x, s) is feasible, then the solution of the scaled system (3.3.1) satisfies

$$\begin{aligned} \bar{u} &= \gamma P_{\bar{Q}}\phi^{-1} + P_{\bar{Q}}(\phi^{-1}\eta) + O(\mu). \\ \bar{v} &= -\phi + \gamma \tilde{P}_{\bar{Q}}\phi^{-1} + \tilde{P}_{\bar{Q}}(\phi^{-1}\eta) + O(\mu). \end{aligned}$$

ii. (Infeasible case) Let (\tilde{x}, \tilde{s}) be such that $Q\tilde{x} + R\tilde{s} = g$. Then the solution of the scaled system (3.3.1) satisfies

$$\begin{aligned}\bar{u} &= \gamma P_{\bar{Q}}(\phi^{-1} + d\tilde{s}) + P_{\bar{Q}}(\phi^{-1}\eta) + O(\mu). \\ \bar{v} &= -\phi + \gamma \tilde{P}_{\bar{Q}}(\phi^{-1} - P_{\bar{Q}}d\tilde{s}) + \tilde{P}_{\bar{Q}}(\phi^{-1}\eta) + O(\mu).\end{aligned}$$

Preuve. It is known (see [14]) that

$$Q\hat{u} + R\hat{v} = 0 \Rightarrow \hat{v} \in \mathcal{R}(Q^T). \quad (3.3.5)$$

Let us recall the proof for the sake of completeness of the paper. Indeed, if (\hat{u}, \hat{v}) satisfies the above relation, then $Q(\hat{u} + u') + R\hat{v} = 0$ for an arbitrary $u' \in \mathcal{N}(Q)$. Therefore $(\hat{u} + u')^T \hat{v} \geq 0$. As u' is an arbitrary element of the vector space $\mathcal{N}(Q)$, it follows that $(u')^T \hat{v} = 0$, i.e. $\hat{v} \in \mathcal{N}(Q)^\perp = \mathcal{R}(Q^T)$, as was to be proved.

i. In the feasible case we have $Qu + Rv = 0$. So, using (3.3.5) applied to the scaled system (3.3.1), it follows that $\bar{v} = dv/\mu \in \mathcal{R}(\bar{Q}^T)$. Set

$$f := -\phi + \gamma\phi^{-1} + \phi^{-1}\eta.$$

Then by (3.3.1):

$$\begin{aligned}\bar{u} &\in f + \mathcal{R}(\bar{Q}^T), \\ \bar{Q}\bar{u} + \bar{R}\bar{v} &= 0.\end{aligned} \quad (3.3.6)$$

This is the optimality system characterizing the orthogonal projection of f over the set defined by the second relation. Since $\bar{R}\bar{v} = Rv$, it follows that

$$\bar{u} = P_{\bar{Q}, Rv}f = P_{\bar{Q}}f + P_{\bar{Q}, Rv}0.$$

This expression can be simplified noticing that $P_{\bar{Q}}\phi = 0$. Indeed, let $(x^*, s^*) \in \mathcal{S}$, then $s^* = 0$ and $Q(x - x^*) + Rs = 0$; and therefore, using (3.3.5), $s \in \mathcal{R}(Q^T)$ and $\phi = ds/\mu \in \mathcal{R}(\bar{Q}^T)$. Hence, we deduce that

$$\bar{u} = P_{\bar{Q}}(\gamma\phi^{-1} + \phi^{-1}\eta) + P_{\bar{Q}, Rv}0. \quad (3.3.7)$$

We obtain the desired expression for \bar{u} by using linearity of $P_{\bar{Q}, Rv}$, and checking that

$$P_{\bar{Q}, Rv}0 = O(\mu). \quad (3.3.8)$$

Indeed, using $\eta = O(1)$, $\|\bar{u} + \bar{v}\| = \|f\| = O(1)$ and, as $\bar{u}^T \bar{v} \geq 0$, we deduce that $\|\bar{v}\| \leq \|f\| = O(1)$, whence, using Lemma 4, $\|v\| = \mu \|d^{-1} \bar{v}\| = O(\mu)$. Now, let Q^- be a right inverse for Q such that $\bar{Q}(D^{-1}Q^-Rv) = Rv$, whence

$$\|P_{\bar{Q}, Rv} 0\| \leq \|D^{-1}Q^-Rv\| = O(\|v\|) = O(\mu).$$

This proves the formula for \bar{u} , from which we deduce that

$$\bar{v} = f - \bar{u} = -\phi + \gamma\phi^{-1} + \phi^{-1}\eta - \gamma P_{\bar{Q}}\phi^{-1} - P_{\bar{Q}}(\phi^{-1}\eta) + O(\mu),$$

and so the last relation of part **i** follows.

ii. Now let us deal with the infeasible case. Set

$$\begin{aligned} \hat{u} &:= \bar{u} - (1 - \gamma)\mu d^{-1} \tilde{x}, \\ \hat{v} &:= \bar{v} - (1 - \gamma)\mu d \frac{\tilde{s}}{\mu}, \\ \hat{f} &:= -(1 - \gamma)\mu(d^{-1} \tilde{x} + d \frac{\tilde{s}}{\mu}). \end{aligned}$$

Then, by using (3.3.1), we get

$$\begin{cases} \hat{u} + \hat{v} &= f + \hat{f}, \\ \bar{Q}\hat{u} + \bar{R}\hat{v} &= 0. \end{cases} \quad (3.3.9)$$

Hence, as in the first part of the proof, we have

$$\begin{aligned} \hat{u} &= P_{\bar{Q}}(f + \hat{f}) + O(\mu), \\ &= P_{\bar{Q}}\left(-\phi - d\tilde{s} + \gamma(\phi^{-1} + d\tilde{s}) + \phi^{-1}\eta - \mu(1 - \gamma)d^{-1}\tilde{x}\right) + O(\mu). \end{aligned}$$

Let us check that $P_{\bar{Q}}(\phi + d\tilde{s}) = 0$. Let $(x^*, s^*) \in \mathcal{S}$, $s^* = 0$, then

$$Q(x + \mu\tilde{x} - x^*) + R(s + \mu\tilde{s}) = 0$$

and from (3.3.5) we deduce that $s + \mu\tilde{s} \in \mathcal{R}(Q^T)$. Therefore $\phi + d\tilde{s} = (d/\mu)(s + \mu\tilde{s}) \in \mathcal{R}(\bar{Q}^T) = \mathcal{N}(\bar{Q})^-$. Combining with the above display and Lemma 4 we deduce

$$\hat{u} = \gamma P_{\bar{Q}}(\phi^{-1} + d\tilde{s}) + P_{\bar{Q}}(\phi^{-1}\eta) + O(\mu).$$

Using $\bar{u} = \hat{u} + O(\mu)$, we obtain the formula for \bar{u} , from which we deduce the formula for $\bar{v} = f - \bar{u}$.

□

We will use the above lemma for the affine-scaling step in the rapid asymptotic linear convergence analysis.

3.3.3 The perturbed predictor corrector algorithm in small neighborhoods

Let us define the *centrality measure* as the mapping

$$\delta(x, s, \mu) = \left\| \frac{xs}{\mu} - \mathbf{1} \right\|.$$

If $(x, s, \mu) \in \mathcal{F}^g$ and $\delta(x, s, \mu) = 0$, then (x, s) is the *infeasible central point* associated with the parameter value μ .

Polynomial convergence

Centering step. We start by describing the effect of a centering step on the centrality measure.

Lemma 6. Let (x, s, μ) be such that $\delta(x, s, \mu) \leq \alpha$. If $\epsilon_c > 0$ is so small that

$$\epsilon_c + \frac{\alpha}{(1 - \alpha)\sqrt{8}}(1 + \epsilon_c)^2 \leq \frac{1}{2}, \quad (3.3.10)$$

then after a centering step, the centrality measure is not more than $\alpha/2$. In other words,

$$\delta_c := \left\| \frac{(x + u^c)(s + v^c)}{\mu} - \mathbf{1} \right\| \leq \frac{\alpha}{2}.$$

This holds in particular if $\epsilon_c \leq 1/12$ whenever $\alpha \leq 1/2$, and $\epsilon_c \leq 1/4$ whenever $\alpha \leq 1/4$.

Preuve. From (3.3.2) with $\gamma = \theta = 1$ we get

$$\frac{(x + u^c)(s + v^c)}{\mu} - \mathbf{1} = \eta + \frac{u^c v^c}{\mu}. \quad (3.3.11)$$

From (3.2.7) and (3.3.3), using $\delta(x, s, \mu) \leq \alpha$, $\gamma = 1$ and $\|\eta\| \leq \epsilon_c \alpha$, we have

$$\frac{\|u^c v^c\|}{\mu} \leq \frac{1}{(1 - \alpha)\sqrt{8}}(\alpha + \|\eta\|)^2 \leq \frac{1}{(1 - \alpha)\sqrt{8}}(1 + \epsilon_c)^2 \alpha^2. \quad (3.3.12)$$

Combining (3.3.12) and (3.3.11) and $\|\eta\| \leq \epsilon_c \alpha$ we deduce

$$\left\| \frac{(x + u^c)(s + v^c)}{\mu} - \mathbf{1} \right\| \leq \epsilon_c \alpha + \frac{1}{(1 - \alpha)\sqrt{8}}(1 + \epsilon_c)^2 \alpha^2.$$

From (3.3.10) we obtain the conclusion. □

Affine-scaling step. We now analyse the effect of an affine-scaling step on the centrality measure, beginning by considering an upper bound for this measure. From (3.3.2) with $\gamma = 0$, we get

$$\begin{aligned} \left\| \frac{(x + \theta u^a)(s + \theta v^a)}{\mu^\sharp} - \mathbf{1} \right\| &= \left\| \frac{xs}{\mu} - \mathbf{1} + \frac{\theta}{1-\theta}\eta + \frac{\theta^2}{1-\theta} \frac{u^a v^a}{\mu} \right\|, \\ &\leq \left\| \frac{xs}{\mu} - \mathbf{1} \right\| + \frac{\theta}{1-\theta} \|\eta\| + \frac{\theta^2}{1-\theta} \frac{\|u^a v^a\|}{\mu}. \end{aligned}$$

By Lemma 6, the centrality measure after a centering step is at most $\alpha/2$. Hence we deduce that the point obtained with a value θ of the step-length along the direction (u^a, v^a) belongs to the small neighborhood whenever

$$\frac{\theta}{1-\theta} \|\eta\| + \frac{\theta^2}{1-\theta} \frac{\|u^a v^a\|}{\mu} \leq \frac{\alpha}{2}. \quad (3.3.13)$$

Therefore the main point is to estimate $\|u^a v^a\|$.

Lemme 7. Let $(x, s, \mu) \in \mathcal{V}_{\alpha/2}^g$. Then

i. (Feasible case) If (x, s) is feasible, we have

$$\frac{\|u^a v^a\|}{\mu} \leq n \frac{(1 + \alpha(1/2 + \epsilon_a))^2}{(1 - \alpha/2)\sqrt{8}}. \quad (3.3.14)$$

If in addition $0 < \epsilon_a \leq 1/2$ and $\hat{\theta} \leq 1/2$ verifies

$$n\hat{\theta}^2 \frac{(1 + \alpha(1/2 + \epsilon_a))^2}{(1 - \alpha/2)\sqrt{2}} \leq \left(\frac{1}{2} - \epsilon_a\right) \alpha, \quad (3.3.15)$$

then $\delta(x + \theta u^a, s + \theta v^a, (1 - \theta)\mu) \leq \alpha$, $\forall \theta \in (0, \hat{\theta}]$. In particular, if $\epsilon_a \leq 1/4$ then $\theta^a \geq \frac{1}{3}\sqrt{\alpha/n}$.

ii. (Infeasible case) Let (x^0, s^0) dominate a solution of (LCP) . If $\epsilon_a = O(n)$, then $\theta^a = \Omega(1/n)$.

Preuve. i. Remembering that $\delta(x, s, \mu) \leq \alpha/2$, and using $(u^a)^T v^a \geq 0$, we deduce from (3.2.7) and (3.3.3) with $\gamma = 0$, that

$$\frac{\|u^a v^a\|}{\mu} \leq \frac{1}{(1 - \alpha/2)\sqrt{8}} \|\phi^2 - \eta\|^2.$$

Using $\|\phi^2\| \leq \sqrt{n} \|\phi^2\|_\infty \leq \sqrt{n}(1 + \alpha/2)$, and $\|\eta\| \leq \epsilon_a \alpha$, we obtain

$$\frac{\|u^a v^a\|}{\mu} \leq \frac{[\sqrt{n} + \alpha(\epsilon_a + \sqrt{n}/2)]^2}{(1 - \alpha/2)\sqrt{8}} \leq n \frac{(1 + \alpha(1/2 + \epsilon_a))^2}{(1 - \alpha/2)\sqrt{8}}.$$

This proves (3.3.14).

Using $\delta(x, s, \mu) \leq \alpha/2$, (3.3.13) and $\|\eta\| \leq \epsilon_a \alpha$, we deduce that θ is feasible whenever

$$\frac{\theta}{1 - \theta} \epsilon_a \alpha + \frac{\theta^2}{1 - \theta} \frac{\|u^a v^a\|}{\mu} \leq \alpha/2.$$

As the function $f : (0, 1) \rightarrow \mathbb{R}$ defined by

$$f(\theta) = \frac{\theta}{1 - \theta} \epsilon_a \alpha + \frac{\theta^2}{1 - \theta} \frac{\|u^a v^a\|}{\mu}$$

is increasing, for obtaining the desired inequality it suffices to show that $f(\hat{\theta}) \leq \alpha/2$. Using $\hat{\theta} \leq \frac{1}{2}$ and $\frac{1}{1 - \hat{\theta}} \leq 2$, we get

$$f(\hat{\theta}) \leq \epsilon_a \alpha + 2\hat{\theta}^2 \frac{\|u^a v^a\|}{\mu}.$$

Hence using part **i** and (3.3.15) we obtain that after a displacement step of value $\hat{\theta}$, the new point belongs to \mathcal{V}_α .

For the proof of the last statement, observe that if $\theta^a \geq 1/2$, the result is obvious. Otherwise it suffices to prove that $\hat{\theta} := \frac{1}{3}\sqrt{\alpha/n}$ satisfies (3.3.15). In fact, as $\alpha \leq 1/2$ and $0 < \epsilon_a \leq 1/4$, we have

$$n\hat{\theta}^2 \frac{(1 + \alpha(1/2 + \epsilon_a))^2}{(1 - \alpha/2)\sqrt{2}} \leq n\hat{\theta}^2 \frac{(11/8)^2}{\frac{3}{4}\sqrt{2}} < \frac{1}{4}\alpha \leq \left(\frac{1}{2} - \epsilon_a\right)\alpha.$$

ii. Combining with (3.3.4) and (3.3.13), we deduce that θ is feasible whenever

$$\theta\|\eta\| + \theta^2 \left(\frac{\|\eta\|}{\sqrt{1 - \alpha}} + \frac{5n}{(1 - \alpha)^{3/2}} \right)^2 \leq (1 - \theta)\frac{\alpha}{2}.$$

If $\theta^a \geq 1/2$, the conclusion is obtained. Otherwise, the right-hand-side is greater than $\alpha/4$. Assuming $\|\eta\| \leq cn$, we see that θ is feasible whenever

$$(n\theta)c + \left(\frac{c}{\sqrt{1 - \alpha}} + \frac{5}{(1 - \alpha)^{3/2}} \right)^2 (n\theta)^2 \leq \frac{\alpha}{4}.$$

The left-hand-side is null when $\theta = 0$. Therefore, the inequality holds whenever $n\theta$ is small enough. The result follows. \square

Using the above results we can easily prove in Subsection 3.3.3 that Algorithm SPC has polynomial convergence.

Rapid asymptotic linear convergence

We now turn to the analysis of the asymptotic speed of convergence of the sequence μ_k .

Lemme 8. One has

i.

$$\frac{\|u^a v^a\|}{\mu} \leq \|\phi^{-1}\|_\infty \|\eta\| \left(\frac{\|\phi^{-1}\|_\infty \|\eta\|}{\sqrt{8}} + \|\phi\|_\infty \right).$$

ii. If in addition $\delta(x, s, \mu) \leq \alpha$ and $\|\eta\| \leq \epsilon_a \alpha$, then

$$\frac{\|u^a v^a\|}{\mu} \leq \epsilon_a \alpha \sqrt{4 + \frac{\epsilon_a^2}{2}} + O(\mu).$$

Preuve. Define $\bar{u}^a := d^{-1}u^a$, $\bar{v}^a := d \frac{v^a}{\mu}$. From Lemma 5 with $\gamma = 0$ we get

$$\frac{u^a v^a}{\mu} = \bar{u}^a \bar{v}^a = (P_{\bar{Q}} \phi^{-1} \eta)(-\phi + \tilde{P}_{\bar{Q}} \phi^{-1} \eta) + O(\mu). \quad (3.3.16)$$

Using Lemma 1 i, and the fact that $P_{\bar{Q}}$ and $\tilde{P}_{\bar{Q}}$ are contractions, we have

$$\begin{aligned} \|(P_{\bar{Q}} \phi^{-1} \eta)(-\phi + \tilde{P}_{\bar{Q}} \phi^{-1} \eta)\| &\leq \|(P_{\bar{Q}} \phi^{-1} \eta)(\tilde{P}_{\bar{Q}} \phi^{-1} \eta)\| + \|(P_{\bar{Q}} \phi^{-1} \eta)\phi\|, \\ &\leq \frac{1}{\sqrt{8}} \|\phi^{-1} \eta\|^2 + \|\phi^{-1} \eta\| \|\phi\|_\infty, \\ &= \|\phi^{-1} \eta\| \left(\frac{\|\phi^{-1} \eta\|}{\sqrt{8}} + \|\phi\|_\infty \right), \\ &\leq \|\phi^{-1}\|_\infty \|\eta\| \left(\frac{\|\phi^{-1}\|_\infty \|\eta\|}{\sqrt{8}} + \|\phi\|_\infty \right), \end{aligned}$$

proving i. Using $\|\phi^{-1}\|_\infty \leq \frac{1}{\sqrt{1-\alpha}}$, $\|\phi\|_\infty \leq \sqrt{1+\alpha}$ and $\|\eta\| \leq \epsilon_a \alpha$, we deduce that

$$\|(P_{\bar{Q}} \phi^{-1} \eta)(-\phi + \tilde{P}_{\bar{Q}} \phi^{-1} \eta)\| \leq \frac{\epsilon_a \alpha}{1-\alpha} \left(\frac{\epsilon_a \alpha}{\sqrt{8}} + \sqrt{1-\alpha^2} \right).$$

Since the concave function $\alpha \rightarrow \frac{\epsilon_a \alpha}{\sqrt{8}} + \sqrt{1 - \alpha^2}$ attains its maximum on $[0, 1/2]$ at $\frac{\epsilon_a}{\sqrt{\epsilon_a^2 + 8}}$, we have, using $(1 - \alpha)^{-1} \leq 2$:

$$\begin{aligned} \|(P_{\tilde{Q}}\phi^{-1}\eta)(-\phi + \tilde{P}_{\tilde{Q}}\phi^{-1}\eta)\| &\leq 2\epsilon_a\alpha \left(\frac{\epsilon_a^2}{\sqrt{8}\sqrt{\epsilon_a^2 + 8}} + \sqrt{\frac{8}{\epsilon_a^2 + 8}} \right) \\ &= 2\epsilon_a\alpha \frac{\epsilon_a^2 + 8}{\sqrt{8}\sqrt{\epsilon_a^2 + 8}} = \epsilon_a\alpha \frac{\sqrt{\epsilon_a^2 + 8}}{\sqrt{2}} = \epsilon_a\alpha \sqrt{4 + \frac{\epsilon_a^2}{2}}. \end{aligned}$$

Combining the above inequality and (3.3.16) we obtain the conclusion. \square

Lemma 9. Let $\beta \in (0, 1)$ and (x, s, μ) such that $\delta(x, s, \mu) \leq \alpha/2$. If there exists a strictly complementary solution and (3.2.11) is satisfied, then $\theta^a \geq 1 - \beta$ for k large enough.

Preuve. Using the same function f as in proof of Lemma 7, it suffices to show $f(1 - \beta) \leq \alpha/2$ to obtain the conclusion. From Lemma 8, we get

$$\begin{aligned} f(1 - \beta) &= \frac{1 - \beta}{\beta} \epsilon_a \alpha + \frac{(1 - \beta)^2 \|u^a v^a\|}{\beta \mu}, \\ &\leq \frac{1 - \beta}{\beta} \epsilon_a \alpha + \epsilon_a \alpha \frac{(1 - \beta)^2}{\beta} \sqrt{4 + \frac{\epsilon_a^2}{2}} + O(\mu), \\ &= \alpha \epsilon_a \frac{1 - \beta}{\beta} \left(1 + (1 - \beta) \sqrt{4 + \frac{\epsilon_a^2}{2}} \right) + O(\mu). \end{aligned}$$

Hence, using (3.2.11) the result follows. \square

In the next subsection we will use the previous results for proving the asymptotic rate of convergence of Algorithm LPC.

Proof of Theorem 1

i. By Lemma 6, we know that the centrality measure after a centering step is at most $\alpha/2$. In the feasible case, from Lemma 7 **i** the step-length of Algorithm **SPC** satisfies $\theta^a \geq \frac{1}{3}\sqrt{\alpha/n}$. Then $\mu_k \leq \left(1 - \frac{1}{3}\sqrt{\frac{\alpha}{n}}\right)^k \mu_0$. Using $\left| \log_2 \left(1 - \frac{1}{3}\sqrt{\alpha/n}\right) \right| \geq \frac{1}{3}\sqrt{\alpha/n}$ we obtain the conclusion.

ii. Similarly, in the infeasible case, from $\theta^a = \Omega(1/n)$, obtained in part ii of Lemma 7, we deduce that no more than $O(nL)$ iterations is necessary.

iii. This is a simple consequence of Lemmas 6 and 9.

3.3.4 The perturbed predictor corrector algorithm using large neighborhoods

In order to measure how interior is a point (x, s, μ) with respect to \mathcal{N}_ν , we will use the distance of xs/μ to the boundary of the set

$$\mathcal{T}^\nu = \{z \in \mathbb{R}^n; \nu \mathbf{1} \leq z \leq \nu^{-1} \mathbf{1}\},$$

measured in the infinity norm.

Polynomial convergence

Centering step. Let us denote

$$x^c = x + \theta^c u^c, \quad s^c = s + \theta^c v^c.$$

Lemma 10. Let $(x, s, \mu) \in \mathcal{N}_\nu$ and $\nu \in (0, 1/2]$. If $\epsilon_c \leq 1/4$, then $(x^c, s^c, \mu) \in \mathcal{N}_\nu$ and

$$\text{dist}\left(\frac{x^c s^c}{\mu}, \partial \mathcal{T}^\nu\right) \geq \frac{\nu^3 \sqrt{2}}{32n}. \quad (3.3.17)$$

Preuve. From (3.3.2) with $\gamma = 1$, we obtain

$$\begin{aligned} \frac{x^c s^c}{\mu} &= (1 - \theta^c) \left(\frac{xs}{\mu} - \mathbf{1} \right) + \mathbf{1} + \theta^c \eta + (\theta^c)^2 \frac{u^c v^c}{\mu} \\ &= (1 - \theta^c) \frac{xs}{\mu} + \theta^c \mathbf{1} + \theta^c \eta + (\theta^c)^2 \frac{u^c v^c}{\mu}. \end{aligned} \quad (3.3.18)$$

Using $(x, s, \mu) \in \mathcal{N}_\nu$ and $\theta^c \leq 1$ we get

$$\begin{aligned} (\nu + \theta^c(1 - \nu)) \mathbf{1} &= (1 - \theta^c) \nu \mathbf{1} + \theta^c \mathbf{1} \leq (1 - \theta^c) \frac{xs}{\mu} + \theta^c \mathbf{1} \leq \left((1 - \theta^c) \frac{1}{\nu} + \theta^c \right) \mathbf{1}, \\ &= \left(\frac{1}{\nu} + \theta^c \left(1 - \frac{1}{\nu} \right) \right) \mathbf{1}. \end{aligned}$$

As

$$\theta^c \left| 1 - \frac{1}{\nu} \right| = \theta^c \frac{|\nu - 1|}{\nu} \geq \theta^c (1 - \nu),$$

we have

$$\text{dist}\left(\left(1 - \theta^c\right)\frac{xs}{\mu} + \theta^c \mathbf{1}, \partial\mathcal{T}^\nu\right) \geq \theta^c (1 - \nu) \geq \frac{\theta^c}{2}.$$

Using (3.3.18), the above inequality and $\theta^c > 0$, we get

$$\begin{aligned} \text{dist}\left(\frac{x^c s^c}{\mu}, \partial\mathcal{T}^\nu\right) &\geq \text{dist}\left(\left(1 - \theta^c\right)\frac{xs}{\mu} + \theta^c \mathbf{1}, \partial\mathcal{T}^\nu\right) - \theta^c \|\eta\|_\infty - (\theta^c)^2 \frac{\|u^c v^c\|_\infty}{\mu}, \\ &\geq \theta^c \left(\frac{1}{2} - \|\eta\|_\infty\right) - (\theta^c)^2 \frac{\|u^c v^c\|_\infty}{\mu}. \end{aligned}$$

Let us first consider the case when $\theta^c = 1$. Then, from (3.2.12),

$$\frac{\|u^c v^c\|_\infty}{\mu} \leq \left(\frac{1}{2} - \|\eta\|_\infty\right)/2.$$

Using $\|\eta\|_\infty \leq 1/4$, we deduce that $\text{dist}\left(\frac{x^c s^c}{\mu}, \partial\mathcal{T}^\nu\right) \geq \left(\frac{1}{2} - \|\eta\|_\infty\right)/2 \geq 1/8$, and the result follows. Otherwise, by (3.2.12):

$$\text{dist}\left(\frac{x^c s^c}{\mu}, \partial\mathcal{T}^\nu\right) \geq \left(\frac{1}{2} - \|\eta\|_\infty\right)^2 \frac{\mu}{4\|u^c v^c\|_\infty} \geq \frac{\mu}{2^6 \|u^c v^c\|_\infty}.$$

Using (3.3.3), $(x, s, \mu) \in \mathcal{N}_\nu$ and $\|\eta\|_\infty \leq 1$, we obtain

$$\begin{aligned} \frac{\|u^c v^c\|_\infty}{\mu} &\leq \frac{\|u^c v^c\|}{\mu} \leq \frac{1}{\nu\sqrt{8}} \left\| \frac{xs}{\mu} - \mathbf{1} + \eta \right\|^2 \leq \frac{n}{\nu\sqrt{8}} \left(\left\| \frac{xs}{\mu} - \mathbf{1} \right\|_\infty + \|\eta\|_\infty \right)^2, \\ &\leq \frac{n}{\nu\sqrt{8}} \left(\left(\frac{1}{\nu} - 1\right) + \|\eta\|_\infty \right)^2 \leq \frac{n}{\nu^3\sqrt{8}}. \end{aligned}$$

Combining with the previous inequality, we obtain the conclusion. \square

Affine-scaling step. From (3.3.2) with $\gamma = 0$, denoting by $(x^\sharp, s^\sharp, \mu_\sharp)$ the point obtained after a step of value θ , we get

$$\frac{x^\sharp s^\sharp}{\mu_\sharp} = \frac{xs}{\mu} + \frac{\theta}{1 - \theta} \eta + \frac{\theta^2}{1 - \theta} \frac{u^a v^a}{\mu} \quad (3.3.19)$$

and therefore, by Lemma 10,

$$\text{dist}\left(\frac{x^\sharp s^\sharp}{\mu_\sharp}, \partial\mathcal{T}^\nu\right) \geq \frac{\nu^3\sqrt{2}}{32n} - \frac{\theta}{1 - \theta} \|\eta\|_\infty - \frac{\theta^2}{1 - \theta} \frac{\|u^a v^a\|_\infty}{\mu}. \quad (3.3.20)$$

The following technical lemma is used in the proof of Theorem 2.

Lemme 11. Let $(x, s, \mu) \in \mathcal{N}_\nu$.

i. Assume that (x, s) is feasible. If $\epsilon_a \leq 1/4$, then

$$\frac{\|u^a v^a\|_\infty}{\mu} \leq \frac{n}{\nu^3}. \quad (3.3.21)$$

Moreover if $\text{dist}\left(\frac{xs}{\mu}, \partial\mathcal{T}^\nu\right) \geq \frac{\nu^3\sqrt{2}}{32n}$ then $\bar{\theta} := \frac{\nu^3}{16n}$ satisfies

$$(x + \theta u^a, s + \theta v^a, (1 - \theta)\mu) \in \mathcal{N}_\nu, \quad \forall \theta \in (0, \bar{\theta}].$$

ii. If (x^0, s^0) dominate a solution of (LCP), $\text{dist}\left(\frac{xs}{\mu}, \partial\mathcal{T}^\nu\right) \geq \frac{\nu^3\sqrt{2}}{32n}$ and $\epsilon_a = O(\sqrt{n})$, then $\theta^a = \Omega(1/(n\sqrt{n}))$.

Preuve. i. Whenever (x, s) is feasible, we have $(u^a)^T v^a \geq 0$. Hence from (3.3.3) with $\gamma = 0$, using $(x, s, \mu) \in \mathcal{N}_\nu$, $\epsilon_a \leq 1/4$ and $\nu \leq 1$, we get

$$\begin{aligned} \frac{\|u^a v^a\|}{\mu} &\leq \frac{1}{\nu\sqrt{8}} \left\| \frac{xs}{\mu} - \eta \right\|^2 \leq \frac{n}{\nu\sqrt{8}} \left(\left\| \frac{xs}{\mu} \right\|_\infty + \|\eta\|_\infty \right)^2, \\ &\leq \frac{n}{\nu\sqrt{8}} \left(\frac{1}{\nu} + \frac{1}{4} \right)^2 \leq \frac{n}{\nu\sqrt{8}} \frac{25}{16\nu^2} \leq \frac{n}{\nu^3}. \end{aligned}$$

This proves (3.3.21).

It is easily checked that for all $\theta \in (0, \bar{\theta}]$, we have

$$2\frac{n}{\nu^3}\theta^2 + \frac{\theta}{2} \leq \frac{\nu^3\sqrt{2}}{32n}. \quad (3.3.22)$$

Using (3.3.20) and (3.3.21) we have whenever $\theta \leq 1/2$ and $\|\eta\|_\infty \leq 1/4$

$$\text{dist}\left(\frac{x^\sharp s^\sharp}{\mu^\sharp}, \partial\mathcal{T}^\nu\right) \geq \frac{\nu^3\sqrt{2}}{32n} - \frac{\theta}{2} - 2\theta^2 \frac{n}{\nu^3}.$$

In view of (3.3.22), the right-hand-side is positive whenever $\theta \in (0, \bar{\theta}]$. The conclusion follows.

ii. If $\theta^a \geq 1/2$, the conclusion is obtained. Otherwise with (3.3.20) and (3.3.4), we get

$$\text{dist}\left(\frac{x^\sharp s^\sharp}{\mu^\sharp}, \partial\mathcal{T}^\nu\right) \geq \frac{\nu^3\sqrt{2}}{32n} - 2\theta\|\eta\|_\infty - 2\theta^2 \left(\frac{\sqrt{n}\|\eta\|_\infty}{\sqrt{\nu}} + \frac{5n}{\nu\sqrt{\nu}} \right)^2.$$

Assuming $\|\eta\|_\infty \leq c\sqrt{n}$, we see that θ is feasible whenever

$$\frac{\nu^3\sqrt{2}}{32} - 2\theta cn\sqrt{n} - 2\theta^2 n \left(\frac{cn}{\sqrt{\nu}} + \frac{5n}{\nu\sqrt{\nu}} \right)^2 \geq 0.$$

Therefore θ is feasible whenever

$$\frac{\nu^3\sqrt{2}}{64} \geq c(\theta n\sqrt{n}) + (\theta n\sqrt{n})^2 \left(\frac{c}{\sqrt{\nu}} + \frac{5}{\nu\sqrt{\nu}} \right)^2.$$

The right-hand-side is null when $\theta = 0$. Therefore, the inequality holds whenever $\theta n\sqrt{n}$ is small enough. The result follows. \square

Using the above results we will prove in Subsection 3.3.4 that Algorithm **LPC** has polynomial convergence.

Asymptotic rate of convergence

Lemma 12. Let $(x, s, \mu) \in \mathcal{N}_\nu$. If there exists a strictly complementary solution and $\epsilon_a \leq \sqrt{8/n}$ then

$$\frac{\|u^a v^a\|_\infty}{\mu} \leq 2 \frac{\sqrt{n}}{\nu} \epsilon_a.$$

Preuve. Using (3.2.8) and Lemma 8i, we obtain

$$\frac{\|u^a v^a\|_\infty}{\mu} \leq \frac{\|u^a v^a\|}{\mu} \leq \|\phi^{-1}\|_\infty \|\eta\| \left(\frac{\|\phi^{-1}\|_\infty \|\eta\|}{\sqrt{8}} + \|\phi\|_\infty \right).$$

Using $\|\phi^{-1}\|_\infty \leq 1/\sqrt{\nu}$, $\|\phi\|_\infty \leq 1/\sqrt{\nu}$, $\|\eta\| \leq \sqrt{n}\|\eta\|_\infty$, $\|\eta\|_\infty \leq \epsilon_a$ and $\epsilon_a \leq \sqrt{8/n}$, it follows that

$$\frac{\|u^a v^a\|_\infty}{\mu} \leq \frac{\epsilon_a \sqrt{n}}{\nu} \left(\frac{\epsilon_a \sqrt{n}}{\sqrt{8}} + 1 \right) \leq \frac{2\sqrt{n}}{\nu} \epsilon_a.$$

Combining the above inequality and (3.3.4) we obtain the conclusion. \square

Lemma 13. Let $\beta \in (0, 1/2)$, $0 \leq \nu \leq 1/2$ and $(x, s, \mu) \in \mathcal{N}_\nu$ such that $\text{dist}\left(\frac{xs}{\mu}, \partial\mathcal{T}^\nu\right) \geq \frac{\nu^3\sqrt{2}}{32n}$. If there exists a strictly complementary solution and

$$\epsilon_a \leq \frac{\nu^4\beta}{64(1-\beta)^2 n^{3/2}},$$

then $\theta^a \geq 1 - \beta$ for k large enough.

Preuve. It suffices to show that the right-hand-side of (3.3.20) is positive whenever $\theta \leq 1 - \beta$, i.e.

$$\frac{1 - \beta}{\beta} \epsilon_a + \frac{(1 - \beta)^2 \|u^a v^a\|_\infty}{\beta \mu} \leq \frac{\nu^3 \sqrt{2}}{32n}.$$

By Lemma 12, this will be satisfied if

$$\frac{1 - \beta + \frac{2\sqrt{n}}{\nu}(1 - \beta)^2}{\beta} \epsilon_a < \frac{\nu^3 \sqrt{2}}{32n},$$

i.e.

$$\epsilon_a < \frac{\beta}{1 - \beta + \frac{2\sqrt{n}}{\nu}(1 - \beta)^2} \frac{\nu^3 \sqrt{2}}{32n} = \frac{1}{\nu + 2\sqrt{n}(1 - \beta)} \frac{\nu^4 \beta}{(1 - \beta)n} \frac{\sqrt{2}}{32}.$$

The conclusion follows. \square

The results of this subsection will be useful for obtaining the asymptotic rate of convergence of Algorithm LPC.

Proof of Theorem 2

i. From Lemma 10, we get

$$\text{dist}\left(\frac{x^c s^c}{\mu}, \partial \mathcal{T}^\nu\right) \geq \frac{\nu^3 \sqrt{2}}{32n};$$

hence from Lemma 11 **i** the step-length of Algorithm **LPC** satisfies $\theta^a \geq \nu^3/16n$.

We obtain the conclusion following the same argument as in proof of part **i** of Theorem 1.

ii. Similarly, in the infeasible case, from $\theta^a = \Omega(\frac{1}{n\sqrt{n}})$, obtained in part **ii** of Lemma 11, we deduce that no more than $O(n\sqrt{n}L)$ iterations are necessary.

iii. This is an immediate consequence of Lemma 13.

3.4 Numerical experiments

In this section we present some numerical results that strongly support the theoretical estimates obtained in the preceding sections. These experiments are limited to

the large neighborhood predictor corrector algorithm, choosing the size of the neighborhood $\nu = 0.01$. The algorithm is the one described in this paper, in which the step-size for the centering displacement is as follows: starting from a unit step, we divide the step by two until the new point belongs to the large neighborhood. This is a rather rough linear search. Because it proved to be efficient, we content of it. We apply this algorithm to a family of linear programming problems in standard form, i.e.

$$\text{Min}_x c^T x ; Ax = b ; x \geq 0, \quad (3.4.1)$$

where $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^p$. The necessary and sufficient optimality conditions for this problem are

$$\begin{aligned} xs &= 0, \\ Ax &= b, \\ c + A^T \lambda &= s, \\ x \geq 0, s &\geq 0, \end{aligned}$$

and may be formally reduced to the linear complementarity format by writing the third condition in the equivalent form $B^T c = B^T s$, where B is a matrix whose columns form a basis of the kernel of A . The associated perturbed Newton step $(u, v, \delta\lambda)$ is solution of

$$\begin{aligned} su + xv &= -xs + \gamma\mu\mathbf{1} + \mu\eta, \\ Au &= (1 - \gamma)\mu g_P, \\ A^T \delta\lambda - v &= (1 - \gamma)\mu g_D, \end{aligned}$$

where g_P and g_D depend on the starting point.

In order to test numerically the robustness of the Newton direction, we choose to generate some perturbations of the right-hand-side of the corresponding linear system in a random way. Then we compare the number of iterations to the one obtained without perturbation.

We generate the data of the problem to be solved as follows. Given an even value for n , we fix $p = n/2$. Then the matrix A is randomly generated (we performed all computations and generations of random numbers using the standard MATLAB functions). We randomly generate two nonnegative vectors \hat{x} and \hat{s} . Then we take

$b = A\hat{x}$ and $c = \hat{s}$. In that way, \hat{x} and $(\hat{s}, \hat{\lambda} = 0)$ are primal and dual feasible, and therefore the linear problem has solutions. The starting point is $x^0 = s^0 = \mathbf{1}$ and $\mu_0 = 1$. The algorithm stops when $\mu < 10^{-10}$.

The test were run on a DEC Alpha 3000 with 62MB of main memory (the machine precision is approximately $2.2204e - 16$).

The perturbation η is chosen so that it satisfies at each iteration

$$\mu\|\eta\| = \epsilon\|f\|,$$

where $f = -xs + \gamma\mu\mathbf{1}$ is the right-hand-side of the equations of the Newton step corresponding to the linearization of the complementarity condition. That is, the perturbation parameter ϵ is the ratio (measured in the L^2 norm) between the perturbation and the right hand side. It varies between 0 and 0.25. Note that $\eta = \epsilon\mu^{-1}\|f\|z$, where z belongs to the unit sphere of \mathbb{R}^n . One difficulty is that we do not have a direct mean for generating an element of the unit sphere of \mathbb{R}^n with uniform probability. When generating at random the components of z we obtain a distribution that is too “uniform”. It seems that the perturbation is more difficult to handle with if it is localized on a limited number of components. Therefore we propose two ways for generating z .

The first method consists in generating at random each component of a vector \hat{z} . Then we obtain z by changing to 0 half of the components of \hat{z} , i.e. we have either $z_i = \hat{z}_i$ or $z_i = 0$, $i = 1, \dots, n$, and finally we renormalize z in order to have $\|z\| = 1$.

Our results are a mean over ten runs except for $n = 10000$ where we perform only three runs. Using this first method for generating z , we obtain the results of tables 1 and 2.

We now consider a second method of generating the direction z where we choose to concentrate the perturbation in one component; i.e., all components of z are 0 except one, of value ± 1 , taken at random. We test this perturbation method only in the case $n = 10000$. Table 3 and 4 summarize the obtained results. We perform three runs for each value of ϵ .

From these results, we may conclude that, at least for randomly generated problems, the large neighborhood predictor corrector algorithm described in this paper is both rapid and very robust with respect to perturbations in the computation of the Newton step.

Table 3.4.1: Mean value of the number of iterations.

n/ϵ	0	0.05	0.1	0.15	0.2	0.25
10	9.3	12.7	15.5	16.5	21.4	24.9
30	10.9	14.9	16.7	19.	22.	24.3
100	15.	17.1	19.2	21.2	23.7	26.7
300	17.5	19.2	21.8	23.7	26.1	28.8
1000	17.9	19.9	21.8	24.7	27.6	30.4
3000	20.	22.9	25.	27.4	30.1	33.3
10000	27.	29.	29.	32.67	36.	37.67

Table 3.4.2: Number of iterations in the worst case.

n/ϵ	0	0.05	0.1	0.15	0.2	0.25
10	10	15	18	21	25	33
30	13	17	18	20	23	26
100	17	19	20	22	24	27
300	19	20	23	27	27	30
1000	21	22	23	29	31	32
3000	24	24	28	30	33	37
10000	29	31	30	34	44	39

Table 3.4.3: Mean value of the number of iterations.

ϵ	0.05	0.1	0.15	0.2	0.25
	54.34	62.67	64.67	68	52.67

Table 3.4.4: Number of iterations in the worst case.

ϵ	0.05	0.1	0.15	0.2	0.25
	62	83	76	74	74

4. Une approche par décomposition des méthodes de points intérieurs pour les problèmes de multiflots

Résumé

Ce chapitre introduit une approche par décomposition d'une méthode de points intérieurs pour la résolution d'un problème de multiflot. Nous présentons d'abord cette approche dans le cadre général de problèmes avec contraintes couplantes. Nous proposons ensuite de spécialiser l'algorithme aux problèmes de multiflot linéaire. Nous exposons cette spécialisation en utilisant la formulation sommets-arcs. Nous nous concentrons ensuite sur la formulation arcs-chemins et nous proposons une méthode de décomposition qui incorpore la méthode de points intérieurs dans la technique de décomposition de Dantzig-Wolfe. Les résultats numériques montrent la supériorité du dernier algorithme. Enfin, nous présentons des résultats numériques obtenus en testant ces algorithmes sur des problèmes fournis par le CNET.

Introduction

Nous nous intéressons au problème de multiflot linéaire de coût minimum PMLCM. Il représente le problème d'optimisation de multiflot le plus souvent étudié [78],[5] et [21].

PMLCM consiste pour un réseau dont les arcs, les nœuds et les capacités sur les arcs sont fixés, à déterminer un acheminement optimal des paquets (de messages, de produits,...) à travers le réseau au sens d'un certain critère de performance.

Le premier ouvrage traitant les problèmes de multiflot dans les réseaux était publié en 1962 par Ford et Fulkerson [37]. Inspiré par leur travail, Dantzig propose dans [29] une méthode de décomposition par les prix. Récemment, Chardaire et Lisser présentent dans [21] des approches variées basées sur des spécialisations de l'algorithme du

simplexe et des méthodes de points intérieurs pour résoudre des problèmes de multiflotts non orientés.

Pour être compétitif dans la résolution de PMLCM, l' algorithme doit exploiter la structure bloc angulaire du problème.

La technique de décomposition de Dantzig-Wolfe exploite cette structure particulière et est considérée comme une approche réussie dans [7] pour les problèmes de multiflotts. Des variantes plus récentes utilisant la décomposition par les prix ont été proposées pour ces problèmes. Nous citons par exemple les travaux de Frangioni [39], de Goffin et al [41], et de Zenios [132], basés respectivement sur la méthode de faisceaux, le centre analytique et les méthodes de pénalité.

La meilleure complexité connue pour la résolution de problème de multiflot est donnée par un algorithme de points intérieurs [74]. Mais jusqu'à maintenant aucune implémentation efficace de cet algorithme n'a été obtenue.

Dans [21], Chardaire et Lisser s'intéressent tout d'abord à la spécialisation de la formulation sommets-arcs. Ils appliquent des méthodes directes (primal simplexe et affine scaling duale) tout en exploitant la structure particulière de la matrice des contraintes. Ces méthodes ne sont pas intéressantes pour résoudre les problèmes de grande taille.

Les auteurs de [21] proposent alors des approches plus puissantes basées sur des méthodes de décomposition spécialisées. Ces dernières résolvent les exemples testés en une fraction du temps de calcul des méthodes directes.

Dans le même esprit nous présentons dans ce chapitre une approche par décomposition d'une méthode de points intérieurs pour résoudre PMLCM.

Notre choix des méthodes de points intérieurs est motivé par leur efficacité dans la résolution des problèmes d'optimisation de grande taille [52, 81]. En fait nous allons nous intéresser particulièrement à la famille des algorithmes de points intérieurs primaux-duaux qui donne les meilleurs résultats.

La tâche principale dans un algorithme de points intérieurs est la résolution des systèmes linéaires nécessaires au calcul des directions de déplacement. Dans [75] Karmarkar et Ramakrishnan introduisent une méthode (*approximate dual projective*) de points intérieurs qui utilise un algorithme de gradient conjugué préconditionné pour résoudre ces systèmes linéaires.

Plusieurs auteurs ont proposé, ensuite différents préconditionnements pour résoudre des problèmes de flotts [114, 75, 116, 73, 95]. Pour une étude plus complète sur les

méthodes de points intérieurs pour la résolution de problèmes de réseau nous nous référons à [120].

Récemment, Castro [20] propose un préconditionnement qui exploite la structure du problème de multiflot linéaire de coût minimum.

Nous avons trouvé des similarités entre notre algorithme de points intérieurs et celui de Castro [20] en terme de schéma de décomposition. L'intérêt de notre algorithme par rapport à celui de Castro réside dans l'utilisation de la formulation arcs-chemins. En effet la matrice de préconditionnement proposée par Castro [20] repose sur la formulation sommets-arcs et ne peut pas être étendue à la formulation arcs-chemins. Nous verrons dans la suite que cette dernière est numériquement plus performante que la formulation sommets-arcs.

L'algorithme de points intérieurs, prédicteur-correcteur, que nous utilisons dans cette étude appartient à la famille des algorithmes de suivi de trajectoire centrale (algorithmes de points intérieurs primaux duaux). Ces algorithmes possèdent à la fois la meilleure estimation de complexité connue à ce jour ($O(\sqrt{n}L)$ itérations) et une convergence quadratique [13, 52]. Nous nous intéressons plus spécifiquement à l'algorithme prédicteur-correcteur de Mizuno, Todd et Ye [102]. Cet algorithme opère à l'intérieur d'un voisinage \mathcal{R} et consiste à alterner les pas de centralisation (de façon à se rapprocher de la trajectoire centrale) et les pas affines (plus grand pas permettant de rester dans \mathcal{R}).

La grande taille des systèmes linéaires associés à ces pas nous amène à utiliser une méthode de *décomposition-coordination*. Nous nous proposons donc de décomposer le système linéaire de calcul de direction de Newton en sous-systèmes qu'on voudrait résolubles séparément et dont la coordination fournit la solution globale. Nous nous intéressons plus particulièrement aux propriétés d'un de ces sous-systèmes que nous appellerons *le système réduit*.

L'intérêt du système réduit réside dans le fait que sa résolution permet de découpler les autres sous-systèmes et de les résoudre en parallèle.

L'étude théorique de cette approche des méthodes de points intérieurs par décomposition va être réalisée dans le cadre général des problèmes avec contraintes couplantes. Nous appliquons ensuite les résultats obtenus dans le cas des deux formulations du problème : sommets-arcs et arcs-chemins. Nous présentons une méthode directe qui exploite la structure de la matrice des contraintes sous formulation sommets-arcs et une méthode de décomposition qui peut être vue comme une incorporation d'une

méthode de points intérieurs dans la technique de décomposition de Dantzig-Wolfe sous formulation arcs-chemins.

Conclusion

Les essais numériques présentés dans ce chapitre sont réalisés dans un environnement MATLAB. Nous constaterons que les problèmes traités avec l'algorithme sous formulation arcs-chemins nécessitent (en moyenne) le quart du temps que l'on met sous la seconde formulation pour des données générées aléatoirement.

Nous avons également testé les algorithmes sur des données fournies par le CNET. Les résultats numériques de ces algorithmes sont prometteurs et montrent la supériorité de l'algorithme sous formulation arcs-chemins.

Rapport INRIA RR-3852/Rapport France Telecom/BD/NT/Cnet/657

soumis à **INFORMS Journal on Computing**

J. Frédéric Bonnans, Mounir Haddou, Abdel Lisser and Raja Rébaï

Interior Point Methods With Decomposition For Multicommodity Flow Problems

Résumé

Cet article introduit une approche par décomposition d'une méthode de points intérieurs pour la résolution d'un problème de multiflot. Nous présentons d'abord cette approche dans le cadre général de problèmes avec contraintes couplantes. Nous proposons ensuite de spécialiser l'algorithme aux problèmes de multiflot linéaire. Nous exposons cette spécialisation en utilisant la formulation nœuds-arcs. Nous nous concentrons ensuite sur la formulation arcs-chemins et nous proposons une méthode de décomposition qui incorpore la méthode de points intérieurs dans la technique de décomposition de Dantzig-Wolfe. Les résultats numériques montrent la supériorité du dernier algorithme. Enfin, nous présentons des résultats numériques obtenus en testant ces algorithmes sur des problèmes fournis par le CNET.

Abstract

This paper introduces an approach by decomposition of an interior point method for solving multicommodity flow problems. First, we present this approach in the general framework of coupling constraints problems. Next, we propose to specialize the algorithm to the linear multicommodity network-flow problems. We expose this specialization using the node-arc formulation. Then, we focus on the arc-path formulation and we propose decomposition method witch incorporates the interior point

method into the Dantzig-Wolfe decomposition technique. The numerical results show the superiority of this last formulation. Finally, we report some numerical results obtained by testing these algorithms with data from the France-Telecom Paris district transmission network.

4.1 Introduction

In this paper, we are concerned with the Least cost Multicommodity Network-flow Problem (LMNP), which is the most studied multicommodity problem. This problem consists in the determination of an optimal routing of the traffic requirements when designing a network.

The first study of multicommodity network-flow problems was conducted by Ford and Fulkerson in 1958 [37]. Their work inspired Dantzig and lead to the price decomposition [29]. Recently, in [21] Chardaire and Lisser present various approaches based on specialization of the simplex algorithm and interior-point methods for solving non-oriented multicommodity flow problems. Another recent work in the context of interior point algorithms for network flow problems, can be found in [119], where it is shown that this kind of method may be very effective.

For being competitive, the algorithm must exploit the structure of multicommodity problem to solve efficiently Newton directions systems.

Most of specialized methods attempt to exploit in some way the block structure of the multicommodity problem. The price directive or Dantzig-Wolfe decomposition is regarded as successful approach in [7] and belongs to the class of cost decomposition approaches for multicommodity flows (see [39], [41] and [132] for recent variants based on bundle methods, analytic centers, and smooth penalty functions, respectively).

The best complexity bound known for multicommodity problems is provided by an interior point algorithm [74], but as yet no efficient implementation has been obtained. The major work in a single iteration of any Interior Point Method (IPM) consists in solving a set of linear equations, the so-called Newton equation system (for details see [8, 13]). All general purpose IPM codes use direct approach [34] to solve the Newton equation system. In [73], Kamath and al. applied a variant of Karmakar's projective algorithm using a preconditioned conjugate gradient (PCG). However, their preconditioner did not take advantage of the multicommodity structure. In [20], Castro exploited this structure and presents a specialization of an interior point algorithm

to multicommodity flow problems. He uses both a preconditioned conjugate gradient solver, and a sparse Cholesky factorization, to solve a linear system of equations at each iteration of the algorithm. Choi and Goldfarb, in [23], presented a decomposition scheme similar to the one in [20]. But they suggest solving a dense-matrix positive definite linear system that appears during the decomposition stage by means of parallel and vector processing. Some similarities were also found between the decomposition scheme presented by Castro in [20] and ours presented in this paper.

The interest of our approach rests in the possibility of solving the arc-path formulation of (LMNP). As will be shown, this formulation is more efficient than the node-arc one.

The linear system to be solved for computing the Newton direction appears, in the case of linear multicommodity network-flow problems, to have the following decoupling property: fixing the value of the coupling variables, we can solve in parallel some equations that coincide with those of the Newton step for the central path associated with the subproblems.

Solving the resulting reduced system allows us to compute the Newton direction, while respecting the above mentioned decomposition principle.

This strategy of computing the Newton direction enables our algorithm to solve efficiently large problems. The interior point algorithm, presented in this paper belongs to the path following interior point algorithms family. These algorithms focused the attention of the community because they both reach the optimal complexity known until now, i.e. convergence within $O(\sqrt{n}L)$ iterations, while converging quadratically [52],[81]. A single iteration of predictor-corrector method needs two solves of the same kind of large, sparse linear system for two different right hand sides. The resolution of these systems represents the main computational burden of the algorithm.

The performance of the predictor-corrector algorithm relies on the efficient solution on those systems.

In the case of linear multicommodity network-flow problems, to have the following decoupling property: fixing the value of the coupling variables, we can solve in parallel some equations that coincide with those of the Newton step for the central path associated with the subproblems.

This paper is organized as follows. Section 2 presents the approach by decomposition of the interior point method in the general framework of coupling constraints

problems. Section 3 describes the specialization of the algorithm to the linear multicommodity network-flow problem. First, we present this specialization using the node-arc formulation. Then, we focus on the arc-path formulation and we propose decomposition method which incorporates the interior point method into the Dantzig-Wolfe decomposition technique. Finally, we report some numerical results in section 4.

Conventions

Given a vector $x \in \mathbb{R}^n$ the relation $x > 0$ is equivalent to $x_i > 0$, $i = 1, 2, \dots, n$, while $x \geq 0$ means $x_i \geq 0$, $i = 1, 2, \dots, n$. We denote $\mathbb{R}_+^n = \{x \in \mathbb{R}^n : x \geq 0\}$ and $\mathbb{R}_{++}^n = \{x \in \mathbb{R}^n : x > 0\}$. Given a vector x , the corresponding upper case symbol denotes as usual the diagonal matrix X defined by the vector. The symbol $\mathbf{1}$ represents the vector of all ones, with dimension given by the context. We denote component-wise operations on vectors by the usual notations for real numbers. Thus, given two vectors u, v of the same dimension, uv , u/v , etc. will denote the vectors with components $u_i v_i$, u_i/v_i , etc. We always formulate an optimization problem as follows:

$$\begin{cases} \text{Min}_X & f(X); \\ \text{s.t.} & g(x) \leq 0, \quad (\Lambda) \end{cases} \quad (4.1.1)$$

where Λ denotes the Lagrange multipliers associated with the constraint.

We denote the null space and range space of a matrix A by $\mathcal{N}(A)$ and $\mathcal{R}(A)$ respectively.

4.2 Linear Optimization problems with coupling constraints

Linear coupling constraints problems usually have a very large number of variables and constraints and arise in a great variety of applications [30, 29]. This section is concerned with the resolution of these problems using an interior point method, the predictor-corrector method.

4.2.1 Problem formulation

We consider the so called linear optimization problem with coupling constraints:

$$\left\{ \begin{array}{l} \text{Min}_{x^0, x^1, \dots, x^K} \sum_{k=0}^K (c^k)^T x^k; \\ \sum_{k=0}^K B^k x^k = b^0, \quad (\lambda^0) \\ A^k x^k = b^k, k = 1, \dots, K, \quad (\lambda^k) \\ x^k \geq 0, k = 0, \dots, K, \quad (s^k) \end{array} \right. \quad (4.2.1)$$

where

- for all $k \in \{0, \dots, K\}$:
 - x^k , c^k and s^k (dual variables of x^k) are elements of \mathbb{R}^{n^k} .
 - b^k and λ^k (dual variable of the k^{th} constraint) are elements of \mathbb{R}^{p^k} .
 - and B^k is a $p^0 \times n^k$ matrix.
- for all $k \in \{1, \dots, K\}$: A^k is a $p^k \times n^k$ matrix.

Without the first constraints problem (4.2.1) would be separable. Indeed, each x^k is in this case solution of the linear problem

$$\text{Min}_{z \in \mathbb{R}^{n^k}} (c^k)^T z ; A^k z = b^k ; z \geq 0. \quad (4.2.2)$$

We assume that A^1, \dots, A^K , $(B^0 \ B^1 \ \dots \ B^K)$, $B^1/\mathcal{N}(A^1)$, \dots , $B^K/\mathcal{N}(A^K)$ and B^0 have full row rank. One recovers the standard form of linear optimization by setting

$$\hat{x} := \begin{pmatrix} x^0 \\ x^1 \\ \vdots \\ x^K \end{pmatrix} \in \mathbb{R}^{n^t}, \quad \hat{c} := \begin{pmatrix} c^0 \\ c^1 \\ \vdots \\ c^K \end{pmatrix} \in \mathbb{R}^{n^t}, \quad \hat{b} := \begin{pmatrix} b^0 \\ b^1 \\ \vdots \\ b^K \end{pmatrix} \in \mathbb{R}^{p^t},$$

$$\text{and the } p^t \times n^t \text{ matrix } \hat{A} = \begin{pmatrix} B^0 & B^1 & \dots & B^K \\ & A^1 & & \\ & & \ddots & O \\ & O & & A^K \end{pmatrix},$$

where $n^t := \sum_{k=0}^K n^k$ and $p^t := \sum_{k=0}^K p^k$. Then (4.2.1) is equivalent to the linear optimization problem in standard form

$$\begin{cases} \text{Min}_{\hat{x}} \hat{c}^T \hat{x}; \\ \hat{A} \hat{x} = \hat{b}, & (\hat{\lambda}) \\ \hat{x} \geq 0, & (\hat{s}) \end{cases} \quad (4.2.3)$$

The Lagrange multipliers vector $\hat{\lambda}$ ($\in \mathbb{R}^{p^t}$) and \hat{s} ($\in \mathbb{R}^{n^t}$) can be expressed with Lagrange multipliers of problem (4.2.1). Indeed, we have

$$\hat{s} := \begin{pmatrix} s^0 \\ s^1 \\ \vdots \\ s^K \end{pmatrix} \quad \text{and} \quad \hat{\lambda} := \begin{pmatrix} \lambda^0 \\ \lambda^1 \\ \vdots \\ \lambda^K \end{pmatrix}.$$

The first bloc of the matrix \hat{A} corresponds to the coupling constraints. Without these constraints, the program (6.2.6) would have a diagonal structure of which we might take advantage.

4.2.2 solving linear optimization problems with coupling constraints using a predictor-corrector algorithm

We are concerned with the most popular primal-dual logarithmic barrier method called predictor-corrector method. This name was first used by Mizuno, Todd and Ye in [102], where they proposed for the first time the method considered in this paper for the small neighborhood case. Like them we alternate (single) primal-dual affine-scaling step and (single) primal-dual centering step but using large neighborhoods of the central path [16]. We choose to use this method because of its good theoretical and practical proprieties. Indeed, the predictor-corrector methods converges with an $O(nL)$ iterations bound (when large neighborhoods are used) and has asymptotically a quadratic convergence (see .e.g., Mehrotra [93, 94], Ye et al [131], Gonzaga and Tapia [53, 54], Ye [130] and Luo and Ye [89]). For more details about primal-dual predictor-corrector methods, we refer the reader to Bonnans et al [13], Roos et al [123], Wright [128].

Interior point methods follow the so-called center path (approximately) as a guideline to the optimal set. Subsequently we exploit the structure of the central path equation to obtain a reduced equation. Then, we deduce a decomposition method for computing the Newton directions.

Predictor-corrector algorithm

Predictor-corrector algorithms follow the central path for solving (4.2.3). The equations of the central path associated with the linear optimization problem in its standard form (4.2.3) are

$$\begin{cases} \hat{x}\hat{s} & = \mu\mathbf{1}, \\ \hat{A}\hat{x} & = \hat{b}, \\ c + \hat{A}^T\hat{\lambda} & = \hat{s}, \end{cases} \quad (4.2.4)$$

where $(\hat{x}, \hat{s}, \hat{\lambda})$ belongs to $\mathbb{R}_{++}^{n^t} \times \mathbb{R}_{++}^{n^t} \times \mathbb{R}^{p^t}$.

At each iteration, the algorithm compute an affine and centering Newton directions of the following form:

$$\begin{cases} \hat{s}\hat{u} + \hat{x}\hat{v} & = \hat{f}, \\ \hat{A}\hat{u} & = \mathbf{0}, \\ \hat{A}^T\hat{\delta} & = \hat{v}, \end{cases} \quad (4.2.5)$$

where $(\hat{u}, \hat{v}, \hat{\delta})$ denotes the step associated with $(\hat{x}, \hat{s}, \hat{\lambda})$, and the right hand side \hat{f} depends on the algorithm.

We state in the following a predictor corrector algorithm in large neighborhoods (**PCL**), in which the set \mathcal{N}_ν denotes a large neighborhood defined by:

$\mathcal{N}_\nu :=$

$$\{(\hat{x}, \hat{s}, \hat{\lambda}, \mu) \in \mathbb{R}_{++}^{n^t} \times \mathbb{R}_{++}^{n^t} \times \mathbb{R}^{p^t} \times \mathbb{R}_{++}; \hat{A}\hat{x} = \hat{b}, c + \hat{A}^T\hat{\lambda} = \hat{s} \text{ and } \nu\mathbf{1} \leq \frac{\hat{x}\hat{s}}{\mu} \leq \nu^{-1}\mathbf{1}, \mu \leq \mu_0\}.$$

The algorithm is as follows

Algorithm PCL

Data: $\mu_\infty > 0$, $\nu \in]0, 1/2]$, $(\hat{x}^0, \hat{s}^0, \hat{s}, \mu_0) \in \mathcal{N}_\nu$. $j := 0$

REPEAT

- $\hat{x} := \hat{x}^j$, $\hat{s} := \hat{s}^j$, $\mu := \mu_j$;
- Corrector step: Compute (\hat{u}^c, \hat{v}^c) solution of (4.2.5) with $\hat{f} = -\hat{x}\hat{s} + \mu\mathbf{1}$.
 $\hat{x}(\theta) := \hat{x} + \theta\hat{u}^c$, $\hat{s}(\theta) := \hat{s} + \theta\hat{v}^c$.
 Compute $\theta^c \in]0, 1]$ such that $(\hat{x}(\theta^c), \hat{s}(\theta^c), \mu) \in \mathcal{N}_\nu$.
 $\hat{x} := \hat{x}(\theta^c)$, $\hat{s} := \hat{s}(\theta^c)$.
- Predictor step: Set $\hat{f} = -\hat{x}\hat{s}$. Compute (\hat{u}^a, \hat{v}^a) solution of (4.2.5).
 $\hat{x}(\theta) := \hat{x} + \theta\hat{u}^a$, $\hat{s}(\theta) := \hat{s} + \theta\hat{v}^a$, $\mu(\theta) := (1 - \theta)\mu$.

Compute θ^a , the largest value in $]0, 1[$ such that

$$(\hat{x}(\theta), \hat{s}(\theta), \mu(\theta)) \in \mathcal{N}_\nu, \forall \theta \leq \theta^a.$$

$$\hat{x}^{j+1} := \hat{x}(\theta^a), \hat{s}^{j+1} := \hat{s}(\theta^a), \mu_{j+1} := (1 - \theta^a)\mu_j.$$

- $j := j + 1$.

UNTIL $\mu_j < \mu_\infty$.

We choose the step-size for the centering displacement is as follows: starting from a unit step, we divide the step by two until the new point belongs to the large neighborhood. This is a rather rough linear search. Because it proved to be efficient, we content of it. The algorithm finds a point such that $\mu_k \leq \mu_\infty$ in at most $O(nL)$ iterations [16].

The central path

We rewrite the system (4.2.4) as follows:

$$\begin{cases} x^0 s^0 & = \mu \mathbf{1}, \\ \sum_{k=0}^K B^k x^k & = b^0, \\ c^0 + (B^0)^T \lambda^0 & = s^0, \end{cases} \quad (4.2.6)$$

and

$$k \in \{1, \dots, K\} \begin{cases} x^k s^k & = \mu \mathbf{1}, \\ A^k x^k & = b^k, \\ c^k + (B^k)^T \lambda^0 + (A^k)^T \lambda^k & = s^k. \end{cases} \quad (4.2.7)$$

With a value of λ^0 (for which we assume that each problem (4.2.7) has an interior feasible solution realizing (4.2.6) is associated some uniquely defined $(x^k(\lambda^0), s^k(\lambda^0), \lambda^k(\lambda^0))$, $k = 1, \dots, K$, solution of the local problem (4.2.7). This local problem may be interpreted as the equation of the central path associated with the local problem (4.2.2) where the unitary cost vector is $c^k + (A^k)^T \lambda^0$.

We may write the central path equation in the *reduced form*:

$$\begin{cases} x^0 s^0 & = \mu \mathbf{1}, \\ \sum_{i=0}^K B^i x^i(\lambda^0) & = b^0, \\ c^0 + B^0 \lambda^0 & = s^0. \end{cases} \quad (4.2.8)$$

Predictor-corrector algorithm follows the central path using Newton directions on the equation of the central path. We apply now similar considerations for computing the Newton step.

4.2.3 The Newton Step

The system (4.2.5) may be written as

$$\left\{ \begin{array}{l} s^0 u^0 + x^0 v^0 \\ \sum_{k=0}^K B^k u^k \\ (B^0)^T \delta^0 \end{array} \right. \begin{array}{l} = f^0, \\ = 0, \\ = v^0, \end{array} \quad (4.2.9)$$

$$k \in \{1, \dots, K\} \left\{ \begin{array}{l} s^k u^k + x^k v^k \\ A^k u^k \\ (A^k)^T \delta^k + (B^k)^T \delta^0 \end{array} \right. \begin{array}{l} = f^k, \\ = 0, \\ = v^k. \end{array}$$

We can solve this linear system as follows. For a given value of δ^0 we can compute in parallel the values of (u^k, v^k, δ^k) , $k = 1, \dots, K$ that satisfy the three last equations. Let us call $u^k(\delta^0)$, $v^k(\delta^0)$, $\delta^k(\delta^0)$ the solution of the k^{th} local problem (4.2.7).

Note that $u^k(\delta^0)$, $v^k(\delta^0)$ and $\delta^k(\delta^0)$ are affine functions of δ^0 .

Computing the Newton step is then equivalent to solving the reduced linear system

$$\left\{ \begin{array}{l} s^0 u^0 + x^0 v^0 \\ \sum_{k=0}^K B^k u^k(\delta^0) \\ (B^0)^T \delta^0 \end{array} \right. \begin{array}{l} = f^0, \\ = 0, \\ = v^0. \end{array} \quad (4.2.10)$$

which can be interpreted as the Newton step equation of (4.2.6) in the feasible case.

Once δ^0 is computed, the resolution of the K systems

$$\left\{ \begin{array}{l} \text{(i)} \quad s^k u^k + x^k v^k \\ \text{(ii)} \quad A^k u^k \\ \text{(iii)} \quad (A^k)^T \delta^k + (B^k)^T \delta^0 \end{array} \right. \begin{array}{l} = f^k, \\ = 0, \\ = v^k, \end{array} \quad k = 1, \dots, K, \quad (4.2.11)$$

may be done in parallel. For each k , this is equivalent to the computation of the Newton direction associated to a single commodity minimum network flow problem.

4.2.4 Resolution of the reduced system

Solving the reduced system (4.2.10), the algorithm gives explicitly the functions $u^k(\delta^0)$.

For each $k \in \{1, \dots, K\}$, we consider the system (4.2.11) and we define in a classical way the scaled variables and operators :

$$\begin{aligned} \bar{f}^k &:= f^k / (\sqrt{x^k s^k}), & d^k &:= \sqrt{x^k / s^k}, & \bar{u}^k &:= (d^k)^{-1} u^k, \\ \bar{v}^k &:= d^k v^k, & \bar{A}^k &:= A^k D^k, & \text{and } \bar{B}^k &:= B^k D^k. \end{aligned}$$

After scaling system (4.2.11), we obtain the equivalent relation :

$$k \in \{1, \dots, K\} \left\{ \begin{array}{l} \text{(i) } \bar{u}^k + \bar{v}^k = \bar{f}^k, \\ \text{(ii) } \bar{A}^k \bar{u}^k = 0, \\ \text{(iii) } (\bar{A}^k)^T \delta^k + (\bar{B}^k)^T \delta^0 = \bar{v}^k. \end{array} \right. \quad (4.2.12)$$

Multiplying equation (iii.4.2.12) by \bar{A}^k on the left, and using $\bar{A}^k \bar{v}^k = \bar{A}^k \bar{f}^k$, we may express δ^k as a function of δ^0 :

$$\bar{A}^k (\bar{A}^k)^T \delta^k = -\bar{A}^k (\bar{B}^k)^T \delta^0 + \bar{A}^k \bar{f}^k. \quad (4.2.13)$$

$\bar{A}^k (\bar{A}^k)^T$ is invertible because A^k has full row rank.

We associate with each $k \in \{1, \dots, K\}$, the p^k order invertible symmetric matrix N^k defined by: $N^k = \bar{A}^k (\bar{A}^k)^T$.

We introduce also for each k the $p^k \times p^0$ matrix M_δ^k and the \mathbb{R}^{p^k} -vector h_δ^k defined respectively by:

$$M_\delta^k = -(N^k)^{-1} \bar{A}^k (\bar{B}^k)^T \text{ and } h_\delta^k = (N^k)^{-1} \bar{A}^k \bar{f}^k. \quad (4.2.14)$$

We have by (4.2.13) and (4.2.14):

$$\delta^k = M_\delta^k \delta^0 + h_\delta^k. \quad (4.2.15)$$

Combining with equations (iii.4.2.12) and (4.2.15), we obtain that:

$$\bar{v}^k = M_v^k \delta^0 + h_v^k, \quad (4.2.16)$$

where

- M_v^k is a $n^k \times p^0$ matrix defined by: $M_v^k = (\bar{B}^k)^T + (\bar{A}^k)^T M_\delta^k$.
- h_v^k is a \mathbb{R}^{n^k} -vector defined by: $h_v^k = (\bar{A}^k)^T h_\delta^k$.

From the equations (4.2.16) and (i.4.2.12), we deduce:

$$u^k = M^k \delta^0 + h^k, \quad (4.2.17)$$

where

- M^k is a $n^k \times p^0$ matrix defined by: $M^k = -D^k M_v^k$.
- h_u^k is a \mathbb{R}^{n^k} -vector defined by: $h^k = D^k(\bar{f}^k - h_v^k)$.

We also express u^0 as a function of δ^0 by using the first and third equations of the system (4.2.10). We obtain then:

$$u^0 = M^0 \delta^0 + h^0, \quad (4.2.18)$$

where

- M^0 is a $n^0 \times p^0$ matrix defined by: $M^0 = -D^0(B^0)^T$.
- h^0 is a \mathbb{R}^{n^0} -vector defined by: $h^0 = f^0/s^0$.

From (4.2.17) and (4.2.18), the reduced system (4.2.10) gives the following p -order linear system:

$$M \delta^0 = h, \quad (4.2.19)$$

where

- M is a p^0 order invertible matrix defined by: $M = \sum_{k=0}^K B^k M^k$.
- h is a \mathbb{R}^{p^0} -vector defined by: $h = -\sum_{k=0}^K B^k h^k$.

Solving the reduced system (4.2.10) amounts to solving system (4.2.19).

Remarque 4.2.1. At every iteration of the predictor-corrector algorithm, two different systems of equations (4.2.19) have to be solved by computing explicitly the matrices $M^k, k = 0, \dots, K$.

It is possible to solve the reduced system (4.2.10) by an iterative algorithm without expliciting those matrixes. In this case we obtain an approximate solution of the linear system (4.2.19).

In [16], the authors show that the infeasible predictor-corrector algorithm in large neighborhoods remains fast under a relative perturbation of the right-hand-side of

the order of 25%. That proves the robustness of the algorithm when the Newton direction is computed approximately.

If we look more into the details of the matrix M , we check upon the following proprieties:

Lemme 14. The matrix M is symmetric negative definite.

Preuve . :

Without loss of generality, let us suppose that $f^k = 0$ and $x^k = s^k = \mathbf{1}$, for $k = 0, \dots, K$. Indeed, we are looking for matrix proprieties and neither the right hand side term nor the positive diagonal scaling matrix has a consequence on these proprieties. Using these hypothesis and system (4.2.9), obtain

$$\begin{cases} B^0 M^0 &= -B^0 (B^0)^T, \\ B^k M^k &= -B^k [I(p_k) - (A^k)^T (A^k (A^k)^T)^{-1} A^k] (B^k)^T, k = 1, \dots, K. \end{cases} \quad (4.2.20)$$

The system (4.2.20) proves that each $B^i M^i$ is symmetric for $i \in \{0, 1, \dots, K\}$.

Since $M = \sum_{k=0}^K B^k M^k$ (4.2.19), M is also symmetric.

Let us check that M is negative definite.

We recall that B^0 has full row rank. Hence $B^0 M^0$ is negative definite.

To complete the proof, it suffice to show that for each $k \in \{1, \dots, K\}$, $[I(p_k) - (A^k)^T (A^k (A^k)^T)^{-1} A^k]$ is positive semi-definite.

For $k \in \{1, \dots, K\}$ and all $y^k \in \mathbb{R}^{p_k}$, we have $y^{kT} [I(p_k) - (A^k)^T (A^k (A^k)^T)^{-1} A^k] y^k = \|\tilde{y}^k\|^2$ where \tilde{y}^k is the projection of y^k on $\mathcal{N}(A^k)$.

The conclusion follows. □

Remarque 4.2.2. Thanks to the good proprieties of the linear system (4.2.19) one can use the conjugate gradient method for solving the reduced system as an iterative method. In this study, we propose a generic predictor-corrector algorithm for solving linear optimization problems with coupling constraints. For more details about using preconditioned conjugate gradient to solve an example of those problem, we refer the reader to [20]. The reduced system matrix M is known as the Schur complement [20]. Choi and Goldfarb in [23] and also Castro in [20] state that the Schur complement becomes completely dense. In order to circumvent this difficulty, Castro propose to solve the reduced system through a preconditioned conjugate gradient.

The preconditioner that he proposed consist of using the inverse of M . However in the specialization proposed in [20], Castro takes advantage only of node-arc linear multicommodity problem structure.

We describe below only one Newton step of the predictor corrector algorithm PCL.

Algorithm PCLMNP

- for $k \in \{1, \dots, K\}$: Compute the matrix M^k and the vector h^k , using (4.2.15),(4.2.16) and (4.2.17):

- $M^k := -D^k(\overline{B}^k)^T + (\overline{A}^k)^T(\overline{A}^k(\overline{A}^k)^T)^{-1}\overline{A}^k(\overline{B}^k)^T$

- $h^k := D^k(\overline{f}^k - (\overline{A}^k)^T(\overline{A}^k(\overline{A}^k)^T)^{-1}\overline{A}^k\overline{f}^k)$.

- Compute the matrix M^0 and the vector h^0 , using (4.2.18)

- $M^0 := -D^0(B^0)^T$.

- $h^0 := f^0/s^0$.

- Solve the reduced system (4.2.19).

Solve the linear system $M\delta^0 = h$, where

- $M := \sum_{k=0}^K B^k M^k$,

- $h := -\sum_{k=0}^K B^k h^k$.

$$v^0 := \delta^0 \text{ and } u^0 := (f^0 - x^0\delta^0)/s^0.$$

- for $k = 1, \dots, K$

- compute (u^k, v^k, δ^k) solution of the system (4.2.11).

4.3 Linear multicommodity network-flow problem

The Multicommodity network problems appear when different commodities share a common network. These problems are generally very difficult to solve, not because of

their large scale, but also because it is not easy to handle the conflicts between several commodities on the same network [22]. There are numerous models in applied optimization that involve multicommodity systems [9], [41], [86], [87]. The Linear multicommodity network-flow problem is the most often studied problem in multicommodity optimization [78],[5] and [21].

This problem consists of the determination of the most economical way of using the available transmission capacities in order to route a traffic matrix through the network.

The best complexity bound known for multicommodity problems is provided by an interior point algorithm [74], though, as yet, no efficient implementation had been obtained. A variant of Karmarkar's projective algorithm is applied in [73], using a preconditioned conjugate gradient (PCG) solver. Contrarily to the preconditioner presented in [73] that did not take advantage of the multicommodity structure, the one proposed in [23] exploit this structure. Recently Castro presented in [20] a specialization of an interior point algorithm to multicommodity flow problems. He uses both a preconditioned conjugate gradient solver, and a sparse Cholesky factorization, to solve a linear system of equations at each iteration of the algorithm.

4.3.1 Framework

Let $G(V, E)$ be a directed graph, where V is a set of p nodes and E is a set of n arcs, and let K be a set of K commodities to be routed through the network represented by G .

Each commodity k , for $k \in K$, has a unique source o^k and a unique sink p^k . We denote by $A \in \mathbb{R}^{p \times n}$ the node-arc incidence matrix of G . Each column of A is related to an arc $e \in E$ and has only two nonzero $(1, -1)$ coefficients in those rows associated with (respectively) the origin and the destination nodes of e . We shall consider that the arcs of the network have a capacity b^0 ($\in \mathbb{R}^n$) for all commodities. We denote by $b^k \in \mathbb{R}^p$, for $k \in K$, a vector of supplies/demands for commodity k at the nodes of the network defined by:

$$b_i^k = \begin{cases} +r^k & \text{if } i = o^k, \\ -r^k & \text{if } i = p^k, \\ 0 & \text{elsewhere,} \end{cases} \quad (4.3.1)$$

where r^k is the value of the flow of the k^{th} commodity. Let us assume that an unitary cost vector $c = (c_e)_{e \in E}$ is given and that this cost does not depend on commodity. We shall distinguish two models of (LMNP): the node-arc model and the arc-path one.

4.3.2 Node-Arc Formulation

The node-arc formulation of (LMNP) can be expressed as follows:

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K} c^T \sum_{k=1}^K x^k; \\ (i) \ x^0 + \sum_{k=1}^K x^k = b^0, \\ (ii) \ Ax^k = b^k, \ k = 1, \dots, K, \\ (iii) \ x^k \geq 0, \ k = 0, \dots, K, \end{array} \right. \quad (4.3.2)$$

where

- x^k is a \mathbb{R}^n -vector, x_e^k is the flow carried by commodity k on arc e , ($e \in E$),
- x^0 is the residual capacity vector ($x^0 \in \mathbb{R}_+^n$).

In the node-arc formulation, the decision variables are the flow of commodities on each arc. This formulation associates to each commodity k flow conservation (4.3.2.ii) and non-negativity (4.3.2.iii) constraints. The equations (4.3.2.i) refer to the capacity constraints. The node-arc formulation of (LMNP) (4.3.2) has $\hat{p} = K \times p + n$ constraints and $\hat{n} = (K+1) \times n$ variables. For real networks such as the Paris district transmission network, the number of nodes can be larger than 100 and the number of arc larger than 300. If one assume that this network has 1000 commodities, the size of the problem can be larger than 100300 constraints and 300300 variables. So, the linear problem (4.3.2) is very large even for a small network. We propose to specialize the method developed in the previous section to this problem. Let us note that the system (4.3.2) corresponds to the system (4.2.1).

Indeed, in the case of the node-arc formulation of the linear multicommodity network-flow problem, we have:

- for all $i \in \{0, \dots, K\}$: $n^i := n$ and $B^i := I(n)$.
- for all $i \in \{1, \dots, K\}$: $A^i := A$, $p^i := p$ and $c^i = c$.

- $p^0 := n$ and $c^0 := 0(n, 1)$.

We refer to this specialization of predictor-corrector algorithm to node-arc formulation of (LMNP) by the NAF algorithm.

4.3.3 Arc-path Formulation

The linear multicommodity network problem can be formulated in term of paths. The decision variables become the flow components on paths. We denote by I^k the set of distinct elementary paths between the source node o_k and the sink node p_k in G . We designate by n^k the cardinal of I^k ($n^k = |I^k|$). An element i of the set I^k is characterized by a \mathbb{R}^n -boolean vector $\{\pi_i^k\}$ satisfying:

$$(\pi_i^k)_e = \begin{cases} 1 & \text{if } e \text{ belongs to } i, \\ 0 & \text{otherwise.} \end{cases} \quad (4.3.3)$$

The variables x^1, \dots, x^K , are now associated to elementary paths rather than to arcs. We denote by x_i^k the component of the commodity k carried by the path characterized by π_i^k . The sum over i of the components x_i^k is the value of the commodity k : $\sum_{i \in I^k} x_i^k = r^k$. The total flow of the commodity k is the \mathbb{R}^n -vector $\sum_{i \in I^k} x_i^k \pi_i^k \in \mathbb{R}^n$. We recall that the \mathbb{R}^n -vector x^0 is the residual capacity vector.

The arc-path formulation of the (LMNP) consist of the following program:

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K} c^T \sum_{k=1}^K \pi^k x^k; \\ (i) \ x^0 + \sum_{k=1}^K \pi^k x^k = b^0, \quad (\lambda^0) \\ k \in \{1, \dots, K\} \left\{ \begin{array}{l} (ii) \ \omega^k x^k = r^k, \quad (\lambda^k) \\ (iii) \ x_i^k \geq 0, i \in I^k, \quad (s_i^k) \end{array} \right. \\ (iv) \ x^0 \geq 0, \quad (s^0) \end{array} \right. \quad (4.3.4)$$

where for all $k \in K$, ω^k is the \mathbb{R}^{n^k} row-vector defined by $\omega_i^k = 1$, for $i = 1, \dots, n^k$. We can recognize the same type of constraints as in the node-arc formulation. Indeed, the equations (4.3.4.ii) and (4.3.4.i) represent respectively the flow conservation constraints associated to each commodity and the capacity constraints.

The general formulation (4.2.1), is specialized for the (LMNP) arc-path formulation (4.3.4) by setting:

- for all $k \in \{1, \dots, K\}$: $c^k := (\pi^k)^T c$, ($c^k \in \mathbb{R}^{n^k}$), $B^k = \pi^k$, $b^k = r^k$, $A^k = \omega^k$ and $p^k = 1$.
- $B^0 = I(n)$ and $p^0 = n$.

Then we can, as in the case of the node-arc formulation, adapt the algorithm described in the general framework of linear optimization with coupling constraints problems to the arc-path formulation of LMNP. We refer to this adaptation by APF. The arc-path formulation associates to each elementary path a variable. The number of variables may then increase exponentially with the size of the graph. This constitutes the main shortcoming for this formulation. However, in practice, an optimal solution is carried by a relatively small number of paths. Indeed the formulation (4.3.4) has only $n + K$ constraints. This implies that (if (4.3.4) has a solution) an optimal solution that has at most $n + K$ strictly nonzero variables exists. Then no more than $n + k$ different paths are needed to satisfy all the requested flow.

We propose to take advantage of this property by limiting the routing problem LMNP to a subset of elementary paths generated by an iterative process. Indeed, the path matrixes π^k are only known implicitly. However, the form of such columns is known, and they can be generated as needed during the course of the algorithm- hence the name column generation [36, 71]. The scheme of column generation algorithm requires one to solve successive linear programs of smaller size. We use the algorithm PCLMNP for solving these programs. The algorithm APF does not take into account all the sets I^k . It starts with a subset I_0^k and adds at each iteration j , no more than one path $\chi^{k,j}$ to the subset I_j^k . In other words APF introduce at each iteration j , no more than one column to the matrix π^k . By reducing the number of the columns, we reduce also the number of variables.

We refer to the stage of paths (columns) generation as oracles. The Master Program consist of the resolution of a routing problem using a limited number of paths. Indeed, at each iteration j a commodity k can be carried only by paths belonging to a subset I_j^k .

At each outer iteration j , algorithm APF solves the following master program:

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K} c^T \sum_{k=1}^K \pi^{kj} x^k; \\ \text{(i) } x^0 + \sum_{k=1}^K \pi^{kj} x^k = b^0, \\ k \in \{1, \dots, K\} \left\{ \begin{array}{l} \text{(ii) } \omega^{kj} x^k = r^k, \\ \text{(iii) } x_i^k \geq 0, i \in I^{jk}, \end{array} \right. \\ \text{(iv) } x^0 \geq 0, \end{array} \right. \quad (4.3.5)$$

where

- $n^{kj} = |I_j^k|$,
- $\omega^{kj} \in \mathbb{R}^{n^{kj}}$ and $\omega_i^{kj} = 1, i = 1, \dots, n^{kj}$,
- $x^k \in \mathbb{R}^{n^{kj}}$,
- π^{kj} is the matrix whose columns are $\pi_i^k, i \in I_j^k$.

The optimality system associated to this problem is:

$$\left\{ \begin{array}{l} x^0 s^0 = 0, \\ x^0 + \sum_{k=1}^K \pi^{kj} x^k = b^0, \\ \lambda^0 = s^0, \\ k \in \{1, \dots, K\} \left\{ \begin{array}{l} x^k s^k = 0, \\ \omega^{kj} x^k = r^k, \\ (c^k + \lambda^0)^T \pi^{kj} + \omega^{kj} \lambda^k = s^k, \end{array} \right. \\ x^k, s^k \geq 0, \quad k = 0, \dots, K, \end{array} \right. \quad (4.3.6)$$

We denote by (x^j, s^j, λ^j) the solution of problem (4.3.5) given by predictor-corrector algorithm. It satisfies the optimality system (4.3.6). We introduce $(\tilde{x}, \tilde{s}, \tilde{\lambda})$ defined by :

$$\left\{ \begin{array}{l} \tilde{x}^0 = x^{0j}, \\ \tilde{s}^0 = s^{0j}, \\ \tilde{\lambda}^0 = \lambda^{0j}, \\ \text{for } k \in \{1, \dots, K\} \left\{ \begin{array}{l} \tilde{x}_i^k = \begin{cases} x_i^k & \text{if } i \in I_j^k, \\ 0 & \text{else.} \end{cases} \\ \tilde{s}_i^k = (c^k + \lambda^{0j})^T \pi_i^k + \lambda^{kj}, \quad i \in I^k. \\ \tilde{\lambda}^k = \lambda^{kj}. \end{array} \right. \end{array} \right. \quad (4.3.7)$$

For all commodity $k \in K$, we designate by $\chi^{k,j}$ a \mathbb{R}^n -boolean vector characterizing a shortest path between o_k and p_k calculated with $c^k + \lambda^{0j}$ as unitary cost vector. We have the following classical result:

Lemme 15. the following two statements are equivalent

- (i) the point $(\tilde{x}, \tilde{s}, \tilde{\lambda})$ defined by (4.3.7) is solution of (LMNP)
- (ii) for all commodity k in $\{1, \dots, K\}$ we have:

$$(c^k + \lambda^{0,j})^T \pi_j^k + \lambda^k \geq 0 \quad (4.3.8)$$

Preuve. Let us first check that (i) imply (ii):

If $(\tilde{x}, \tilde{s}, \tilde{\lambda})$ is a solution of (4.3.4), then it satisfies the associated optimality system.

We obtain this optimality system, by setting $I_j^k = I^k$ in (4.3.6).

The dual constraint $(c^k + \lambda^0)^T \pi_i^k + \lambda^k \geq 0$ is satisfied for all paths in I^k . In particular it remains true for $\chi^{k,j}$.

Let us check now that (ii) leads to (i):

For all paths in I^k we have $(c^k + \lambda^0)^T \pi_i^k \geq (c^k + \lambda^0)^T \chi^{k,j}$.

If $(c^k + \lambda^{0j})^T \chi^{k,j} + \lambda^{k,j} \geq 0$ then $(c^k + \lambda^{0j})^T \pi_i^k + \lambda^{k,j} \geq 0$ for all $i \in I^k$.

Then $(\tilde{x}, \tilde{s}, \tilde{\lambda})$ verifies the optimality system associated to (LMNP).

≡

Management of the sets I^k

At each outer iteration j and for all commodity k in $\{1, \dots, K\}$, the algorithm APF computes a shortest path $\chi^{k,j}$. When the dual variable associated to this path is negative ($(c^k + \lambda^{0j})^T \chi^{k,j} + \lambda^{k,j} < 0$), the algorithm adds the path $\chi^{k,j}$ to the set I_j^k :

$$I_{j+1}^k = I_j^k + \chi^{k,j}.$$

Now we are ready to state the (APF) algorithm.

Algorithme 1. • **Initialisation:** $j = 0$, for $k = 1, \dots, K$:

- Solve shortest path problems: I_0^k .
- If $\sum_{k=1}^K \pi^{k,0} r^k > b^0$ then add *artificial vector capacity* and use *Big M* method.

- **Master program:** Solve an (LMNP) limited to subsets I_j^1, \dots, I_j^K by predictor corrector algorithm PCLMNP.

Deduce $(\tilde{x}, \tilde{s}, \tilde{\lambda})$, (4.3.7).

- **Oracle:** for $k = 1, \dots, K$:
Solve shortest path problem: Compute $\chi^{k,j}$ shortest path between o_k and p_k calculated with $c^k + \lambda^{0j}$.
- **Stop test:** $test = 0$,
 - for $k = 1, \dots, K$: if $(c^k + \lambda^{0j})^T \chi^{k,j} + \lambda^{kj} < 0$ then $I_{j+1}^k = I_j^k + \chi^{k,j}$ and $test = test + 1$.
 - if $test > 0$ then $j = j + 1$, go to 2, else Stop.

4.4 Numerical Results

In this paper, the numerical tests were performed in MATLAB environment. For this reason, the computing time is not significant.

4.4.1 Randomly generated Problems

Performance of algorithm NAF

We consider networks with $n = 60$ arcs and $p = 30$ nodes randomly generated. The number of commodities varies up to 1000. We denote by K , i_a and i_w , respectively, the number of commodities, the average value of the number of iterations and the number of iterations in the worst case.

For each value of $K \in \{10, 50, 100, 500, 1000\}$, we generate randomly 10 networks.

The generation of a problem consists of the generation of node-arc incidence matrix of the graph $G = (V, E)$.

The starting point is $x^0 = s^0 = \mathbf{1}$ and $\mu_0 = 1$. The algorithm stops when $\mu < 10^{-10}$.

The numerical results obtained by NAF are summarized in the table 4.4.1.

Comparison between NAF and APF performances

We generate randomly 5 LMNP. We test the performances of the algorithms for the case $p = 30$, $n = 70$, and $K = 25$.

We obtain the following table: where

K	i_a	i_w
10	18.8	20
50	22.7	25
100	24.2	26
500	29	32
1000	35.20	39

Table 4.4.1: Performance of the algorithm NAF on Networks with 30 nodes and 60 arcs

K	t1	t2	i1	i2
pb1	376	80	42	15
pb2	348	63	40	12
pb3	324	80	37	15
pb4	314	82	36	15
pb5	305	82	37	15

Table 4.4.2: Comparison between NAF and APF performances

- $t1$ is the computing time with (NAF).
- $t2$ is the computing time with (APF).
- $i1$ is the number of iterations with (NAF).
- $i2$ is the number of iterations with (APF).

We establish that the algorithm APF needs (in average) the one fourth of the computing time required by NAF.

4.4.2 Tests With Real Data

We report in this section some numerical results obtained by testing these algorithms with instances using real data from Paris district area transmission network. Since the information is classified, we are not allowed to give details about the data. We can only give the size of the problems in their standard form (see Table 4.4.3).

Our implementation of the algorithm APF uses the “Big M” method. The capacity

of the generated paths may not be sufficient to meet all demands and one must add a supplement capacity with a very large cost (M) to assure the routing of the demands. Test NOE26 is particularly interesting. It is a highly degenerate problem as all the routing costs c_i are equal to 1. Moreover, all the links are saturated in the optimal solution.

Test DRIF is encumbering because of the memory requirements. This explains why algorithm NAF was not able to treat this problem.

p	n	K	\hat{p}	\hat{n}	name
26	42	257	6981	34994	NOE26
119	302	900	108302	758102	DRIF

Table 4.4.3: Size of problems solved

Table 4.4.4 shows results obtained by NAF. In the table Iter is the number of iteration of the algorithm.

Iter	optimal cost	computing time (sec)	name
17	9879.004	2472.883	NOE26
X	X	X	DRIF

Table 4.4.4: Performance of the algorithm NAF

Unfortunately, we were not able to solve the problem DRIF because of the shortage of memory mentioned before.

The results for APF are displayed in Table 4.4.5 where Outer Iter is the number of master programs solved and Inner Iter, the iteration number of predictor-corrector algorithm.

Outer Iter	Inner Iter	optimal cost	computing time (sec)	name
5	68	9879.022	1501.117	NOE26
5	79	6040.601	51352.37	DRIF

Table 4.4.5: Performance of the algorithm APF

This numerical results show the superiority of APF.

4.5 Conclusion

We have developed routing algorithms for node-arc and arc-path formulation and we have obtained promising results. They both are suitable for a distributed implementation on a massively parallel computer. The algorithm NAF proposed in this paper can be improved because the aggregation of the commodities is possible in the case of node-arc formulation. Since the cost per unit of flow on a given link does not depend on the commodity, all the commodities which have a common endpoint can be merged. Therefore the problem reduces to a number of commodities not larger than p the number of nodes.

p	n	K	\hat{p}	\hat{n}	name
26	42	24	690	3306	NOE26-merged
119	302	63	7862	53348	DRIF-merged

Table 4.5.6: size of aggregated dates

DRIF problem can then be solved with NAF. We can also improve both of NAF and APF algorithms by solving the Newton Direction approximatively. The matrix M is symmetric positive definite. Then one can use an iterative method to solve the reduced system as the conjugate gradient method.

5. Conception de réseaux de télécommunications

5.1 Motivation

Damage to fiber cable hinders phone service (22 octobre 1987) [1]

Phone snafu isolates New Jersey - Long distance cable severed (19 Novembre 1988) [3]

Fire in fiber gateway sparks flight delays, problems at brokerages (11 Mai 1988) [2]

Chicago's O'Hare Airport came to a standstill - emergency 911 was no more - Automatic teller machines in the Chicago area were down - Dollar estimates of lost business ranged from the hundreds of millions to the tens of billions (8 Mai 1988) [133].

Ces titres ont choqué le public et les compagnies de téléphone.

Dans ces cas, les réseaux de télécommunications n'ont pas été conçus avec assez de surplus pour survivre en cas de panne. La sécurisation n'était pas importante pour la conception de réseaux de cuivre traditionnel. Ceci est dû à la capacité limitée des câbles en cuivre qui entraîne la diversification du routage. Par conséquent la défaillance d'un arc n'affecte qu'une petite proportion du trafic.

Grâce à la technologie de la fibre optique, de très grandes capacités peuvent être installées sur les arcs d'un réseau. Ce qui entraîne la possibilité de topologie plus creuse de ces réseaux.

Cette faible densité des topologies de réseaux appelle à une plus grande attention à la conception de réseaux en terme de sécurisation. Un réseau est dit *sécurisable* s'il reste encore opérationnel en cas de pannes.

Les réseaux de télécommunications sont des systèmes complexes destinés à transmettre des informations entre des paires de nœuds. Les composantes de ces systèmes sont exposées à des pannes. Dans ce cas le trafic doit être rétabli dans le plus bref délai. De nos jours, l'une des plus importantes questions de conception des réseaux de télécommunications est la *sécurisation*.

Les réseaux sécurisés coûtent, en général, moins cher que ceux qui ont une conception moins robuste [25].

Il est donc essentiel de trouver un compromis entre le coût et la sécurisation. Wu dans [129] ainsi que Grötschel, Monma et Stoer dans [56] ont identifié la sécurisation de réseau comme une des questions les plus significatives pour concevoir un réseau de télécommunications.

5.2 Introduction

Dans ce chapitre nous décrivons une série de modèles mathématiques de conception de réseaux de télécommunications qui ont été développés récemment.

Un problème de conception de réseau est défini par une suite de décisions telle que le choix des technologies utilisées, la topologie du réseau, la planification de la capacité des composantes du réseau, la décision concernant la stratégie de routage et les traitements des pannes. Concevoir un réseau qui survit à certaines pannes au moindre coût est l'une des priorités des opérateurs de télécommunications.

Le problème le plus général est le suivant :

Supposons que nous avons une demande entre chaque paire de nœuds.

Nous considérons le problème de choisir **la capacité** à installer sur chaque arc possible du réseau afin de :

- i.* satisfaire toutes les demandes,
- ii.* minimiser le coût de la construction du réseau.

En plus de la détermination de la **topologie** du réseau et des capacités des arcs, nous devons fournir à chaque demande, un **routage** vérifiant :

- iii.* aucun chemin ne peut porter plus d'un **pourcentage** de la demande,
- iv.* aucun chemin de routage ne dépasse une certaine **longueur**.

Nous devons également nous assurer que :

- v.* pour toute panne élémentaire d'arc ou de nœud, un certain **pourcentage** de la demande puisse être **rerouté**.

De plus, pour toute panne, un routage réalisable doit être fourni.

Ce modèle général a été introduit et formulé comme un programme mixte par Alevras et al dans [6].

Selon les critères de conception du réseau, nous spécialisons le modèle décrit ci-dessus. Si l'installation de capacité fait partie du problème, il faut décider si celle-ci est discrète ou continue. Plusieurs modèles dans la littérature considèrent le problème d'installation de capacités discrètes ou modulaires selon les normes des opérateurs : [11, 12, 90, 10].

Dans certains cas, nous pouvons être amenés à limiter la longueur des chemins dans le réseau [92]. Ceci se produit par exemple dans le cas où les capacités sont très petites. Le terme de *sécurisation* ne se réduit pas à un seul type de problèmes mais comprend une large classe de problèmes. Chacun de ces derniers s'intéresse à un certain aspect du réseau.

Plus généralement, un problème de sécurisation de réseau de télécommunications a pour but d'assurer les propriétés défensives contre ses défaillances. Ces propriétés peuvent être extrêmement diverses. Nous citons dans la suite des exemples de propriétés défensives qui ont été abordées dans la littérature.

5.3 Sécurisation topologique

Une importante propriété défensive est de garder le réseau connecté en cas de panne de certaines composantes. Il s'agit alors de la sécurisation topologique. Stoer a consacré un ouvrage [127] à l'étude de ce type de sécurisation.

Pour garantir la sécurisation topologique d'un réseau il est fréquent de le concevoir de telle sorte que toute paire de nœuds soit liée par un certain nombre de chemins disjoints par nœuds et /ou par arcs. Ce concept s'intitule la **k-connexité** [6]. Le problème de sécurisation topologique est l'un des premiers problèmes de conception de réseau étudié [126, 24]. Le concept de sécurisation de certaines topologies de réseau est relié au concept de connexité. Comme cas particulier de ces problèmes, nous citons celui de l'*arbre de Steiner*. Le cas bi-connexe a été étudié par Monma et Shallcross dans [107], où ils ont présenté une approche heuristique qui permet d'obtenir des solutions approchées pour des réseaux de tailles réelles ayant jusqu'à 200 sommets. Dans [106], Monma, Munson et Pulleyblank se sont intéressés à l'étude théorique de la structure de la solution de ce problème. Des approches basées sur les méthodes

branch and cut ont été proposées pour la résolution de problèmes réels de basse connectivité ($k \leq 2$) [58, 59, 60]. Plus récemment, Fortz, Labbe et Maffioli ont proposé dans [38] une approche similaire qui prend en compte des contraintes supplémentaires (*Mesh constraints*). Ces contraintes imposent que toutes les arêtes du graphe de support appartiennent à un cycle de longueur bornée.

Par ailleurs, Stoer [127] montre que le problème de haute connectivité ($k > 2$) peut être résolu pour des réseaux ayant jusqu'à 500 nœuds.

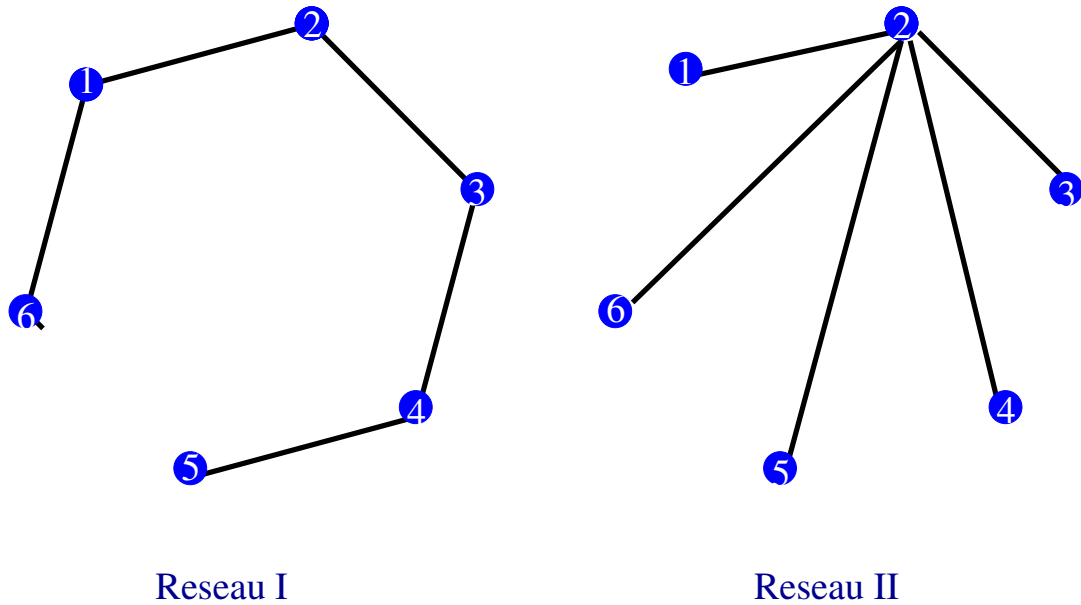
Pour une étude plus détaillée de ces types de problèmes de sécurisation et des travaux effectués dans ce cadre nous nous référons à [56].

Monma et Shallcross [107], Grötschel et Monma [57] et Grötschel, Monma et Stoer [59, 60, 56] ont considéré le problème de conception de réseau de télécommunications sujet à certaines contraintes de connectivité.

Dans leurs modèles, l'*arc-connectivité* (respectivement *nœud-connectivité*) d'une paire de nœuds est définie comme le nombre de chemins disjoints par arcs (respectivement par nœuds) liants ces nœuds. Lors de défaillance d'une composante du réseau, le service peut être restauré en reroutant le trafic autour de la partie en panne de ce réseau, si une connexion supplémentaire est disponible dans la topologie du réseau. Les contraintes de connectivité assure l'existence d'un nombre de tels chemins entre toute paire de nœuds du réseau. Cependant ces contraintes ne garantissent pas le niveau de sécurisation du réseau. Des niveaux variés de sécurisation sont possibles pour des réseaux satisfaisant les mêmes contraintes de connectivité.

Dans les réseaux de communication, la sécurisation est habituellement définie comme un pourcentage du trafic total survivant à une certaine panne (calculé dans le pire des cas). Un effort de conception de réseaux sécurisés a été effectué mais ces travaux ne traitent pas directement les contraintes de sécurisation. Une grande partie des modèles de conception de réseaux propose d'assurer indirectement la sécurisation du réseau en ayant recours aux contraintes de connectivité. En effet dans ces modèles, le taux de sécurisation intervient uniquement pour trancher entre les différentes architectures obtenues avec la même connectivité.

Récemment, Myung, Kim et Tcha [109] ont introduit un nouveau modèle de conception de réseau qui traduit la sécurisation directement en terme de contraintes de sécurisation spécifiant le niveau toléré de perte du trafic en cas de pannes suivant des conditions prescrites.



Taux de Connexite 1- connexe

Taux de Securisation 53,33%

Taux de Securisation 33,33%

Securisation et Connexite

Figure 5.3.1: Exemple de réseaux avec la même connexité

Ce nouveau modèle permet de considérer un ensemble de topologies plus riche que celui des modèles de connexité et de mettre ces derniers dans un cadre plus général. La **k-arête-sécurisation** d'un réseau est définie dans [109] comme une portion relative du trafic qui est encore intacte dans le cas de la pire panne de k liens. Le problème de conception de réseau avec des contraintes de k -arête-sécurisation étudié dans [109] peut être étendu pour traiter les pannes de nœuds en définissant par analogie la sécurisation k -nœud.

Dans [109] on montre que le problème de k -arête-sécurisation peut être résolu en temps polynômial pour un réseau k -arête-connexe.

5.4 Dimensionnement d'un réseau et Routage

Une autre propriété défensive importante est la résistance du routage aux pannes. Si le problème de conception du réseau comporte le routage, une stratégie raisonnable pour garder le réseau en vie est d'exiger qu'aucun chemin ne transporte plus d'un certain pourcentage du trafic total entre toutes paires de nœuds. Ce concept est connu sous le nom de **diversification**. Dans le cas où le modèle intègre la planification de la capacité et le routage, une variante naturelle du concept de k -connexité est d'imposer le reroutage d'un certain pourcentage du trafic entre toute paire de nœuds, en cas de toute défaillance. Ce concept est appelé la **réserve**.

Minoux [98] a été le premier à considérer la sécurisation dans un modèle général de multiflot avec capacités continues en généralisant l'algorithme de Gomory et Hu [45] pour les flots non-simultanés. Son modèle assure la survie par réserve et peut résoudre des réseaux ayant jusqu'à 40 nœuds avec une précision de 5%.

Dahl et Stoer [27, 28] étaient les premiers à considérer le modèle de sécurisation avec capacités discrètes. Leur modèle suit les concepts de diversification et de réserve. Pour résoudre le problème général de conception de réseau décrit à l'introduction de ce chapitre, Alevras, Grötschel et Wessäly distinguent dans [6] deux types de capacités discrètes (capacités modulaires et capacités discrètes arbitraires) et trois façons d'assurer la survie du réseau (diversification, réserve et reroutage des demandes affectées). Ils proposent de résoudre les six modèles de conception obtenus en combinant les deux types de capacités et les trois approches de survie.

D'après leur étude numérique, la diversification est plus chère mais moins coûteuse en temps de calcul. Ce résultat est prévisible, puisqu'en adoptant la survie par diversification, les capacités des arcs à l'état opérationnel normal sont choisies de telle sorte que toutes les situations de pannes peuvent être maîtrisées sans reroutage. Ce qui revient à la résolution d'un problème de routage.

La réserve est par contre moins chère et plus coûteuse en temps de calcul. Le meilleur compromis entre le coût et l'effort de maintenance semble être obtenu avec le modèle de reroutage des demandes affectées. Les solutions sont proches de celles obtenues avec la réserve et l'effort de reroutage des demandes affectées est relativement petit (par rapport à la réserve).

En coopération avec France-Télécom, Lisser, Sarkissian et Vial [86] ont développé un

modèle traitant simultanément le dimensionnement de réseau et le routage des demandes. Ils considèrent que le reroutage de la partie de la demande affectée (par une panne) s'effectue dans un réseau indépendant appelé *réseau de réserve*. Ce dernier est installé en parallèle avec le réseau nominal et ne devient actif qu'en cas de pannes qui résultent de nouvelles demandes. L'avantage de l'installation d'un réseau de réserve est de satisfaire aux demandes créées par une panne sans interférer avec la distribution du trafic courant.

Les deux réseaux partagent le même support topologique. Par conséquent ils ont les mêmes défaillances.

Pour rerouter le trafic interrompu nous distinguons deux types de reroutage : le reroutage local et le reroutage global.

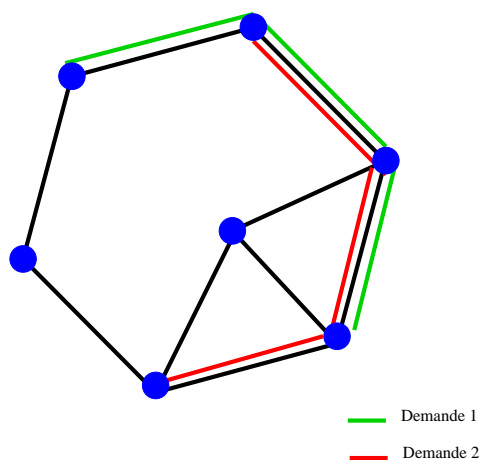


Figure 5.4.2: Routage des demandes

La stratégie locale est de considérer que la rupture d'un arc crée entre ses extrémités une demande égale à la somme des flots qu'il transmettait. Dans ce cas, à chaque panne d'arc nous associons un problème de flot simple.

L'approche globale du reroutage consiste à analyser le flot créé par la panne et à trouver les fractions des demandes affectées par cette panne dans le réseau nominal. A chaque panne d'arc nous associons dans ce cas un problème de multiflot.

Le problème de sécurisation qui nous intéresse dans cette étude, a été traité dans [86] en considérant le reroutage local. Il consiste à trouver le routage et l'investissement en capacités nominale et de réserve de moindre coût pour un certain jeu de demandes. Ces capacités doivent suffire pour permettre respectivement un routage du trafic

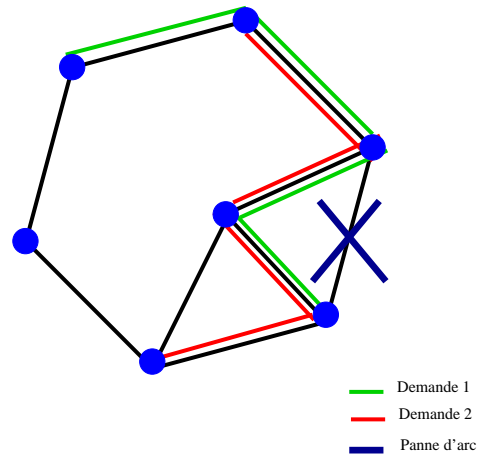


Figure 5.4.3: Reroutage local

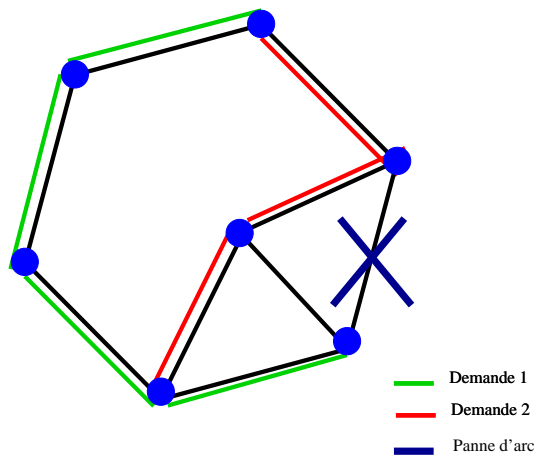


Figure 5.4.4: Reroutage global

nominal optimal et un reroutage (local dans [86]) de ce trafic dans le cas de n'importe quelle panne simple d'arc. Dans [86] Lissier, Sarkissian et Vial formulent le problème comme un programme linéaire de très grande taille qu'ils résolvent à l'aide d'un algorithme de plans sécants basé sur le concept de centre analytique (ACCPM). La méthode permet ainsi de sécuriser avec l'approche locale des réseaux d'une taille allant jusqu'à 60 noeuds et 120 arcs avec un saut de dualité de 7.10^{-5} , ce qui correspond à une modélisation réaliste du réseau de transmission d'interconnexion de France-Télécom. La stratégie locale est sous optimale et la globale est plus complexe. Pour remédier partiellement à cette imperfection de l'approche locale, Lissier et al proposent une approche hybride qui procède comme suit. Dans un premier temps l'algorithme résout le problème de sécurisation avec reroutage local. Ensuite,

le routage et la capacité nominale obtenus seront considérés comme des données fixes. Le réseau de réserve est alors reconçu pour réaliser la survie du routage avec reroutage global. La dernière étape consiste à trouver l'investissement en capacité de réserve le moins coûteux pour assurer le reroutage global du trafic nominal obtenu à la première étape. Dans [39] Frangioni traite le problème de la dernière étape par une approche basée sur la méthode de faisceaux [85].

Le modèle de conception de réseau qui nous intéresse est proche de celui de Lisser, Sarkissian et Vial présenté dans [86]. Pour une vue globale du modèle de [86], des approches de résolutions, et des résultats obtenus, nous renvoyons le lecteur à la thèse de Sarkissian [124].

Dans la suite de ce chapitre nous introduisons le problème de sécurisation globale. Les formulations possibles de ce problème sont exposées dans le chapitre suivant.

5.5 Sécurisation globale des réseaux de transmission

Dans la suite de notre étude le réseau de télécommunications est dit *sécurisable* si, à la suite d'une panne, le trafic interrompu peut être rerouté à travers le réseau en utilisant la capacité résiduelle.

Le Problème de Sécurisation Globale (PSG) consiste à trouver le multiflot de coût minimum et l'investissement en capacités nominale et de réserve de moindre coût. De plus, la capacité nominale doit être suffisante pour assurer le routage et celle du réserve doit garantir la survie du trafic nominal par reroutage **global** en cas de n'importe quelle panne d'arc.

Nous appelons *panne simple* toute panne d'un seul élément du réseau. Cet élément peut être un arc ou un nœud. La possibilité d'avoir simultanément plusieurs pannes est extrêmement improbable. La défaillance d'un nœud touche tous les arcs qui lui sont adjacents dans le réseau.

Nous nous intéressons uniquement aux pannes élémentaires d'arcs non simultanées. Nous nous sommes limités à ce cas pour des raisons de complexité numérique.

Dorénavant, toute panne sera associée à un seul arc. L'ensemble des pannes n'est autre que l'ensemble des arcs du réseau.

Une panne est dite **active**, si elle perturbe le routage d'au moins une demande.

L'ensemble des pannes actives coïncide avec l'ensemble des arcs qui portent ce routage. Dans notre modèle nous considérons que la topologie du réseau est donnée et est fixe. Nous supposons que le réseau est au moins bi-connexe pour assurer sa sécurisation topologique. Une panne active crée une demande entre les extrémités de l'arc. Dans notre modèle, nous considérons l'existence d'un second réseau installé en parallèle avec le réseau nominal et appelé le réseau de réserve. Les deux réseaux ont une topologie identique et sont de même type. En rappelant que le réseau de réserve ne devient actif qu'en cas de pannes actives, nous distinguons trois possibilités d'utiliser les capacités nominale et de réserve pour le reroutage. Avant d'énumérer ces possibilités, nous précisons qu'une panne active libère la capacité nominale qui était occupée par les demandes affectées.

- La première possibilité consiste à utiliser uniquement la capacité nominale présentée sous forme de capacité résiduelle nominale et capacité libérée par la panne. Mais avec cette utilisation des capacités nous utilisons un seul réseau (le nominal). Ce qui est incompatible avec le choix de notre modèle.
- La deuxième possibilité est d'exploiter le réseau de réserve et d'effectuer le reroutage en considérant les capacités libérée et résiduelle dans le nominal et la capacité de réserve.
- Enfin nous pouvons dissocier les réseaux nominal et de réserve, en consacrant le premier exclusivement au routage et le second au reroutage. Cette possibilité est plus simple à gérer et justifie vraiment l'introduction du réseau de réserve.

Nous allons voir dans le chapitre suivant que la dernière possibilité est la seule à pouvoir être traduite en formulation sommets-arcs (la capacité libérée ne peut pas être calculée).

Nous adoptons uniquement la dernière possibilité de gestion des capacités.

Dans notre modèle le fractionnement du trafic est autorisé. Le problème PSG revient à trouver les capacités nominale et de réserve et le multiflot (routage) dans le réseau nominal vérifiant :

- le multiflot satisfait toutes les demandes,
- le multiflot est compatible avec la capacité du réseau nominal,

- le multiflot peut être sécurisé par reroutage global dans le réseau de réserve dans le cas de n'importe quelle panne,
- le coût total est minimum.

Malgré les simplifications apportées au modèle, ce dernier reste suffisamment compliqué. Nous allons voir que le PSG se formule comme un problème linéaire de très grande taille avec plusieurs niveaux de couplage et que la formulation arcs-chemins est la plus appropriée.

6. Modélisation du problème de sécurisation globale de réseaux

6.1 Introduction

Au chapitre précédent nous avons introduit le problème de sécurisation globale (PSG) dans le cadre général de problèmes de conception de réseau. Nous avons également discuté le choix de son modèle. Ce chapitre est consacré à l'étude mathématique de PSG. Comme le problème de routage, le PSG peut être écrit sous formulation sommets-arcs ou arcs-chemins [72]. Nous allons surtout insister sur la formulation arcs-chemins, compte tenu de sa supériorité sur la formulation sommets-arcs pour le problème de routage.

Dans l'approche globale du reroutage chaque éventuelle panne crée un problème de multiflot (par contre dans l'approche locale, une panne crée un problème de flot simple). Il faut donc reconnaître les fractions des demandes affectées par la panne dans le réseau nominal pour pouvoir les rerouter. Il est toujours possible de calculer ces fractions de demandes mais seule la formulation arcs-chemins permet d'avoir la capacité libérée dans le réseau nominal.

En effet, la formulation sommets-arcs permet de calculer la partie de la demande qui était portée par l'arc en panne, mais elle ne fournit aucune information concernant le trafic nominal non affecté par cette panne. L'information qui manque à la formulation sommets-arcs est l'ensemble des chemins qui passent par l'arc en panne. La reconstitution de ces chemins est possible mais alourdit la mise en œuvre de l'algorithme. Par contre avec la formulation arcs-chemins, cette information devient accessible, puisque les variables de décision sont les flots sur les chemins.

Dans le cas d'une panne, les routages interrompus doivent être reroutés en évitant la composante défaillante. La capacité de secours nécessaire au routage des demandes créées par la panne peut être sous forme de capacité nominale résiduelle, ou sous forme de capacité de réserve fournie par le réseau de réserve ou encore sous forme de

capacité nominale libérée par la panne. Nous avons choisi de ne pas utiliser la capacité libérée par la panne (ni la capacité résiduelle). Par conséquent, notre modèle peut aussi bien être formulé en terme de flots sur les arcs (sommets-arcs), qu'en terme de flots sur les chemins (arcs-chemins). Nous allons voir dans la suite que PSG peut être formulé comme un problème linéaire de très grande taille avec plusieurs niveaux de couplage.

Ce chapitre est organisé comme suit. Nous commençons par poser et formuler PSG (section 6.2). Nous nous concentrons sur la formulation arcs-chemins dans un premier temps. Dans le cadre de cette formulation, nous explicitons une méthode de génération de chemins (section 6.3) et nous proposons des garanties de solution (section 6.4). La 5^{ème} section est consacrée à la formulation sommets-arcs de PSG. Nous discutons cette formulation et nous montrons ses limites numériques. Dans la 6^{ème} section nous présentons des réseaux particuliers : les réseaux symétriques (différents des graphes symétriques). Ces réseaux nous seront utiles au stade des essais numériques.

6.2 Description du problème

Le PSG est caractérisé par la donnée d'un graphe $\mathcal{G}(\mathcal{V}, \mathcal{E})$ et l'ensemble \mathcal{K} de K flots qui le traversent. Le graphe \mathcal{G} est bi-connexe non orienté et est défini par \mathcal{V} , un ensemble de p nœuds, et \mathcal{E} , un ensemble de n arcs. Ce graphe représente le support topologique de deux réseaux : le réseau nominal et le réseau de réserve. La capacité est une fonction de \mathcal{E} dans \mathbb{R} . Elle sera notée comme un vecteur de \mathbb{R}^n .

Nous désignons par y le vecteur capacité du réseau nominal et par z celui du réseau de réserve.

Un flot k de \mathcal{K} est défini par une paire de nœuds source-puits (o^k, p^k) et une quantité de trafic, $r^k \in \mathbb{R}$.

Pour obtenir une formulation convenable à la décomposition, nous proposons d'utiliser le fait que toute demande peut être décomposable en une somme de flots portés par des chemins simples liant la paire origine-destination de la demande.

Nous associons à tout flot $k \in \mathcal{K}$ les ensembles $I^{0,k}, I^{1,k}, \dots, I^{n,k}$ définis par :

- $I^{0,k}$: ensemble de chemins nominaux élémentaires distincts entre le nœud source o^k et le nœud puits p_k .
- $I^{e,k}$: ensemble de chemins de réserve élémentaires distincts entre le nœud source

o^k et le nœud puits p^k en cas de panne de l'arc e dans \mathcal{G} .

Pour tout $j \in \{0, \dots, n\}$, l'ensemble de chemins $I^{j,k}$ est caractérisé par une matrice $\Pi^{j,k}$ de taille $n \times n^{j,k}$, $n^{j,k}$ étant le cardinal de l'ensemble $I^{j,k}$. Toute colonne $\Pi_i^{j,k}$ de cette matrice, caractérise un chemin de l'ensemble $I^{j,k}$ et vérifie :

$$(\Pi_i^{j,k})_e = \begin{cases} 1 & \text{si } e \text{ appartient au chemin } i, \\ 0 & \text{autrement.} \end{cases} \quad (6.2.1)$$

Nous notons $x_i^{j,k}$ le flux du flot k porté par le chemin caractérisé par $\Pi_i^{j,k}$.

La somme sur i de tous les flux $x_i^{0,k}$ est égal à la valeur de la demande r^k :

$$\sum_{i \in I^k} x_i^{0,k} = r^k.$$

Pour simplifier la formulation de PSG nous posons $x^{0,k} = x^k$, $\Pi^{0,k} = \Pi^k$, $I^{0,k} = I^k$ et $n^{0,k} = n^k$ pour tout flot $k \in \mathcal{K}$.

Le problème de sécurisation globale peut ainsi s'exprimer en fonction des variables de routage uniquement.

En effet, le coût minimum d'investissement de capacité de réserve pour sécuriser un routage $x = (x^1, \dots, x^K)$ dépend de ce routage et peut être désigné par $f(x)$.

Le problème peut alors être formulé comme suit :

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K, y} c^T y + f(x) \\ \sum_{k=1}^K \sum_{i \in I^k} x_i^k \Pi_i^k \leq y, \\ k \in \mathcal{K} \left\{ \begin{array}{l} \sum_{i \in I^k} x_i^k = r^k, \\ x_i^k \geq 0, \end{array} \right. \\ y \geq 0. \end{array} \right. \quad (6.2.2)$$

Explicitons la façon de calculer la fonction $f(x)$. La panne d'un arc e crée au plus K demandes. Nous notons \mathcal{K}^e l'ensemble des flots affectés par la panne e . A chaque k dans \mathcal{K}^e nous faisons correspondre $r^{e,k}$, la quantité de flot à rerouter dans le réseau de réserve. Cette quantité représente le flux du flot k qui traversait l'arc e dans le réseau nominal :

$$r^{e,k} = \sum_{i \in I^k} x_i^k (\Pi_i^k)_e.$$

Comme dans le cas du routage, le problème de reroutage doit respecter des contraintes de satisfaction des demandes et des contraintes de capacité.

Pour tout flot k et toute panne e nous associons la contrainte de satisfaction de demande suivante, que nous appellerons la contrainte de reroutage :

$$\sum_{i \in I^{e,k}} x_i^{e,k} = \Pi_e^k x^k.$$

Rappelons que les réseaux nominal et de réserve partagent le même support topologique. Lorsqu'une composante du nominal est défaillante, la composante correspondante dans le réseau de réserve l'est aussi. Dans le cas d'une panne d'un arc e dans le réseau nominal, la capacité de l'arc correspondant dans le réseau de réserve est donc nulle.

Le réseau nominal libère de la capacité suite à une panne e qui n'est autre que

$$y_e = \sum_{k=1}^K \sum_{i \in I^k} x_i^k (\Pi_i^k)_e \Pi_i^k.$$

Nous désignons par $\hat{\Delta}_e$ la matrice diagonale d'ordre n définie par:

$$\hat{\Delta}_e = \text{diag}(\mathbf{1} - \delta_e),$$

avec δ_e est un vecteur de \mathbb{R}^n ayant toutes ses composantes nulles sauf $(\delta_e)_e$ qui vaut 1.

L'expression de la capacité de secours dépend de la stratégie de reroutage :

1. Le reroutage est effectué dans le réseau nominal.

Dans ce cas l'algorithme utilise la capacité libérée par la panne e dans le réseau nominal pour effectuer le reroutage. Nous pouvons envisager les deux sous-cas suivants :

- (a) le trafic interrompu est rerouté uniquement dans le réseau nominal en utilisant la capacité libérée par la panne. La capacité de secours en cas de la panne de l'arc e est alors égale à $(\mathbf{1} - \Delta_e)y_e$.

- (b) le trafic interrompu est rerouté dans les réseaux nominal et de réserve, en utilisant la capacité libérée par la panne e dans le réseau nominal et la capacité z dans le réseau de réserve.

La capacité de secours est dans ce cas égale à $(\mathbf{1} - \Delta_e)(z + y_e)$.

2. Le reroutage est effectué dans le réseau de réserve uniquement :

Dans ce cas la capacité de secours en cas de la panne e est égale à $(\mathbf{1} - \Delta_e)z$

Pour éviter les interférences entre les flots du routage et ceux du reroutage, nous choisissons le dernier cas de reroutage. Nous adoptons donc la formulation suivante :

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K, y, x^{1,1}, \dots, x^{n,K}, z} c^T y + d^T z \\ \text{(i)} \sum_{k=1}^K \sum_{i \in I^k} x_i^k \Pi_i^k \leq y, \\ k \in \mathcal{K} \left\{ \begin{array}{l} \text{(ii.k)} \sum_{i \in I^k} x_i^k = r^k, \\ \text{(iii.k)} x^k \geq 0, \end{array} \right. \\ \text{(iv)} y \geq 0, \\ e \in E \left\{ \begin{array}{l} \text{(v.e)} \sum_{k=1}^K \sum_{i \in I^{e,k}} x_i^{e,k} \Pi_i^{e,k} \leq \hat{\Delta}_e z, \\ \text{(vi.e.k)} \sum_{i \in I^{e,k}} x_i^{e,k} = r^{e,k} \\ \text{(vii.e.k)} r^{e,k} = \sum_{i \in I^k} x_i^k (\Pi_i^k)_e, \\ \text{(ix.e.k)} x^{e,k} \geq 0, \end{array} \right. \\ \text{(x)} z \geq 0, \end{array} \right. \quad (6.2.3)$$

Les contraintes (6.2.3.i) et (6.2.3.v.e) sont des contraintes de capacité. Elles expriment le fait que la capacité nominale et celle du réseau de réserve ne peuvent pas être dépassées à chaque état opérationnel. Puisque la capacité nominale n'intervient qu'à l'étape de routage, la capacité résiduelle optimale est nulle. La contrainte (6.2.3.i) peut être donc remplacée par une contrainte d'égalité (à l'optimum les n contraintes de capacité sont actives).

Les contraintes (6.2.3.ii.k) ne sont autres que les contraintes de routage des flots dans le réseau nominal. Elles imposent la satisfaction des demandes nominales.

Le second terme d'une contrainte (6.2.3.vi.e.k), dont la valeur est calculée dans (6.2.3.vii.e.k), correspond à la demande créée par la panne de l'arc e entre la paire origine-destination (o^k, p^k) . Les contraintes (6.2.3.vi.e.k) représentent les contraintes de reroutage des routages interrompus. Le reste des contraintes, (6.2.3.iii.k), (6.2.3.iv), (6.2.3.ix.e.k) et (6.2.3.x) ne sont autres que des contraintes de positivité.

En exprimant la capacité nominale y en fonction des flots x^k et en adoptant une

représentation matricielle des contraintes de PSG, nous obtenons la formulation simplifiée suivante :

$$\left\{ \begin{array}{l}
 \text{Min}_{x^1, \dots, x^K, x^{1,1}, \dots, x^{n,K}, z} c^T \sum_{k=1}^K \Pi^k x^k + d^T z \\
 \\
 k \in \mathcal{K} \left\{ \begin{array}{ll}
 \text{(i.k)} \omega^k x^k = r^k, & (\alpha^k) \\
 \text{(ii.k)} x^k \geq 0, & (s^k)
 \end{array} \right. \\
 \\
 e \in E \left\{ \begin{array}{ll}
 \text{(iii.e)} x^{e,0} + \sum_{k=1}^K \Pi^{\epsilon,k} x^{\epsilon,k} = \hat{\Delta}_e z, & (\lambda^e) \\
 \text{(iv.e)} x^{e,0} \geq 0, & (s^{e,0}) \\
 \\
 k \in \mathcal{K}^e \left\{ \begin{array}{ll}
 \text{(v.e.k)} \omega^{\epsilon,k} x^{\epsilon,k} = \Pi_e^k x^k, & (\beta_e^k) \\
 \text{(vi.e.k)} x^{\epsilon,k} \geq 0, & (s^{\epsilon,k}).
 \end{array} \right.
 \end{array} \right. \\
 \\
 \text{(vii)} z \geq 0, & (t)
 \end{array} \right. \quad (6.2.4)$$

où

- $x^{e,0}$ est le vecteur capacité résiduelle de réserve associé à la panne de l'arc e .
- ω^k est le vecteur ligne unité : $\omega^k = (1, \dots, 1)$ de taille n^k .
- $\omega^{\epsilon,k}$ est le vecteur ligne unité de taille $n^{\epsilon,k}$.

Dans la formulation (6.2.4), nous écrivons entre parenthèses les multiplicateurs de Lagrange en face des contraintes correspondantes. Nous introduisons alors les multiplicateurs de Lagrange suivants :

- α^k : associé à la contrainte de satisfaction de la demande k (6.2.4.i.k) ($\alpha^k \leq 0$),
- s^k : associé à la contrainte de positivité du flot k (6.2.4.ii.k) ($s^k \geq 0$),
- λ^e : associé à la contrainte de capacité de secours en cas de la panne e (6.2.4.iii.e) ($\lambda^e \geq 0$),

- $s^{e,0}$: associé à la contrainte de positivité de la capacité résiduelle $x^{e,0}$ de la capacité de réserve en cas de la panne e (6.2.4. iv.e) ($s^{e,0} \geq 0$),
- β_e^k : associé à la contrainte de satisfaction de la demande du flot k créée par la panne e (6.2.4.v.e.k) ($\beta_e^k \leq 0$),
- $s^{e,k}$: associé au vecteur de reroutage $x^{e,k}$ du flot k en cas de la panne e (6.2.4.vi.e.k) ($s^{e,k} \geq 0$),
- t : associé à la contrainte de positivité de la capacité de réserve z (6.2.4.vii) ($t \geq 0$).

Dans la suite, pour des expressions telles que $xy = v$, avec x, y, v des vecteurs, nous entendons que l'opération est effectuée composante par composante.

Avec cette convention, le système d'optimalité associé au problème (6.2.4) s'écrit comme suit:

$$\left\{ \begin{array}{l}
 k \in \mathcal{K} \left\{ \begin{array}{l}
 x^k s^k = 0, \\
 \omega^k x^k = r^k, \\
 \Pi^{kT} c + \omega^{kT} \alpha^k - \Pi^{kT} \beta^k = s^k, \text{ (k.i)} \\
 x^k, s^k \geq 0,
 \end{array} \right. \\
 \\
 e \in E \left\{ \begin{array}{l}
 x^{e,0} s^{e,0} = 0, \\
 x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z, \\
 \lambda^{e,0} = s^{e,0}, \\
 x^{e,0}, s^{e,0} \geq 0, \\
 \\
 k \in \mathcal{K} \left\{ \begin{array}{l}
 x^{e,k} s^{e,k} = 0, \\
 \omega^{e,k} x^{e,k} = \Pi_e^k x^k, \\
 \beta_e^k \omega^{e,kT} + \Pi^{e,kT} \lambda^e = s^{e,k}, \text{ (e.k.ii)} \\
 x^{e,k}, s^{e,k} \geq 0,
 \end{array} \right. \\
 \\
 tz = 0, \\
 d - \sum_{e=1}^n \hat{\Delta}_e^T \lambda^e = t, \text{ (iii)} \\
 z, t \geq 0,
 \end{array} \right. \tag{6.2.5}
 \end{array} \right.$$

Il est important de voir que la formulation arcs-chemins ci dessus fait intervenir les matrices de chemins $\Pi^k, \Pi^{1,k}, \dots, \Pi^{n,k}$, pour $k = 1, \dots, K$. Le nombre de variables associées à ces chemins et les dimensions de ces matrices croissent exponentiellement avec la taille du graphe, ce qui peut constituer un inconvénient pour cette formulation.

Cependant, le nombre de contraintes associées à la formulation arcs-chemins est généralement moins important que le nombre de celles associées à la formulation sommets-arcs. Dans le cas de PSG, on remarque que la formulation (6.2.4) comporte

- K contraintes de routage (6.2.4.i.k),
- au pire n^2 contraintes de capacités de réserve (6.2.4.iii.e),
- et $n \times K$ contraintes de reroutage (6.2.4.v.e.k).

En effet, la formulation (6.2.4) associe n contraintes de capacité de réserve (6.2.4.iii.e) à chaque panne active e , dont une contrainte sur l'arc en panne : $x_e^{e,0} + \sum_{k=1}^K \Pi^{e,k} x_e^{e,k} = 0$. Ce qui revient à dire que $x_e^{e,0} = 0$ pour tout $e \in \{1, \dots, n\}$. Cette information peut être utile pour réduire la taille de la matrice des contraintes de n lignes et n colonnes. Si n^a désigne le nombre de pannes actives dans PSG, nous comptons $n^a \times n$ contraintes de capacités de réserve dans notre modélisation. Le nombre de contraintes de reroutage (6.2.4.v.e.k) dépend du nombre de couples (panne-flot affecté). Le nombre de contraintes maximal $\hat{p}^{pire} = K + n \times K + n^2 - n$ est atteint si tous les flots sont affectés par toutes les pannes. Il est important de signaler que ceci ne se produit que dans le cas des réseaux peu maillés (i.e. n petit). Donc, même si le nombre de contraintes au pire des cas est de l'ordre de n^2 , cette borne est atteinte lorsque n est de l'ordre de p . En tout cas le nombre de contraintes dans (6.2.4) n'est pas important relativement à la taille du problème. Une méthode de génération de colonnes, donc de chemins, s'impose (voir chapitre II).

Lorsque (6.2.4) a des solutions, il existe une solution optimale avec au plus $K + n \times K + n^2$ variables non nulles [44]. Certaines variables non nulles pourraient correspondre à des capacités résiduelles de réserve sur des arcs ($x_i^{0,e}$). Le nombre de variables de capacités résiduelles non nulles varie entre zéro et $(n^2 - 2 \times n)$. Il est nul lorsque toutes les pannes saturent tous les arcs du réseau de réserve. Sa valeur maximale

$n^2 - 2 \times n$, s'explique par le fait que tout arc admet au moins un reroutage (correspondant à une panne) qui sature sa capacité de réserve. Les n autres variables de capacités résiduelles nulles correspondent aux $(x_e^{0,e})$ pour tout $e \in \{1, \dots, n\}$. Donc PSG aurait besoin d'au plus \hat{p}^{pire} différents chemins pour assurer le routage et la survie de ce dernier. Ce nombre maximal de chemins générés permet de prévoir les limites numériques de la méthode. A chaque itération j , l'algorithme résout d'abord un problème de sécurisation globale en se limitant à des sous-ensembles de chemins : le *problème maître*.

La formulation standard du $j^{\text{ème}}$ problème maître est comme suit :

$$\left\{ \begin{array}{l} \text{Min } \hat{c}^j T \hat{X}^j ; \\ \hat{X}^j \\ ; \hat{A}^j \hat{X}^j = \hat{b}^j , \\ \hat{X}^j \geq 0, \end{array} \right. \quad (6.2.6)$$

où \hat{A}^j est définie par un sous-ensemble des colonnes de la matrice $\hat{A} =$

$$\left(\begin{array}{cccccccc} \omega^1 & & & & & & & \\ & \ddots & & & & & & \\ & & \omega^K & & & & & \\ \Pi_1^1 & & & \omega^{1,1} & & & & \\ & \ddots & & & \ddots & & & \\ & & \Pi_1^K & & \omega^{1,K} & & & \\ & & & \Pi^{1,1} & \dots & \Pi^{1,K} & Id(n) & -\hat{\Delta}_1 \\ \vdots & & \vdots & & & \dots & \dots & \vdots \\ \Pi_n^1 & & & & & \omega^{n,1} & & \\ & \ddots & & & & & \ddots & \\ & & \Pi_n^K & & & \omega^{n,K} & & \\ & & & \Pi^{n,1} & \dots & \Pi^{n,K} & Id(n) & -\hat{\Delta}_n \end{array} \right) .$$

$\hat{X}^j = ((x^1)^j, \dots, (x^K)^j, (x^{1,1})^j, \dots, (x^{1,K})^j, (x^{1,0})^j, \dots, (x^{n,1})^j, \dots, (x^{n,K})^j, (x^{n,0})^j, z^j)$,
 $\hat{c}^j = (c^T \Pi^1, \dots, c^T \Pi^K, 0, \dots, 0, 0, \dots, 0, \dots, 0, 0, d)$ et $\hat{b}^j = (r^1, \dots, r^K, 0, \dots, 0)$.

Ensuite l'algorithme met à jour les sous-ensembles de chemins nominaux et de réserve associés à chaque demande : *les problèmes satellites*.

Nous détaillons dans la suite les problèmes satellites que l'algorithme résout à chaque itération majeure. La résolution du problème maître ne sera pas décrite dans ce chapitre. Nous nous limitons à l'étude générale du problème et nous proposons des

estimations supérieures et inférieures de la solution. Des méthodes de résolution du programme maître seront proposées aux chapitres suivants.

6.3 Génération de chemins

Nous rappelons que le problème maître est un problème de sécurisation globale limité à des sous-ensembles de chemins.

La méthode de génération de chemins que nous adoptons n'est autre qu'une spécialisation de la méthode de générations de colonnes décrite au deuxième chapitre.

A chaque flot k et à chaque itération externe j , nous associons les ensembles suivants:

- I_j^k : sous-ensemble de I^k sur lequel l'algorithme restreint le routage du flot k dans le réseau nominal à la $j^{\text{ème}}$ itération externe.
- $I_j^{e,k}$: sous-ensemble de $I^{e,k}$ sur lequel l'algorithme restreint le reroutage du flot k dans le réseau de réserve en cas de la panne e , à la $j^{\text{ème}}$ itération externe.

Nous allons voir qu'un sous-ensemble $I_j^{i,k}$ admet au plus j chemins. La gestion des sous-ensembles I_j^k et $I_j^{e,k}$, $e \in \{1, \dots, n\}$ est expliquée dans la sous-section qui suit.

A chaque itération externe, l'algorithme commence par résoudre un problème maître (réduit) en calculant une solution $(x_j, x_j^1, \dots, x_j^n, z_j)$. Les composantes de cette solution associées aux chemins qui sont dans $I^k - I_j^k$ et $I^{k,e} - I_j^{k,e}$ sont nulles.

Dire que $(x_j, x_j^1, \dots, x_j^n, z_j)$ est solution du problème de sécurisation globale (6.2.4) signifie qu'il existe des multiplicateurs $(\alpha^{1^*}, \dots, \alpha^{K^*}, \beta^{1^*}, \dots, \beta^{K^*}, \lambda^{1^*}, \dots, \lambda^{n^*})$ tels que le système d'optimalité associé au problème (6.2.5) soit vérifié.

Ceci revient à dire que $(x_j, x_j^1, \dots, x_j^n, z_j)$ est optimal si et seulement si en considérant les multiplicateurs associés on a, pour tout flot k , les inégalités suivantes :

$$\left\{ \begin{array}{l} (i) \Pi_i^{kT} (c - \beta^k) + \alpha^k \geq 0, \quad i \in I^k \\ e \in \mathcal{E} \left\{ \begin{array}{l} (ii.1) \beta_1^k + \Pi_i^{1,kT} \lambda^1 \geq 0, \quad i \in I^{1,k} \\ \vdots \\ (ii.e) \beta_e^k + \Pi_i^{e,kT} \lambda^e \geq 0, \quad i \in I^{e,k} \\ \vdots \\ (ii.n) \beta_n^k + \Pi_i^{n,kT} \lambda^n \geq 0, \quad i \in I^{n,k} \end{array} \right. \end{array} \right. \quad (6.3.1)$$

Nous rappelons que pour $(\lambda^1, \dots, \lambda^n)$, solution duale du programme maître (6.2.6), nous avons $d \geq \sum_{e=1}^n \hat{\Delta}_e^T \lambda^e$.

La première inégalité (6.3.1.i) est vérifiée par les chemins appartenant à I_j^k .

Pour toute panne e et tout flot k , l'inégalité (6.3.1.ii.e) est vérifiée par les chemins de l'ensemble $I_j^{e,k}$ correspondant.

Pour prouver l'optimalité du point $(x_j, x_j^1, \dots, x_j^n, z_j)$ pour le problème de sécurisation globale il faut et il suffit de montrer que les inégalités (6.3.1.i), (6.3.1.ii,1), \dots , (6.3.1.ii,n) peuvent respectivement s'étendre au reste des ensembles $I^k, I^{1,k}, \dots, I^{n,k}$.

Nous désignons par :

- Π_{jk}^k un plus court chemin entre le nœud source o^k et le nœud puits p^k obtenu en considérant comme vecteur coût unitaire le vecteur $c - \beta^k$.
- Π_{jke}^k un plus court chemin de secours entre le nœud source o^k et le nœud puits p^k en cas de la panne de l'arc e obtenu en considérant comme vecteur coût unitaire le vecteur λ^e . Nous savons que pour tout $e \in \{1, \dots, n\}$, λ^e est positif, puisque λ^e n'est autre que la variable duale associée à la capacité résiduelle de réserve en cas de panne de l'arc e (6.2.5).

Lemme 16. Le point $(x_j, x_j^1, \dots, x_j^n, z_j)$ est optimal pour le problème de sécurisation globale (6.2.4) si pour tout flot $k \in 1, \dots, K$:

- le chemin Π_{jk}^k vérifie (6.3.1.i).
- et pour toute panne $e \in E$ le chemin Π_{jke}^k vérifie (6.3.1.ii.e).

Preuve. Si le chemin Π_{jk}^k vérifie l'inégalité (6.3.1.i) alors tout chemin $\Pi_i^k \in I^k$ la vérifie aussi. Puisque pour tout chemin $\Pi_i^k \in I^k$, nous avons :

$$\Pi_i^{kT}(c - \beta^k) + \alpha^k \geq \Pi_{jk}^{kT}(c - \beta^k) + \alpha^k.$$

L'inégalité (6.3.1.i) peut être donc étendue à l'ensemble I^k .

Par analogie nous pouvons montrer que le reste du système (6.3.1) est vérifié.

Ce qui prouve l'optimalité du point $(x_j, x_j^1, \dots, x_j^n, z_j)$ pour le problème de sécurisation globale (6.2.4). □

Remarque 6.3.1. Le schéma de résolution de PSG que nous avons exposé exclut les méthodes directes et nécessite une génération de chemins nominaux et de réserve à

chaque fin d'itération externe.

Le principe de génération de chemin décrit ci-dessus peut être utilisable en pratique si nous avons les informations nécessaires (qui sont les valeurs des multiplicateurs). Nous allons voir que certains des algorithmes proposés ne calculent pas tous les multiplicateurs (λ^e dans DP1 et DP2) mais seulement des estimations de ces multiplicateurs. La génération de nouveaux chemins se ramène à un calcul de plus courts chemins avec coût unitaire positif, que nous calculons avec l'algorithme de Dijkstra [46].

6.4 Calcul d'une estimation inférieure

Nous présentons dans cette section une estimation inférieure en supposant que tous les multiplicateurs sont calculés par l'algorithme qui résout le programme maître. Afin d'avoir une meilleure estimation, nous proposons une méthode de perturbations des multiplicateurs. Cette méthode de perturbation sera particulièrement utile lorsque le programme maître n'est pas résolu jusqu'à l'optimalité (comme dans [51] et [42]).

6.4.1 Choix de l'estimation inférieure

Considérons la formulation suivante du problème de sécurisation globale:

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K, x^{1,1}, \dots, x^{n,K}, z} c^T \sum_{k=1}^K \Pi^k x^k + d^T z; \\ \omega^k x^k = r^k, \\ x^k \geq 0, k = 1, \dots, K, \\ e \in E \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z, \\ \omega^{e,k} x^{e,k} = \Pi_e^k x^k, \quad (\beta_e^k) \\ x^{e,k} \geq 0, k = 0, \dots, K, \quad . \end{array} \right. \\ z \geq 0, \end{array} \right. \quad (6.4.1)$$

Les contraintes de reroutage peuvent être remplacées par une inégalité :

$$\omega^{e,k} x^{e,k} \geq \Pi_e^k x^k.$$

Les multiplicateurs β_e^k associés à ces contraintes sont alors négatifs.

Nous proposons de relâcher les contraintes de reroutage et de résoudre le problème

qui suit:

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K, x^{1,1}, \dots, x^{n,K}, z} \sum_{k=1}^K (c - \beta^k) \Pi^k x^k + \sum_{e=1}^n \sum_{k=1}^K \beta_e^k \omega^{e,k} x^{e,k} + d^T z; \\ \omega^k x^k = r^k, \\ x^k \geq 0, k = 1, \dots, K, \\ e \in E \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z, \\ x^{e,k} \geq 0, k = 0, \dots, K, \end{array} \right. \\ z \geq 0, \end{array} \right. \quad \begin{array}{l} (\alpha^k) \\ (s^k) \\ (\lambda^e) \\ (s^{e,k}) \\ (t) \end{array} \quad (6.4.2)$$

En relâchant les contraintes de reroutage, nous éliminons les contraintes qui lient les variables de routage à celles de reroutage.

Le problème (6.4.2) peut donc se décomposer en deux problèmes :

1. un problème de routage qui se décompose en K problèmes de flots de coût minimum :

$$\left\{ \begin{array}{l} \text{Min}_{x^k} (c - \beta^k) \Pi^k x^k; \\ \omega^k x^k = r^k, \\ x^k \geq 0. \end{array} \right. \quad (6.4.3)$$

Ce qui revient à résoudre K problèmes de plus court chemin.

Le coût unitaire $(c - \beta^k)$ reste positif puisque le vecteur $-\beta^k$ et c le sont.

La résolution de cette première partie du problème (6.4.2) ne demande aucun calcul supplémentaire si les multiplicateurs β^k ne sont pas perturbés. En effet, l'algorithme calcule ces plus courts chemins à chaque itération pour générer les nouveaux chemins. Mais si la procédure de perturbation est activée, l'algorithme calcule de nouveau K plus courts chemins (6.4.3).

2. un problème dont les contraintes couplent les variables de reroutage.

$$\left\{ \begin{array}{l} \text{Min}_{x^{1,1}, \dots, x^{n,K}, z} \sum_{e=1}^n \sum_{k=1}^K \beta_e^k \omega^{e,k} x^{e,k} + d^T z; \\ e \in E \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z, \\ x^{e,k} \geq 0, k = 0, \dots, K, \end{array} \right. \\ z \geq 0, \end{array} \right. \quad (\lambda^e) \quad (6.4.4)$$

Le problème (6.4.4) est homogène. Suivant que le problème dual soit réalisable ou non, la valeur optimale de (6.4.4) est nulle ou infinie. En effet, la valeur optimale de (6.4.4) est nulle si et seulement si le problème dual est réalisable. Explicitons le problème dual à (6.4.4) :

$$\left\{ \begin{array}{l} \text{Max}_{\lambda^1, \dots, \lambda^n} 0; \\ \epsilon \in E \left\{ \begin{array}{l} \text{(i.e)} \lambda^\epsilon \geq 0, \\ \text{(ii.e.1)} \omega^{\epsilon,1T} \beta_\epsilon^1 + \Pi^{\epsilon,1T} \lambda^\epsilon \geq 0 \\ \vdots \\ \text{(ii.e.K)} \omega^{\epsilon,KT} \beta_\epsilon^K + \Pi^{\epsilon,KT} \lambda^\epsilon \geq 0 \\ \text{(iii)} d - \sum_{\epsilon=1}^n \hat{\Delta}_\epsilon^T \lambda^\epsilon \geq 0, \end{array} \right. \end{array} \right. \quad (6.4.5)$$

D'après le système d'optimalité (6.2.5) de PSG, nous avons la positivité de $((\omega^{\epsilon,kT} \beta_\epsilon^k + \Pi^{\epsilon,kT} \lambda^\epsilon), (d - \sum_{\epsilon=1}^n \hat{\Delta}_\epsilon \lambda^\epsilon), \lambda^\epsilon)$ à l'optimum du programme maître. Le problème dual (6.4.5) est alors réalisable mais ceci n'est pas garanti si l'algorithme calcule une solution approchée du programme maître. Pour avoir une estimation inférieure (finie) nous proposons de perturber d'abord les multiplicateurs λ^ϵ de façon à avoir $d^T - \sum_{\epsilon=1}^n \lambda^{\epsilon T} \hat{\Delta}_\epsilon$ positif et ensuite les multiplicateurs β^k pour que $\beta_\epsilon^k \omega^{\epsilon,k} + \lambda^{\epsilon T} \Pi^{\epsilon,k}$ soit positif.

Le calcul de l'estimation inférieure se réduit alors à la recherche de K plus courts chemins (6.4.3).

6.4.2 Méthode de perturbation des multiplicateurs

Nous supposons que le programme maître fournit les multiplicateurs $\lambda^1, \dots, \lambda^n, \beta_1^k, \dots, \beta_n^k$. Ces valeurs ne sont pas forcément optimales puisque le programme maître calcule une solution approchée. Pour calculer un point réalisable du problème dual (6.4.5), nous proposons de perturber les multiplicateurs obtenus par le programme maître. Pour satisfaire les contraintes (6.4.5.i.e), nous projetons les λ^ϵ sur \mathbb{R}_+^n :

$$\lambda^\epsilon = [\lambda^\epsilon]^+.$$

Nous introduisons le vecteur L de \mathbb{R}^n , défini par :

$$L = d / \left(\sum_{\epsilon=1}^n \hat{\Delta}_\epsilon \lambda^\epsilon \right).$$

Nous désignons par L_m la composante minimale du vecteur L : $L_m = \min_i(l_i)$. En prenant $\lambda^e := L_m \lambda^e$, nous garantissons la vérification des contraintes (6.4.5iii). Pour tout flot $k \in \mathcal{K}$, et pour toute panne e affectant ce flot nous calculons un plus court chemin de réserve $\tilde{\chi}^{e,k}$ avec λ^e comme coût unitaire sur les arcs. En choisissant $\beta_e^k = \lambda^{eT} \tilde{\chi}^{e,k}$, les contraintes (6.4.5.ii.e.k) sont vérifiées. De plus, ce choix de multiplicateurs β_e^k donne la meilleure estimation inférieure utilisant les informations fournies par le programme maître (6.4.3). En effet, pour obtenir une bonne estimation inférieure, nous devons choisir β_e^k le plus petit possible en assurant la positivité de $\beta_e^k \omega^{e,k} + \lambda^{eT} \Pi^{e,k}$. C'est exactement ce que vérifie les β_e^k que nous avons choisis, puisque toutes les contraintes (6.4.5iii) sont actives.

Grâce à cette méthode de perturbation des multiplicateurs, nous pouvons associer une estimation inférieure au coût pour toute itération mineure (interne).

6.5 Calcul d'une estimation supérieure

La solution du problème maître (6.2.6) fournit un point réalisable de PSG (6.2.4) et donc une estimation supérieure de sa valeur optimale. Dans le cas où le programme maître ne fournit qu'une solution approchée (cas de DP1 et DP2), le calcul d'une estimation supérieure dépendra de l'algorithme choisi. Il faut exploiter les informations fournies par la solution approchée du programme maître pour trouver un point réalisable pour PSG (6.2.4). Dans les chapitres suivants nous proposons quatre algorithmes pour résoudre le programme maître. Les deux premiers sont basés sur les techniques proximales et les deux autres utilisent des méthodes de points intérieurs. Les deux premiers algorithmes nécessitent une méthode particulière de calcul de l'estimation supérieure. Cette méthode est introduite au chapitre suivant (garanties d'optimalité).

6.6 Formulation sommets-arcs de PSG

Nous décrivons dans cette section, la formulation sommets-arcs de PSG. Nous notons A la matrice d'incidence sommets-arcs associée au graphe $G(V, E)$. Dans la formulation sommets-arcs, chaque flot k est caractérisé par un vecteur de \mathbb{R}^n (n est le nombre d'arcs) que nous notons encore x^k . Une composante x_i^k , $i \in \{1, \dots, n\}$ représente la quantité du flot k transporté par l'arc i . Cet arc i peut être le croisement de plusieurs chemins liant la paire origine-destination (o^k, p^k) mais il est difficile de retrouver ces

chemins. Le routage des K flots traversant le réseau $(x^1, \dots, x^K) \in \mathbb{R}^{n \times K}$ est exprimé en terme de contribution sur les arcs du réseau.

Comme pour la formulation arcs-chemins, nous introduisons pour toute paire panne-flot (e, k) les variables $x^{e,k}$, qui sont dans ce cas des vecteurs de \mathbb{R}^n .

Dans (6.2.4) grâce à la méthode de génération de chemins, chaque programme maître se limite à des sous-ensembles de \mathcal{K}^e , $e \in \{1, \dots, n\}$. Ce qui a contribué à la réduction de la taille de PSG avec la formulation (6.2.3). La formulation sommets-arcs ne peut pas fournir la liste \mathcal{K}^e des flots affectés par une panne d'arc e donnée mais reconnaît la valeur de la fraction x_e^k à rerouter de tout flot k dans \mathcal{K} .

Nous considérons alors que tout flot $k \in \mathcal{K}$ est susceptible d'être touché par toute panne d'arc. La défaillance d'un arc e crée pour tout flot k une demande définie par la paire origine-destination (o^k, p^k) et sa valeur $r^{e,k} = x_e^k$. Suivant la valeur de $r^{e,k}$ à l'optimum, nous pouvons déduire l'appartenance ou pas de k à \mathcal{K}^e (si $r^{e,k} = 0$ alors $k \notin \mathcal{K}^e$). En choisissant la formulation sommets-arcs, nous devons prévoir $K \times n \times (n-1) + n^2 + n$ variables rien que pour le reroutage. Si K est de l'ordre de n , le nombre total de variables est de l'ordre de n^3 . Les demandes créées par une panne d'arc e peuvent être transportées par tous les arcs du réseau excepté l'arc e . Ce qui peut être traduit par

$$\hat{A}^e x^{e,k} = x_e^k \times \frac{b^k}{r^k},$$

avec $\hat{A}^e = (A^1 \ \dots \ A^{p-1} \ A^{p+1} \ \dots \ A^n)$, et A^i désigne la $i^{\text{ème}}$ colonne de la matrice d'incidence sommets-arcs A .

Le PSG s'écrit sous formulation sommets-arcs comme suit :

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K, z} c^T \sum_{k \in \mathcal{K}} x^k + d^T z \\ k \in \mathcal{K} \left\{ \begin{array}{ll} \text{(i.k)} Ax^k = b^k, & (\alpha^k) \\ \text{(ii.k)} x^k \geq 0, & (s^k) \end{array} \right. \\ e \in E \left\{ \begin{array}{ll} \text{(iii.e)} x^{e,0} + \sum_{k \in \mathcal{K}} \hat{A}^e x^{e,k} = \hat{\Delta}_e z, & (\lambda^e) \\ \text{(iv.e)} x^{e,0} \geq 0, & (s^{e,0}) \\ k \in \mathcal{K} \left\{ \begin{array}{ll} \text{(v.e.k)} \hat{A}^e x^{e,k} = x_e^k \times \frac{b^k}{r^k}, & (\beta^{k,e}) \\ \text{(vi.e.k)} x^{e,k} \geq 0, & (s^{e,k}) \end{array} \right. \\ \text{(vii)} z \geq 0, & (t) \end{array} \right. \end{array} \right. \quad (6.6.1)$$

Les multiplicateurs de Lagrange sont écrits en face des contraintes correspondantes. Nous rappelons que nous avons repris les notations de la formulation (6.2.4) mais

la matrice identité d'ordre n .

La matrice des contraintes possède $n+n^2 \times (K+1)$ colonnes et $n^2+K \times p \times (n+1)$ lignes mais est très creuse. Cependant la résolution de PSG sous formulation sommets-arcs (6.6.1) est possible bien qu'elle ne semble pas avantageuse.

6.7 Exemple de problèmes de sécurisation

Afin de valider et tester numériquement les algorithmes proposés pour résoudre PSG, nous avons programmé un générateur aléatoire de données `rdgene`. Les problèmes générés ont la particularité d'avoir un graphe complet, des coûts unitaires (nominal et de réserve) aléatoires positifs et des demandes entre chaque paire de nœuds de valeur aléatoire.

Le nombre d'arcs n et le nombre de flots K sont, dans le cas des problèmes générés par `rdgene`, égaux à $(p \times (p - 1))/2$ (p est le nombre de nœuds).

Si tous les arcs ont le même coût dans les réseaux nominal et de réserve ($c_i = c_j$, et $d_i = d_j$ pour toute paire d'arcs (i, j)) et si en plus toutes les demandes ont la même valeur ($r^k = r^l$, pour tous k et l dans \mathcal{K}), le réseau est dit symétrique (à ne pas confondre avec les graphes symétriques). Nous allons utiliser ce type de réseau pour tester, dans un premier temps, les algorithmes étudiés dans les prochains chapitres.

Soit un réseau symétrique défini par p nœuds et r^0 la valeur de toute demande qui le traverse. Tout arc a un coût nominal et un coût de réserve respectivement égaux à c^0 et d^0 , avec $d^0 > c^0$. Il est facile de vérifier qu'à l'optimum le vecteur capacité nominal $U = r^0 \mathbf{1}$ et celui de la capacité de réserve $z = \frac{r^0}{p-2} \mathbf{1}$.

Routage d'une demande du reseau

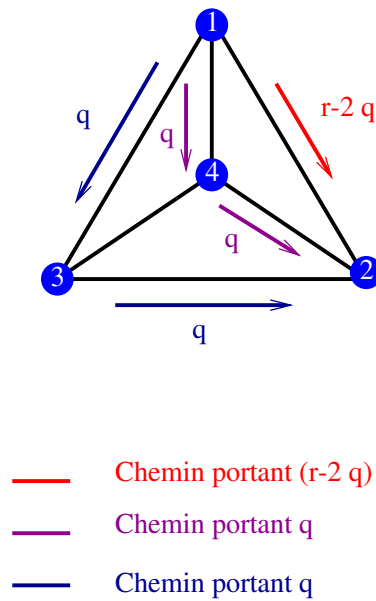


Figure 6.7.1: Routage du flot 1

Reroutage en cas de la panne 1

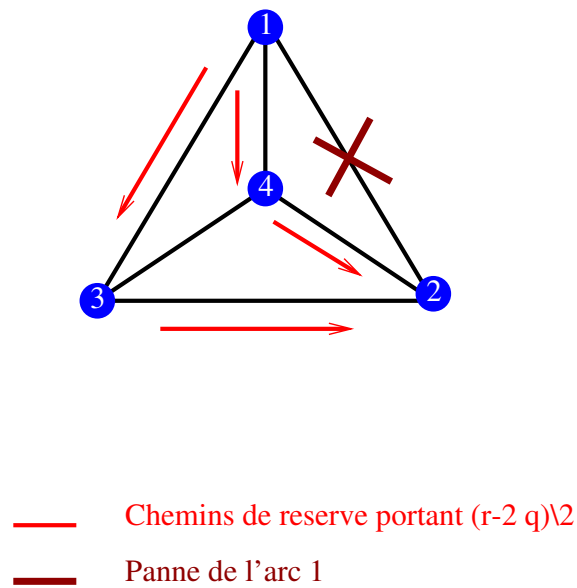
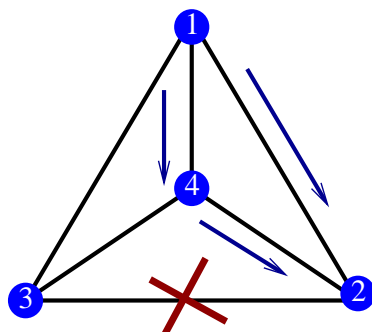


Figure 6.7.2: Reroutage du flot 1

Reroutage en cas de la panne 4





-  Chemins de reserve portant $q/2$
-  Panne de l'arc 4

Figure 6.7.3: Reroutage du flot 1

Reroutage en cas de la panne 5

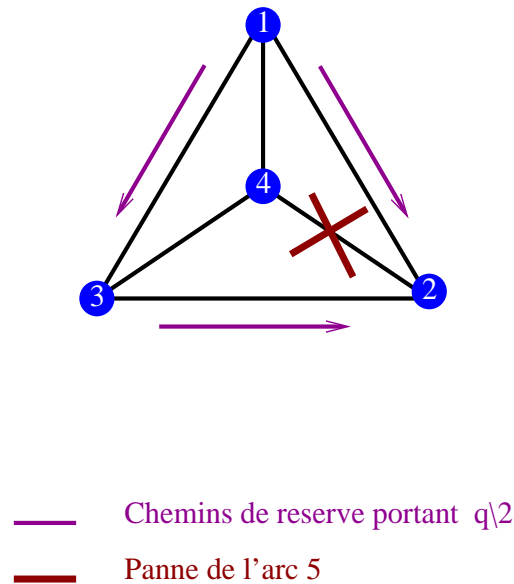


Figure 6.7.4: Reroutage du flot 1

6.8 Conclusion

Les performances de la formulation arcs-chemins dans le cas du problème de routage nous a conduit à nous intéresser uniquement à cette formulation. Nous avons posé et analysé le problème sous formulation arcs-chemins. La méthode de génération de colonnes (chemins) et les garanties de qualité de la solutions ont été également présentées dans le cadre de cette formulation. Nous avons tout de même consacré une section à la formulation sommets-arcs de PSG qui sera sûrement utile pour sa résolution directe. Un schéma de décomposition peut également être intégré dans cette formulation. Par exemple, la résolution d'une suite de sous-problèmes qui se limitent à des sous-ensembles de pannes ou d'arcs. Mais tenant compte des performances des deux formulations dans le cas d'un problème de routage, la formulation arcs-chemins s'est avérée de loin la meilleure. Dorénavant nous utilisons uniquement la formulation (6.2.4) de PSG.

7. Sécurisation par la méthode de décomposition proximale

Dans ce chapitre nous proposons deux stratégies de décomposition basées sur la méthode de décomposition proximale et la génération de chemins.

Dans ces deux approches la tâche principale consiste en la résolution d'un problème quadratique. La structure découplée de la matrice de base se prête à la mise en œuvre d'un algorithme de gradient réduit.

Nous adaptons aux deux stratégies les techniques de génération de chemins (nominaux et de réserve) et de calcul d'une estimation inférieure, décrites au chapitre précédent. Nous proposons également une estimation supérieure à la valeur optimale de PSG valable même en dehors de l'optimalité de la solution fournie par le programme maître (6.2.6). Nous commençons ce chapitre par une introduction de la méthode de décomposition proximale dans le cadre de l'optimisation convexe (section 1). La deuxième section est consacrée à la première stratégie de décomposition. Le développement de cette première stratégie nous amène à en proposer une deuxième (section 3). Le programme maître est résolu par DP1 ou DP2 selon que nous choisissons la première ou la deuxième stratégie de décomposition. Les algorithmes DP1 et DP2 découlent de l'application de la méthode de décomposition proximale au problème transformé respectivement par la première et la deuxième stratégie de décomposition.

Dans une quatrième section nous spécialisons aux deux algorithmes la méthode de génération de chemins et l'estimation inférieure proposées au chapitre précédent.

7.1 Méthode de décomposition proximale en optimisation convexe

7.1.1 Motivation

Dans ce chapitre nous utilisons une méthode basée sur la décomposition proximale sur le graphe d'un opérateur pour résoudre le problème suivant :

$$\text{Trouver } (x, y) \in \mathcal{A} \times \mathcal{A}^- \text{ tel que } y \in T(x) \quad (7.1.1)$$

où \mathcal{A} est un sous-espace vectoriel d'un espace de Hilbert de dimension finie \mathcal{H} et T un opérateur maximal monotone. La méthode de décomposition proximale sur le graphe d'un opérateur maximal monotone (MDP) a été introduite par Mahey, Oualibouch et Tao [91]. MDP est une spécialisation de la méthode de l'inverse partiel (MIP) aux problèmes convexes séparables. Nous rappelons que MIP a été proposée par Spingarn dans [125] pour la résolution de (7.1.1). La preuve de convergence de MDP fournie dans [91] n'utilise pas le concept de l'inverse partiel.

Il y'a plusieurs manières de transformer un problème séparable convexe sous la forme de (7.1.1). L'idée générale est de représenter le couplage entre des sous-systèmes par des sous-espaces de produit d'espaces correspondant aux copies des variables primales et duales.

L'algorithme procède à chaque itération en deux étapes distinctes :

1. une étape proximale qui régularise la fonction objectif en ajoutant un terme quadratique dépendant des solutions primale et duale de l'itéré précédent,
2. et une étape de projection sur les sous-espaces correspondants.

Dans [91], Mahey, Oualibouch et Tao donnent la valeur optimale du paramètre de la décomposition proximale dans le cas d'un opérateur strictement monotone. L'algorithme de décomposition proximale utilisé dans cette étude a été déjà proposé pour résoudre d'autres problèmes de télécommunications [22]. Dans [111], Ouorou, Mahey et Vial présentent une étude des algorithmes existants dans la littérature pour la résolution de problèmes convexes de multiflot. L'étude [111] était motivée par le problème de routage, avec un délai moyen de transit minimal, dans les réseaux de télécommunications. L'algorithme de décomposition proximale appliqué au problème

de multiflot strictement convexe de [111], donne des résultats satisfaisants en comparant avec ACCPM (Analytic Center Cutting Plane Method) [42].

Bien que l'algorithme de décomposition proximale soit efficace pour la résolution des problèmes de multiflot, nous allons voir qu'il est lent lorsque le nombre de pannes est très grand. Une des raisons de cette lenteur réside dans le fait que l'algorithme résout à chaque itération des sous-problèmes quadratiques de multiflot.

Dans le problème de routage tout comme dans le problème de sécurisation, les chemins associés aux flots ne sont pas uniques. La formulation arcs-chemins de ces problèmes impose l'intégration de la méthode de génération de colonnes (chemins). Par conséquent, les solutions avec un grand nombre de chemins sont moins convenables que celles qui utilisent un petit nombre de chemins. Sur ce point, l'algorithme de décomposition proximale est plus adapté pour trouver des solutions avec peu de chemins. La raison pour laquelle l'algorithme de décomposition proximale utilise un petit nombre de chemins peut être expliquée intuitivement par la procédure de la mise à jour des chemins qui tend à forcer plusieurs d'entre eux à zéro. Cette intuition est confirmée empiriquement par les expériences menées dans [111] ainsi que par nos résultats numériques.

7.1.2 Opérateur maximal monotone

Nous désignons par $Gr(T)$ le graphe de l'opérateur T , défini par

$$Gr(T) = \{(x, y) \in \mathcal{H} \times \mathcal{H} : y \in T(x)\}.$$

Nous désignons respectivement par $\langle \cdot, \cdot \rangle$ et $\|\cdot\|$, le produit scalaire et la norme associés à l'espace de Hilbert \mathcal{H} .

Définition 20. D'après Minty [100], un opérateur T est dit **monotone** si

$$\langle x - x', y - y' \rangle \geq 0 \text{ pour tout } x, x' \in \mathcal{H} \text{ et } y \in T(x), y' \in T(x').$$

Définition 21. L'opérateur T est **fortement monotone** s'il existe une constante $a > 0$ telle que $\langle x - x', y - y' \rangle \geq a\|x - x'\|$.

Définition 22. On dit qu'un opérateur monotone T est **maximal monotone** si son graphe n'est contenu dans aucun autre graphe d'opérateur monotone.

Nous notons I l'opérateur identité ($I(x) = x$).

Les opérateurs maximaux monotones jouissent de la propriété suivante prouvée par Minty [100] :

Propriété 1. Pour tout $x \in \mathcal{H}$ et $\Lambda > 0$ il existe un unique $J_{\Lambda T}(x) \in \mathcal{H}$ tel que $x - J_{\Lambda T}(x) \in \Lambda T(J_{\Lambda T}(x))$.

Ce qui équivaut à dire que $J_{\Lambda T} = (I + \Lambda T)^{-1}$ est univoque.

Définition 23. La fonction $J_{\Lambda T} : \mathcal{H} \rightarrow \mathcal{H}$ est la *fonction proximale* associée à T et Λ , appelée aussi *résolvante* de T associée à Λ .

Dans ce paragraphe nous n'avons donné qu'un petit aperçu sur la résolvante et ses propriétés. Pour plus de détails nous invitons le lecteur à consulter [110] et [19]. Plusieurs problèmes d'optimisation sont équivalents à trouver un zéro d'un opérateur maximal monotone.

Dans la section suivante, nous présentons un algorithme qui exploite les propriétés de la résolvante pour trouver un point x de \mathcal{H} tel que $0 \in T(x)$.

7.1.3 L'algorithme du point proximal

Nous considérons le problème suivant :

$$\text{Trouver } x \in \mathbb{R}^n \text{ tel que } 0 \in T(x). \quad (7.1.2)$$

L'opérateur T est supposé maximal monotone. Pour résoudre (7.1.2) l'algorithme du point proximal (APP) part d'un point quelconque x^0 et utilise la procédure itérative suivante :

$$x^{j+1} = J_{\Lambda T}(x^j). \quad (7.1.3)$$

Toute solution de (7.1.2) est un point fixe de $J_{\Lambda T}$.

La convergence de la méthode du point proximal a été prouvée par Rockafellar dans [122]. Récemment, Güler dans [63] résout une question ouverte posée par Rockafellar dans [122] en montrant la convergence de la suite x^j générée par l'algorithme de point proximal sous sa forme exacte avec un taux de convergence valable même dans le cas où la fonction ne possède pas de minimum. Un algorithme de type point proximal avec un meilleur taux de convergence a été proposé par Güler [64]. Plusieurs études d'extension et de généralisation de la méthode de point proximal ont été menées pour

améliorer sa convergence. Nous citons à titre d'exemple [65] et [69].

Un problème de minimisation d'une fonction f convexe propre semi-continue inférieurement sur un espace de Hilbert \mathcal{H} , peut se ramener à un problème de recherche d'un zéro d'un opérateur maximal monotone. En effet l'opérateur T est dans ce cas le sous-différentiel de f noté ∂f . De plus ∂f est maximal monotone dans le cas où $f : \mathcal{H} \rightarrow]-\infty, +\infty]$ est convexe, propre et semi-continue inférieurement [101]. Dans [68] Ibaraki et Fukushima appliquent un algorithme de point proximal primal dual aux problèmes de multiflot convexe. Dans le cas de critère séparable, l'algorithme proposé dans [68] ressemble à celui proposé par Chifflet, Mahey et Reynier dans [22].

7.1.4 L'algorithme de décomposition proximale

Nous nous intéressons à la résolution du problème (7.1.1) par la méthode de décomposition proximale.

Nous précisons que MIP est une généralisation de APP basée sur la notion de l'opérateur inverse partiel. Et nous rappelons que l'algorithme de décomposition proximale exposé dans ce paragraphe, a été introduit dans [91] comme un cas particulier de l'algorithme d'inverse partiel [125].

Commençons par définir le concept de décomposition proximale.

Définition 24. Nous appelons *décomposition proximale* de $z \in \mathcal{H}$ sur le graphe de T , l'unique couple (u, v) tel que $z = u + v$ et $v \in T(u)$.

L'unicité de la décomposition proximale découle du fait que l'opérateur T soit maximal.

$$\text{Nous avons } \begin{cases} u = (I + T)^{-1}(z), \\ v = (I + T^{-1})^{-1}(z). \end{cases}$$

L'algorithme de décomposition proximale (ADP), alterne décomposition proximale sur le graphe de T et projection sur $\mathcal{A} \times \mathcal{A}^-$. Nous énonçons dans la suite l'algorithme ADP :

Algorithme ADP

1. Choisir $(x^0, y^0) \in \mathcal{A} \times \mathcal{A}^-$, $j := 0$,

$$2. \quad u^j := (I + T)^{-1}(x^j + y^j),$$

$$v^j := x^j + y^j - u^j,$$

Si $(u^j, v^j) \in \mathcal{A} \times \mathcal{A}^-$ alors **Stop**.

$$3. \quad x^{j+1} := u^j_{/\mathcal{A}}, \quad y^{j+1} := v^j_{/\mathcal{A}^\perp};$$

$j := j + 1$, retour à **2**.

La convergence de l'algorithme ADP est prouvée dans [91] sans utiliser le concept d'inverse partiel.

Définition 25. Nous appelons *décomposition proximale avec facteur d'échelle* $\Lambda > 0$ de $(x, y) \in \mathcal{H} \times \mathcal{H}$ sur le graphe d'un opérateur maximal monotone T , l'unique couple $(u, v) \in \mathcal{H} \times \mathcal{H}$ tel que : $x + \Lambda y = u + \Lambda v$ et $(u, v) \in Gr(T)$.

La décomposition proximale avec facteur d'échelle Λ de $(x, y) \in \mathcal{H} \times \mathcal{H}$ peut aussi être définie comme étant le couple (u, v) tel que
$$\begin{cases} u = (I + \Lambda T)^{-1}(x + \Lambda y), \\ v = (I + \Lambda^{-1}T^{-1})^{-1}(\Lambda^{-1}x + y). \end{cases}$$

L'algorithme suivant est une version de l'algorithme ADP avec facteur d'échelle.

Algorithme ADP Λ

1. Choisir $(x^0, y^0) \in \mathcal{A} \times \mathcal{A}^-$, $j := 0$, $\Lambda > 0$, $\rho \in]0, 2[$.

$$2. \quad u^j := (I + \Lambda T)^{-1}(x^j + \Lambda y^j),$$

$$v^j := \frac{x^j + \Lambda y^j - u^j}{\Lambda},$$

Si $(u^j, v^j) \in \mathcal{A} \times \mathcal{A}^-$ alors **Stop**.

$$3. \quad x^{j+1} := \rho u^j_{/\mathcal{A}} + (1 - \rho)x^j, \quad y^{j+1} := \rho v^j_{/\mathcal{A}^\perp} + (1 - \rho)y^j;$$

$j := j + 1$, retour à **2**.

L'algorithme ADPA est sensible au choix du paramètre Λ en terme de vitesse de convergence. Une valeur de Λ peut être bonne ou mauvaise selon les données du problème à résoudre.

Mahey, Oualibouch et Tao ont prouvé que le paramètre optimal Λ est égal à L^{-1} dans le cas où l'opérateur T est fortement monotone de module L [91]. La convergence de la suite $\{(x^j, y^j)\}$, calculée par l'algorithme ADPA est dans ce cas linéaire avec une estimation de taux : $\sqrt{1 - \frac{2\Lambda\alpha}{(1 + \Lambda L)^2}}$, où α est le coefficient de Lipschitz.

7.1.5 Optimisation convexe avec ADP

Dans le cas où l'opérateur T est le sous-différentiel ∂f d'une fonction convexe, propre, semi-continue inférieurement f de $\mathcal{H} = \mathbb{R}^n$ à valeurs dans $] - \infty, +\infty]$, le problème (7.1.1) est équivalent au problème d'optimisation :

$$\begin{aligned} \text{Min}_x f(x) \\ x \in \mathcal{A} \end{aligned} \tag{7.1.4}$$

En effet, un vecteur x dans \mathbb{R}^n est solution optimale de (7.1.4) si et seulement si

$$0 \in \partial(\mathcal{X}_{\mathcal{A}} + f)(x), \text{ où } \mathcal{X}_{\mathcal{A}}(x) = \begin{cases} 0 & \text{si } x \in \mathcal{A}, \\ +\infty & \text{ailleurs} \end{cases}.$$

D'après le théorème 23.8 dans [121], $0 \in \partial\mathcal{X}_{\mathcal{A}}(x) + \partial f(x)$. Puisque $\partial\mathcal{X}_{\mathcal{A}}(x) = \mathcal{A}^-$, $x \in \mathcal{A}$ est une solution optimale de (7.1.4) s'il existe $y \in \mathcal{A}^-$ tel que $y \in \partial f(x)$. Le problème (7.1.4) est alors équivalent à trouver $x \in \mathcal{A}, y \in \mathcal{A}^- / y \in \partial f(x)$.

Rappelons que ∂f est maximal monotone dans le cas où f est convexe propre et s.c.i. L'étape proximale (correspond à l'étape **2** dans l'algorithme ADPA) s'écrit dans le cas de l'optimisation convexe comme suit : $0 \in -x^j - \Lambda y^j + (I + \Lambda \partial f)u^j$.

L'étape proximale revient donc à trouver u^j qui minimise $f(u) + \frac{1}{2\Lambda} \|u - (x^j + \Lambda y^j)\|^2$. L'algorithme de la décomposition proximale dans le cas de la minimisation d'une fonction convexe propre s.c.i sur un sous-espace fermé de \mathbb{R}^n procède comme suit :

Algorithme ADPOC

1. Choisir $(x^0, y^0) \in \mathcal{A} \times \mathcal{A}^-$, $j := 0$, $\Lambda > 0$, $\rho \in]0, 2[$.
2. $u^j = \operatorname{argmin}_u f(u) + \frac{1}{2\Lambda} \|u - (x^j + \Lambda y^j)\|^2$,

$$v^j := \frac{x^j + \Lambda y^j - u^j}{\Lambda},$$

Si $(u^j, v^j) \in \mathcal{A} \times \mathcal{A}^-$ alors **Stop**.

$$3. \quad x^{j+1} := \rho u_{/\mathcal{A}}^j + (1 - \rho)x^j, \quad y^{j+1} := \rho v_{/\mathcal{A}^\perp}^j + (1 - \rho)y^j;$$

$j := j + 1$, retour à **2**.

L'algorithme ADPOC reste convergent même si l'étape proximale calcule une solution approchée u^j avec une précision α^j vérifiant $\sum_j \alpha_j < +\infty$. Autrement dit :

$$u^j \in \alpha_j - \operatorname{argmin}_u f(u) + \frac{1}{2\Lambda} \|u - (x^j + \Lambda y^j)\|^2, \text{ et } \sum_j \alpha_j < +\infty.$$

Si $f(u) = \sum f_i(x_i)$, alors le calcul de l'étape **2** se décompose en plusieurs problèmes indépendants (pour plus de détails voir [110]).

Le facteur de relaxation ρ peut varier d'une itération à l'autre, avec

$$2 > \sup_j \rho^j \geq \inf_j \rho^k > 0.$$

Il permet de stabiliser la méthode.

7.2 Première stratégie de décomposition (DP1)

Dans ce paragraphe nous proposons une stratégie de décomposition motivée par la structure particulière de PSG (6.2.4). La complexité de ce problème réside dans les couplages qui existent entre les variables de routage et celles du reroutage, entre les pannes et entre les flots. La stratégie choisie doit "réduire" le nombre de niveaux de couplage pour simplifier un peu le modèle.

En observant (6.2.4) ou la matrice des contraintes associée au problème maître (6.2.6), nous pouvons déduire que le routage $x = (x^1, \dots, x^k)$ et le vecteur de capacité de réserve z sont des variables couplantes. Nous copions ces variables autant de fois qu'elles sont susceptibles de coupler. Chaque copie de z correspond à une panne e et sera notée $z(e)$. Les contraintes de capacité de réserve associées à toute panne e ne concernent que la copie $z(e)$ maintenant.

Pour les copies de routage le choix des copies et des contraintes associées est moins évident. D'après la formulation (6.2.4) le routage $x = (x^1, \dots, x^k)$ doit satisfaire deux

types de contraintes : de routage (6.2.4.i.k) et de reroutage (6.2.4.v.e.k). Dans un premier temps nous proposons d'attribuer le premier type de contraintes à une seule copie du routage $x(0) = (x^1(0), \dots, x^k(0))$, que nous appellerons **copie principale**. Toute autre copie du routage $x(e) = (x^1(e), \dots, x^k(e))$, $e \in \{1, \dots, n\}$ sera associée à une panne e et satisfera des contraintes de reroutage (6.2.4.v.e.k) uniquement (et bien sûr les contraintes de positivité (6.2.4.vi.e.k)).

Des contraintes d'égalité de ces copies sont ajoutées au modèle pour avoir l'équivalence entre cette nouvelle formulation et (6.2.4) de PSG.

Faisons le bilan de notre premier choix de copies. Seulement la copie principale du routage intervient dans les contraintes de routage (6.2.4.i.k) (et de positivité (6.2.4.ii.k)). Les contraintes de capacité de réserve (6.2.4.iii.e) ne concernent que la copie $z(e)$ et les variables associées à cette panne $(x^{e,0}, x^{e,1}, \dots, x^{e,K})$. Toute copie $x(e)$ du routage n'intervient que dans les contraintes de reroutage (6.2.4.v.e.k) et celles de positivité (6.2.4.ii.k). Par conséquent, en oubliant l'égalité des différentes copies, nous avons éliminé les couplages entre les pannes et entre routage-reroutage. Les seules contraintes couplantes sont les copies des contraintes de capacité de réserve (6.2.4.iii.e). Nous allons voir dans la suite que ce couplage par flot n'est pas très important, puisque seulement les flots affectés par la panne e interviennent dans (6.2.4.iii.e).

Pour finir la modélisation de cette première stratégie de décomposition, nous devons donner des prix aux copies. L'importance d'une panne par rapport à une autre peut être mesurée en terme de dominance.

Une panne est dite dominante si toutes les contraintes de capacité (6.2.4.iii.e) sont actives. L'idéal est de répartir d sur les copies de z en fonction de l'importance de la panne (La panne la plus dominante est la plus chère). A priori nous n'avons aucune information sur la dominance des pannes. Nous nous sommes alors contentés de répartir le prix unitaire d uniformément sur toutes les copies de z .

Concernant les copies du routage, nous avons choisi de ne payer que la copie principale.

En suivant cette stratégie de décomposition, PSG se formule comme suit :

$$\left\{ \begin{array}{l}
\text{Min}_{x^1(0), \dots, x^K(n), x^{1,0}, \dots, x^{n,K}, z(1), \dots, z(n)} c^T \sum_{k=1}^K \Pi^k x^k(0) + \sum_{e=1}^n D^T z(e); \\
k \in \mathcal{K} \left\{ \begin{array}{l} \omega^k x^k(0) = r^k, \quad (\alpha^k) \\ x^k(0) \geq 0, \quad (s^k(0)) \end{array} \right. \\
e \in E \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{\epsilon,k} x^{\epsilon,k} = \hat{\Delta}_e z(e), \quad (\lambda^e) \\ x^{e,0} \geq 0, \quad (s^{e,0}) \\ z(e) \geq 0, \quad (s_z(e)) \\ k \in \mathcal{K} \left\{ \begin{array}{l} \omega^{\epsilon,k} x^{\epsilon,k} = \Pi_e^k x^k(e), \quad (\beta_e^k) \\ x^k(e) \geq 0, \quad (s^k(e)) \\ x^{\epsilon,k} \geq 0, \quad (s^{\epsilon,k}) \end{array} \right. \\
z(1) = \dots = z(n), \quad (\tilde{t}) \\
x(0) = x(1) = \dots = x(n), \quad (\tilde{y})
\end{array} \right. \quad (7.2.1)$$

où

- $D_i = \frac{d_i}{n-1}$
- $s^k(0)$ est la variable duale à la copie principale du routage.
- pour $e = 1, \dots, n$,
 - $s_z(e)$ est la variable duale à la $e^{\text{ème}}$ copie de la capacité de réserve.
 - $s^k(e)$ est la variable duale à la copie $x^k(e)$ du routage de la demande k , pour $k = 1, \dots, K$.
- $\tilde{y} = (\tilde{y}(0), \dots, \tilde{y}(n))$ est le multiplicateur associé à la contrainte d'égalité des copies du routage.
- $\tilde{t} = (\tilde{t}(1), \dots, \tilde{t}(n))$ est le multiplicateur associé à la contrainte d'égalité des copies de capacité de réserve.

Les techniques proximales exploitent avantageusement la structure du modèle (7.2.1) du problème de sécurisation globale.

7.2.1 Résolution du problème de sécurisation globale par la méthode de décomposition proximale

La formulation (7.2.1) de PSG (6.2.4) peut aussi s'écrire comme suit :

$$\left\{ \begin{array}{l} \text{Min}_{x(0), \dots, x(n), x^{1,0}, \dots, x^{n,K}, z(1), \dots, z(n)} f_0(x(0)) + \sum_{i=1}^n f_i(x(i), x^i, z(i)); \\ z(1) = \dots = z(n), \quad (\tilde{t}) \\ x(0) = x(1) = \dots = x(n), \quad (\tilde{y}) \end{array} \right. \quad (7.2.2)$$

avec

•

$$f_0(x(0)) = \begin{cases} c^T \sum_{k=1}^K \Pi^k x^k(0); & \text{si pour } k \in \mathcal{K} \begin{cases} \omega^k x^k(0) = r^k, \\ x^k(0) \geq 0, \end{cases} \\ +\infty & \text{sinon.} \end{cases} \quad (7.2.3)$$

• et pour toute panne d'arc e :

$$f_e(x(e), x^e, z(e)) = \begin{cases} D^T z(e); & \text{si } \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z(e), \\ x^{e,0} \geq 0, \\ z(e) \geq 0, \\ k \in \mathcal{K} \begin{cases} \omega^{e,k} x^{e,k} = \Pi_e^k x^k(e), \\ x^k(e) \geq 0, \\ x^{e,k} \geq 0. \end{cases} \end{array} \right. \\ +\infty & \text{sinon.} \end{cases} \quad (7.2.4)$$

Les fonctions f_i , $i \in \{0, \dots, n\}$ sont clairement indépendantes.

Posons $X^0 = (x(0), \dots, x(n))$ et $Z = (z(1), \dots, z(n))$ les vecteurs des copies du routage et de la capacité de réserve. A chaque panne $e \in \{1, \dots, n\}$ nous associons un vecteur de reroutage $X^e = (x^{e,1}, \dots, x^{e,K}, x^{e,0})$. Le vecteur (X^0, \dots, X^n, Z) appartient à l'espace

$$\mathcal{H} = \mathbb{R}^{(n+1) \times n_n} \times \mathbb{R}^{n_r + n^2} \times \mathbb{R}^{n^2},$$

où n_n est le nombre total de chemins nominaux et n_r celui des chemins de réserve. Le sous-espace réalisable pour (7.2.2) sera noté \mathcal{A} :

$$\mathcal{A} = \{(X, Z) \in \mathcal{H}; x(0) = \dots = x(n) \text{ et } z(1) = \dots = z(n)\}.$$

Chaque itération j de la méthode de décomposition proximale (Algorithme ADPOC) se fait en deux parties :

1. Une première partie de minimisation sans la contrainte $(X, Z) \in \mathcal{A}$. Cette partie correspond à l'étape proximale (étape **2** de l'algorithme ADPOC). Nous verrons que le vecteur solution $u^j = (u(0)^j, \dots, u(n)^j, u^1^j, \dots, u^{n_j}, u_z(1)^j, \dots, u_z(n)^j)$ est obtenu par résolution de $(n + 1)$ sous-problèmes indépendants.
2. Une deuxième partie de projection sur \mathcal{A} et \mathcal{A}^\perp . Elle correspond à l'étape **3** de l'algorithme ADPOC. A chaque étape de projection, l'algorithme de décomposition proximale calcule les valeurs des vecteurs X^{j+1} , Y^{j+1} , Z^{j+1} et T^{j+1} définis par :

$$\begin{cases} (X^{j+1}, Z^{j+1}) = P_{/\mathcal{A}}(u^j), \\ (Y^{j+1}, T^{j+1}) = P_{/\mathcal{A}^\perp}(v^j), \end{cases} \quad (7.2.5)$$

avec

$$v^j = \frac{(X^j, Z^j) + \Lambda(Y^j, T^j) - u^j}{\Lambda}.$$

Nous détaillons la spécialisation de ces deux étapes à PSG (7.2.2) dans les sous-sections qui suivent.

Étape proximale

En appliquant l'algorithme ADPOC au problème (7.2.2), l'étape proximale revient à calculer

$$u^j = \operatorname{argmin}_u f_0(u(0)) + \sum_{\epsilon=1}^n f_\epsilon(u(\epsilon), u^\epsilon, u_z(\epsilon)) + \frac{1}{2\Lambda} \|u - ((X, Z)^j + \Lambda(Y, T)^j)\|^2,$$

avec f_0 et f_ϵ définies respectivement par (7.2.3) et (7.2.4). Le terme quadratique ajouté à la somme des fonctions f_i est séparable et n'affecte pas l'indépendance de ces fonctions.

Le problème traité à l'étape proximale,

$$\left\{ \begin{array}{l} \text{Min}_{u^1(0), \dots, u^K(n), u^{1,0}, \dots, u^{n,K}, u_z(1), \dots, u_z(n)} c^T \sum_{k=1}^K \Pi^k u^k(0) + \sum_{e=1}^n D^T u_z(e) + \\ \frac{1}{2\Lambda} \sum_{e=0}^n \|u(e) - (x(e)^j + \Lambda y^k(e)^j)\|^2 + \\ \frac{1}{2\Lambda} \sum_{e=1}^n \|u_z(e) - (z(e)^j + \Lambda t(e)^j)\|^2; \\ k \in \mathcal{K} \left\{ \begin{array}{l} \omega^k u^k(0) = r^k, \\ u^k(0) \geq 0, \end{array} \right. \quad (\alpha^k) \\ \\ e \in E \left\{ \begin{array}{l} u^{e,0} + \sum_{k=1}^K \Pi^{e,k} u^{e,k} = \hat{\Delta}_e u_z(e), \\ u^{e,0} \geq 0, \\ u_z(e) \geq 0, \end{array} \right. \quad (\lambda^e) \star \\ \\ k \in \mathcal{K} \left\{ \begin{array}{l} \omega^{e,k} u^{e,k} = \Pi_e^k u^k(e), \\ u^k(e) \geq 0, \\ u^{e,k} \geq 0, \end{array} \right. \quad (\beta_e^k) \end{array} \right. \quad (7.2.6)$$

peut s'écrire comme une suite de $(n + 1)$ problèmes indépendants. Le premier sous-problème concerne la fonction f_0 de la copie principale du routage $u(0)$:

$$\left\{ \begin{array}{l} \text{Min}_{u(0)} \sum_{k=1}^K c^T \Pi^k u^k(0) + \frac{1}{2\Lambda} \|u^k(0) - (x^k(0)^j + \Lambda y^k(0)^j)\|^2; \\ k \in \mathcal{K} \left\{ \begin{array}{l} \omega^k u^k(0) = r^k, \\ u^k(0) \geq 0, \end{array} \right. \end{array} \right. \quad (7.2.7)$$

Chaque sous-problème des n suivants est associé à une panne e et met en jeu les variables de reroutage $(u^{e,0}, \dots, u^{e,K})$ et les $e^{\text{ème}}$ copies de la capacité de réserve

$u_z(\epsilon)$ et du routage $u(\epsilon)$:

$$\left\{ \begin{array}{l} \text{Min}_{u(\epsilon), u^\epsilon} D^T(u^{\epsilon,0} + \sum_{k=1}^K \Pi^{\epsilon,k} u^{\epsilon,k}) + \phi(u(\epsilon), u^\epsilon); \\ u^{\epsilon,0} \geq 0, \\ k \in \mathcal{K} \left\{ \begin{array}{l} \omega^{\epsilon,k} u^{\epsilon,k} = \Pi_e^k u^k(\epsilon), \\ u^k(\epsilon) \geq 0, \\ u^{\epsilon,k} \geq 0, \end{array} \right. \end{array} \right. \quad (7.2.8)$$

où

$$\begin{aligned} \phi(u(\epsilon), u^\epsilon) &= \frac{1}{2\Lambda} \left[\|u(\epsilon) - (u^j(\epsilon) + \Lambda y^j(\epsilon))\|^2 \right. \\ &\quad \left. + \|u^{\epsilon,0} + \sum_{k=1}^K \Pi^{\epsilon,k} u^{\epsilon,k} - (z^j(\epsilon) + \Lambda t^j(\epsilon))\|^2 \right]. \end{aligned}$$

Dans cette formulation de la fonction f_ϵ , nous avons exprimé la copie de la capacité de réserve $z(\epsilon)$ en fonction des variables de reroutage u^ϵ (7.2.6.★). La composante $u_z(\epsilon)_\epsilon$ pose un problème parce qu'elle n'intervient pas dans les contraintes de capacité ($\hat{\Delta}_\epsilon u_z = u^{\epsilon,0} + \sum_{k=1}^K \Pi^{\epsilon,k} u^{\epsilon,k}$). Mais puisqu'elle doit être nulle (ϵ en panne), nous pouvons remplacer $u_z(\epsilon)$ par $\hat{\Delta}_\epsilon u_z$ et répartir le prix unitaire de réserve d sur les $(n-1)$ vraies copies de $(u_z)_\epsilon$. Ceci explique le choix de $D = d/(n-1)$. Nous précisons que le sous-espace \mathcal{A} sera modifié pour tenir compte de ce changement (détails à l'étape de projection).

Étape de Projection

A cette étape l'algorithme ADPOC projette la solution obtenue à l'étape proximale $u^j = (u(0)^j, \dots, u(n)^j, u^{1j}, \dots, u^{nj}, u_z(1)^j, \dots, u_z(n)^j)$ sur le sous-espace \mathcal{A} et son orthogonal \mathcal{A}^- pour calculer respectivement (X^{j+1}, Z^{j+1}) et (Y^{j+1}, T^{j+1}) . Puisque

$$u^j + \Lambda v^j = (X^{j+1}, Z^{j+1}) + \Lambda(Y^{j+1}, T^{j+1}), \quad (7.2.9)$$

l'algorithme calcule uniquement le projeté de u sur \mathcal{A} .

Le sous-espace \mathcal{A} doit être modifié pour tenir compte des changements apportés aux fonctions f_1, \dots, f_n . Pour tout $\epsilon \in \{1, \dots, n\}$, nous avons éliminé la copie de la capacité de réserve de l'arc en panne $(u_z(\epsilon))_\epsilon$ parce qu'elle est trivialement nulle. La

contrainte d'égalité des copies de la $e^{\text{ème}}$ composante de capacité de réserve doit être remplacée par

$$(z(1))_e = \cdots = (z(e-1))_e = (z(e+1))_e = \cdots = (z(n))_e.$$

Par conséquent

$$\mathcal{A} = \{(X, Z) \in \mathcal{H}; x(0) = \cdots = x(n) \text{ et pour tout } e \in \{1, \dots, n\}, \\ (z(1))_e = \cdots = (z(e-1))_e = (z(e+1))_e = \cdots = (z(n))_e \text{ et } (z(e))_e = 0\}.$$

La projection sur le sous-espace \mathcal{A} n'est autre qu'un calcul de moyenne. Nous avons

$$\begin{cases} x^{j+1}(i) = \frac{\sum_{e=0}^n u^j(e)}{n+1} \text{ pour tout } i \in \{0, \dots, n\}. \\ z^{j+1}(i) = \frac{\sum_{e=1}^n u_z^j(e)}{n-1} \text{ pour tout } i \in \{1, \dots, n\}. \end{cases} \quad (7.2.10)$$

Pour calculer (X^{j+1}, Z^{j+1}) , l'algorithme a besoin uniquement de deux vecteurs $varx$ et $varz$ respectivement de taille n_n et n . A chaque résolution d'un sous-problème 7.2.8, l'algorithme met à jour ces vecteurs en rajoutant la copie du routage $u^j(e)$ à $varx$ et la copie de la capacité de réserve $u_z^j(e)$ à $varz$.

Mais si l'algorithme utilise la relation (7.2.9) pour calculer (Y^{j+1}, T^{j+1}) , toutes les copies de routage et de la capacité de réserve doivent être stockées. Ce qui risque de saturer rapidement la mémoire même pour de petits réseaux. Pour éviter de garder toutes ces copies en mémoire nous utilisons la technique suivante.

Comme $u^j + \Lambda v^j = (X^j, Z^j) + \Lambda(y^j, t^j)$, la projection de cette égalité sur \mathcal{A}^- donne la relation suivante:

$$\Lambda(Y^{j+1}, T^{j+1}) - (X^{j+1}, Z^{j+1}) = \Lambda(Y^j, T^j) - (u(0)^j, \dots, u(n)^j, u_z). \quad (7.2.11)$$

Pour tout $e \in \{1, \dots, n\}$, le vecteur $\Lambda y^{j+1}(e) - x^{j+1}(e)$ (resp $\Lambda t^{j+1}(e) - z^{j+1}(e)$) s'exprime en fonction de l'itéré précédent $y^j(e)$ (resp $t^j(e)$) et de la solution courante $u^j(e)$ (resp $u_z^j(e)$).

A chaque résolution d'un sous-problème (7.2.8), l'algorithme garde alors en mémoire les vecteurs $\Lambda y^{j+1}(e) - x^{j+1}(e)$ et $\Lambda t^{j+1}(e) - z^{j+1}(e)$ calculés grâce à la relation (7.2.11). A la fin de l'étape proximale l'algorithme calcule les valeurs des projetés sur \mathcal{A} , $x^{j+1}(e)$ et $z^{j+1}(e)$ (7.2.10), et peut déduire les valeurs des projetés sur \mathcal{A}^- $y^{j+1}(e)$ et $t^{j+1}(e)$. Au lieu de garder tout le vecteur des copies en mémoire $[(n-1)$ ou $(n+1)$ fois la

taille d'une copie], l'algorithme conserve la moyenne des copies de l'itéré précédent. Ce qui permet d'utiliser les mêmes tableaux pour la résolution des sous-problèmes (7.2.8). Pour plus de détails sur l'implémentation de cette étape et de gestion de la base de données nous renvoyons le lecteur à l'annexe de ce mémoire.

7.2.2 Étape proximale (a)

En observant le sous-problème (7.2.7), nous remarquons que, grâce à l'absence des contraintes de capacité nominale, nous n'avons plus de couplage entre les flots. Nous rappelons que la disparition de ces contraintes couplantes est liée à notre choix de la gestion de la capacité pour le reroutage (voir chapitre précédent).

Le sous-problème (7.2.7) peut donc être considéré comme K problèmes indépendants de flot simple. De plus, à tout flot k , nous associons un problème quadratique avec une seule contrainte linéaire et des contraintes de positivité. La résolution de ce dernier est très spécifique [66]. Pour l'exposer, nous allons considérer une version simplifiée des problèmes de flots simples pris dans (7.2.7) en omettant les indices de flots k , d'itération j et de copie (0). Il s'agit alors de :

$$\begin{cases} \text{Min}_{u \in \mathbb{R}^m} C^T u + \frac{1}{2\Lambda} \|u - \bar{x}\|^2; \\ \omega u = r, & (\alpha) \\ u \geq 0, & (v) \end{cases} \quad (7.2.12)$$

où α est le multiplicateur associé à la contrainte de routage, ω est le vecteur ligne unité de taille m , $C^T = c^T \Pi$ et $\bar{x} = (x + \Lambda y)$.

D'après le théorème de Kuhn-Tucker [84], u est solution de (7.2.12) si et seulement si il existe des multiplicateurs α et v tels que pour tout $i \in \{1, \dots, m\}$:

$$\begin{cases} \Lambda \alpha + u_i + \Lambda c_i - \bar{x}_i = \Lambda v_i, \\ v_i \geq 0, \\ u_i v_i = 0. \end{cases} \quad (7.2.13)$$

En posant

$$\begin{cases} u_i(\alpha) = \max(-\Lambda \alpha + \bar{x}_i - \Lambda c_i, 0), \\ v_i(\alpha) = \max(\alpha + c_i - \frac{\bar{x}_i}{\Lambda}, 0), \end{cases} \quad (7.2.14)$$

les conditions de Kuhn-Tucker (7.2.13) sont satisfaites pour $u = u(\alpha)$ et $v = v(\alpha)$ et nous avons $u \geq 0$. Il suffit alors de trouver la valeur de α^* pour laquelle $\omega u(\alpha^*) = r$

et nous obtenons la solution. La fonction $\omega u(\alpha) = \sum_{i=1}^m u_i(\alpha)$ est décroissante, linéaire par morceaux et atteint des valeurs négatives.

Supposons que les composantes du vecteur $\tilde{x} = \frac{\bar{x}}{\Lambda} - c$ sont ordonnées dans l'ordre croissant $\tilde{x}_1 \leq \tilde{x}_2 \leq \dots \leq \tilde{x}_m$. Les valeurs de α qui délimitent les morceaux linéaires de la fonction $\sum_{i=1}^m u_i(\alpha)$ sont $\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_m$.

Pour $l \in \{1, \dots, m\}$, nous avons

$$\sum_{i=1}^m u_i(\tilde{x}) = \sum_{i=l+1}^m (-\Lambda \tilde{x}_l + \bar{x}_i - \Lambda C_i).$$

Puisque $\omega u(\alpha^*) = r$, pour tout $\tilde{x}_l < \alpha^*$, nous avons $u_l(\alpha^*) = 0$ et $\sum_{i=l+1}^m (-\Lambda \tilde{x}_l + \bar{x}_i - \Lambda C_i) > r$.

Pour l_* solution de

$$l_* := \underset{l \in \{1, \dots, m\}}{\text{Max}} \ l; \tag{7.2.15}$$

$$\frac{\tilde{x}_{l+1} + \dots + \tilde{x}_m - r}{m - l} > \tilde{x}_l.$$

Le multiplicateur solution α^* doit vérifier :

$$\sum_{i=l_*+1}^m (-\Lambda \alpha^* + \bar{x}_i - \Lambda C_i) = r. \tag{7.2.16}$$

7.2.3 Étape proximale (b)

La deuxième phase de l'étape proximale est consacrée aux n problèmes (7.2.8). Pour toute panne e , la résolution de ce problème donne la $e^{\text{ème}}$ copie du routage $u(e)$ et le reroutage des demandes créées par la panne e , u^e .

Les sous-problèmes (7.2.8) ont une structure très particulière. La matrice de contraintes associée à une panne e est bloc-diagonale. De plus, chaque bloc correspond à une seule contrainte linéaire : la contrainte de reroutage d'un des flots. Cette contrainte de reroutage n'a aucun sens si le flot k ne passe pas par l'arc e . Le sous-problème (7.2.8) a autant de contraintes de reroutage que de flot dans \mathcal{K}^e .

Soit A^ϵ la matrice des contraintes associée à (7.2.8), nous avons alors

$$A^\epsilon = \begin{pmatrix} A^{\epsilon,1} & & & \\ & A^{\epsilon,2} & & \\ & & \ddots & \\ & & & A^{\epsilon,K} \end{pmatrix}$$

avec

$$A^{\epsilon,k} = \begin{cases} (-\Pi_\epsilon^k & \omega^{\epsilon,k}) \text{ si l'arc } \epsilon \text{ porte le flot } k, \\ \{ & \} \text{ sinon.} \end{cases} \quad (7.2.17)$$

Pour tout $k \in \mathcal{K}^\epsilon$ les matrices de base possible $B^{\epsilon,k}$ associées à la matrice $A^{\epsilon,k}$ sont définies par :

$$B^{\epsilon,k} = \{1\} \text{ si la variable basique est de reroutage et } \{-1\} \text{ autrement.}$$

Nous désignons par $L(\epsilon)$ et $Q(\epsilon)$ respectivement la partie linéaire et la partie quadratique du critère du sous-problème (7.2.8). Nous avons :

- $L(\epsilon) = \sum_{k=1}^K [D^T \Pi^{\epsilon,k} - \frac{1}{\Lambda}(z(\epsilon) + \Lambda t(\epsilon)) \Pi^{\epsilon,k}] u^{\epsilon,k} + [D^T - \frac{1}{\Lambda}(z(\epsilon) + \Lambda t(\epsilon))] u^{\epsilon,0}$
- $Q(\epsilon) = \frac{1}{2}(u(\epsilon), u^\epsilon)^T H(u(\epsilon), u^\epsilon)$ avec

$$H = \frac{1}{\Lambda} \begin{pmatrix} Id(n_n, n_n) & O(n_n, n^\epsilon) & O(n_n, n) \\ O(n^\epsilon, n_n) & L(n^\epsilon, n^\epsilon) & \delta(n^\epsilon, n) \\ O(n, n_n) & \delta(n^\epsilon, n)^T & Id(n) \end{pmatrix}, \quad (7.2.18)$$

où

- $n^\epsilon = \sum_{k=1}^K n^{\epsilon,k}$,
- $n_n = \sum_{k=1}^K n^k$,
- $O(i, i')$ et $Id(i, i')$ sont respectivement les matrices nulle et identité de taille $i \times i'$,
- L est une matrice symétrique d'ordre n^ϵ définie par :
 $L_i^j =$ nombre d'arcs communs aux $i^{\text{ème}}$ et $j^{\text{ème}}$ chemins de réserve.

– δ est une matrice de taille $n^e \times n$ définie par :

$$\delta^T = (\Pi^{e,1} \quad \dots \quad \Pi^{e,K}).$$

Nous résolvons les problèmes découplés de l'étape proximale (b) (7.2.8) par une méthode de gradient réduit.

Nous allons voir dans la suite que le nombre de ces sous-problèmes peut être inférieur à n et que leur taille peut être réduite.

7.2.4 Réduction du nombre de copies

Pour introduire notre stratégie de copies nous avons annoncé qu'il s'agit de copier les variables couplantes autant de fois qu'elles sont susceptibles de coupler. Pourtant la formulation (7.2.1) peut contenir des copies inutiles. Par exemple, les flots k qui ne sont pas affectés par la panne e , n'ont pas besoin de copies $x^k(e)$ de routage. Et même si un flot k est dans \mathcal{K}^e , tous les chemins nominaux transportant k ne passent pas forcément par e . Le flux qui est porté par ces chemins n'intervient pas dans le couplage de f_e (7.2.4) et sa copie est complètement sans intérêt ($\Pi_e^{kch} = 0$). Une autre situation qui doit être prise en compte est le cas où une panne d'arc e n'est pas active. Autrement dit, l'arc e ne porte aucun trafic nominal. Dans ce cas toutes les copies associées à cette panne sont inutiles.

L'implémentation de cette méthode de décomposition tient compte de ces remarques. En pratique nous ne copions que ce qui doit l'être. Pour avoir une présentation plus simple de l'algorithme, nous avons préféré garder la formulation (7.2.8) et remplacer uniquement \mathcal{K} par \mathcal{K}^e .

L'étape de projection est aussi concernée par ce changement, puisque le nombre de copies n'est pas toujours égal à $(n + 1)$ pour le routage et $(n - 1)$ pour la capacité de réserve. Pour le routage le calcul de la moyenne dépend du nombre de copies du chemin correspondant à chaque composante.

Nous distinguons entre l'ensemble des arcs du graphe \mathcal{E} et l'ensemble des pannes actives que nous notons \mathcal{P} . Nous copions $|\mathcal{P}|$ fois la capacité des arcs de \mathcal{E}/\mathcal{P} et $|\mathcal{P}| - 1$ fois celle des arcs de \mathcal{P} . Le vecteur prix unitaire D est par conséquent défini par :

$$D_i = \frac{d_i}{|\mathcal{P}| - \delta_i^{\mathcal{P}}}. \quad (7.2.19)$$

avec $\delta_i^{\mathcal{P}} = \begin{cases} 1 & \text{si } i \in \mathcal{P} \\ 0 & \text{sinon.} \end{cases}$ L'ensemble des pannes actives \mathcal{P} peut changer au cours des itérations externes suite à une nouvelle génération (ou élimination) de chemins. Par conséquent, d'après (7.2.19), le coût unitaire des copies utiles D_i change en fonction de l'itération majeure. Un chemin dans le réseau de réserve peut avoir alors différents prix au cours de la résolution de PSG. Pour des raisons pratiques, nous préférons supposer D_i toujours égal à $\frac{d_i}{n-1}$, modifier l'étape de projection et gérer les copies inutiles comme suit. Pour tout $e' \notin \mathcal{P}$,

$$z_i^{j+1}(e') = \begin{cases} \frac{\sum_{e \in \mathcal{P}} u_z^j(e)_i}{|\mathcal{P}| - 1} & \text{si } i \in \mathcal{P}, \\ \frac{\sum_{e \in \mathcal{P}} u_z^j(e)_i}{|\mathcal{P}|} & \text{si } i \notin \mathcal{P}, \end{cases} \quad (7.2.20)$$

où les autres copies $u_z^j(e)$ $e \in \mathcal{P}$ sont évidemment obtenues par résolution des sous-problèmes (7.2.8). L'implémentation de cet algorithme et le choix de la base de données tiennent compte de tous ces changements.

7.2.5 Conclusion

Cette stratégie nous a permis de décomposer le problème en $(n + 1)$ sous-problèmes simples à résoudre.

Dans la suite nous considérons une deuxième stratégie obtenue en gardant les contraintes de routage et en partageant le prix nominal entre les copies du routage.

7.3 Deuxième stratégie de décomposition (DP2)

Dans cette deuxième stratégie de décomposition, nous copions uniformément le routage. Toute copie $x(e)$ du routage est maintenant associée à une panne e et satisfait la contrainte de routage (6.2.4.i.k), en plus des contraintes de reroutage (6.2.4.v.e.k). Il n'y aura donc plus de copie principale. Le coût unitaire d'un chemin nominal $i \in I^k$, $c^T \Pi_i^k$, est réparti uniformément entre les copies $x_i^k(e)$ du flux qu'il porte. Les copies de la capacité de réserve $z(e)$ sont gérées comme dans DP1.

Cette section est organisée comme suit. D'abord nous formulons le PSG en suivant cette nouvelle stratégie. Ensuite nous spécialisons l'algorithme de décomposition proximale à cette nouvelle formulation de PSG. Enfin nous expliquons comment nous pouvons réduire le nombre de copies et donc la taille du problème avec copies.

Les figures 7.3.1 et 7.3.2 résument les transformations de la matrice des contraintes par cette deuxième stratégie dans le cas d'un problème de sécurisation à deux pannes.

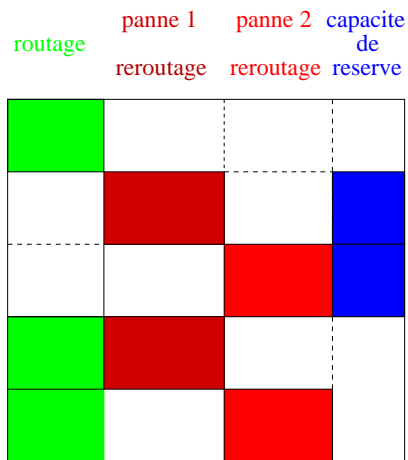


Figure 7.3.1: Matrice des contraintes d'un problème à deux pannes

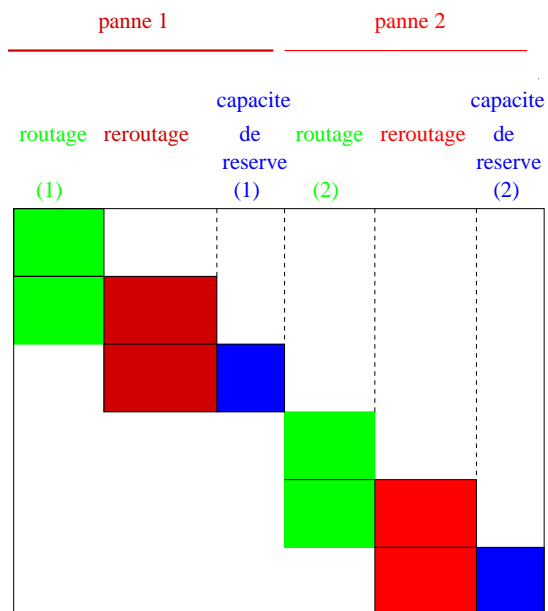


Figure 7.3.2: Matrice des contraintes du problème avec copies

Nous précisons que la matrice de la figure 7.3.2 ne tient pas compte des contraintes d'égalité des copies.

7.3.1 Formulation du problème avec copies

En suivant cette nouvelle stratégie de copies, nous obtenons la formulation suivante:

$$\left\{ \begin{array}{l} \text{Min}_{x^1(1), \dots, x^K(n), x^{1,0}, \dots, x^{n,K}, z(1), \dots, z(n)} \sum_{e=1}^n C^T \sum_{k=1}^K \Pi^k x^k(e) + \sum_{e=1}^n D^T \hat{\Delta}_e z(e); \\ \\ e \in E \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z(e), \quad (\lambda^e) \\ x^{e,0} \geq 0, \quad (s^{e,0}) \\ z(e) \geq 0, \quad (s_z(e)) \\ \\ k \in \mathcal{K} \left\{ \begin{array}{l} \omega^k x^k(e) = r^k, \quad (\alpha_e^k) \\ \omega^{e,k} x^{e,k} = \Pi_e^k x^k(e), \quad (\beta_e^k) \\ x^k(e) \geq 0, \quad (s^k(e)) \\ x^{e,k} \geq 0, \quad (s^{e,k}) \end{array} \right. \\ \\ z(1) = \dots = z(n), \quad (\tilde{t}) \\ x(1) = \dots = x(n), \quad (\tilde{y}) \end{array} \right. \quad (7.3.1)
 \end{array} \right.$$

où

- $D_i = \frac{d_i}{n-1}$.
- $C_i = \frac{c_i}{n}$.
- pour $e = 1, \dots, n$,
 - $t(e)$ est la variable duale à la $e^{\text{ème}}$ copie de la capacité de réserve.
 - $s^k(e)$ est la variable duale à la copie du routage de la demande k , $x^k(e)$ pour $k \in \mathcal{K}$.
 - α_e^k est le multiplicateur de Lagrange associé à la contrainte de routage satisfaite par $x^k(e)$.
- $\tilde{y} = (\tilde{y}(1), \dots, \tilde{y}(n))$ est le multiplicateur associé à la contrainte d'égalité des copies du routage.

- $\tilde{t} = (\tilde{t}(1), \dots, \tilde{t}(n))$ est le multiplicateur associé à la contrainte d'égalité des copies de capacité de réserve.

Le reste des multiplicateurs joue le même rôle que dans la formulation (6.2.4).

Contrairement à la formulation (7.2.1) de PSG, dans (7.3.1) nous avons un seul type de sous-problèmes. Nous pouvons écrire la formulation (7.3.1) comme suit :

$$\left\{ \begin{array}{l} \text{Min}_{x(1), \dots, x(n), x^{1,0}, \dots, x^{n,K}, z(1), \dots, z(n)} \sum_{e=1}^n f_e(x(e), x^e, z(e)); \\ z(1) = \dots = z(n), \quad (\tilde{t}) \\ x(1) = x(1) = \dots = x(n), \quad (\tilde{y}) \end{array} \right. \quad (7.3.2)$$

avec

$$f_e(x(e), x^e, z(e)) = \left\{ \begin{array}{l} C^T \sum_{k=1}^K \Pi^k x^k(e) + D^T \hat{\Delta}_e z(e) \text{ si } \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z(e), \\ x^{e,0} \geq 0, \\ z(e) \geq 0, \\ k \in \mathcal{K} \left\{ \begin{array}{l} \omega^k x^k(e) = r^k, \\ \omega^{e,k} x^{e,k} = \Pi_e^k x^k(e), \\ x^k(e) \geq 0, \\ x^{e,k} \geq 0, \end{array} \right. \end{array} \right. \\ +\infty \text{ sinon.} \end{array} \right. \quad (7.3.3)$$

Comme dans le cas de la première stratégie de décomposition (7.2.2), les fonctions f_e sont indépendantes.

Nous désignons encore par X^0 le vecteur des copies du routage qui est dans ce cas $(x(1), \dots, x(n))$ et appartient à $\mathbb{R}^{n \times n_n}$. Suite à ce changement, l'espace vectoriel \mathcal{H} et le sous-espace réalisable \mathcal{A} sont dans le cas de (7.3.2) définis par :

- $\mathcal{H} = \mathbb{R}^{(n) \times n_n} \times \mathbb{R}^{n_r + n^2} \times \mathbb{R}^{n^2}$,
- $\mathcal{A} = \{(X, Z) \in \mathcal{H}; x(1) = \dots = x(n) \text{ et } z(1) = \dots = z(n)\}$.

Nous précisons que les vecteurs de reroutage $X^e = (x^{e,1}, \dots, x^{e,K}, x^{e,0})$ et de copie de capacité de réserve $Z = (z(1), \dots, z(n))$ ont les mêmes dimensions que dans la

première stratégie (7.2.2). Toutes les remarques faites pour (7.2.2) concernant ces variables restent valables pour la formulation (7.3.2).

Ce qui diffère (7.3.2) de (7.2.2) est le fait que toutes les copies du routage satisfont des contraintes de routage et portent un prix. Pour résoudre le programme maître, l'algorithme que nous proposons dans cette partie applique la méthode de décomposition proximale à (7.3.1). L'algorithme ADP alterne des étapes proximales et des étapes de projection jusqu'à l'obtention d'une solution. Ces deux étapes sont détaillées dans la suite.

7.3.2 Etape proximale

L'étape proximale (étape **2**) de l'algorithme ADP, appliquée au cas du problème (7.3.2) consiste à trouver u^j définie par

$$u^j = \underset{u}{\operatorname{argmin}} \sum_{e=1}^n f_e(u(e), u^e, u_z(e)) + \frac{1}{2\Lambda} \|u - ((X, Z)^j + \Lambda(Y, T)^j)\|^2,$$

où f_e est définie par (7.3.3) et les vecteurs $(X, Z)^j$ et $(Y, T)^j$ sont respectivement les projetés sur \mathcal{A} et \mathcal{A}^- obtenus à l'étape de projection précédente. La partie quadratique rajoutée au critère n'affecte pas la séparabilité du critère.

L'algorithme résout alors pour toute panne e un problème quadratique avec des contraintes de routage et de reroutage :

$$\left\{ \begin{array}{l} \underset{u(e), u^e}{\operatorname{Min}} \sum_{k=1}^K C^T \Pi^k u^k(e) + \frac{1}{2\Lambda} \|u^k(e) - x^{kj}(e) + \Lambda y^{kj}(e)\|^2 + \\ D^T (u^{\epsilon,0} + \sum_{k=1}^K \Pi^{\epsilon,k} u^{\epsilon,k}) + \frac{1}{2\Lambda} \|u^{\epsilon,0} + \sum_{k=1}^K \Pi^{\epsilon,k} u^{\epsilon,k} - (z^j(e) + \Lambda t^j(e))\|^2; \\ k \in \mathcal{K}^e \left\{ \begin{array}{l} \omega^k u^k(e) = u^k, \quad (\alpha_e^k) \\ \omega^{\epsilon,k} u^{\epsilon,k} = \Pi_e^k u^k(e), \quad (\beta_e^k) \\ u^k(e) \geq 0, \\ x^{\epsilon,k} \geq 0, \end{array} \right. \\ k \notin \mathcal{K}^e \left\{ \begin{array}{l} \omega^k u^k(e) = r^k, \quad (\alpha_e^k) \\ u^k(e) \geq 0, \\ u^{\epsilon,k} = 0, \end{array} \right. \end{array} \right. \quad (7.3.4)$$

où \mathcal{K}^e représente toujours l'ensemble des flots affectés par la panne e et pour tout flot $k \in \mathcal{K}$, ω^k et $\omega^{\epsilon,k}$ désignent encore les vecteurs lignes unités de tailles respectives n^k

et $n^{e,k}$.

Le premier bloc de contraintes dans (7.3.4) met en jeu les variables associées aux flots affectés par la panne. Il comporte pour chaque flot affecté une contrainte de routage, une contrainte de reroutage et des contraintes de positivité.

Le deuxième bloc de contraintes correspond aux autres flots (non affectés par la panne). Ces flots n'ont aucune importance par rapport à la panne e , puisqu'ils ne doivent vérifier que des contraintes de routage et de positivité. Le reste des contraintes de ce bloc ($x^{e,k} = 0$) provient du fait qu'aucun chemin nominal associé à ces flots ne contient l'arc e ($\Pi_e^k = 0$). Comme dans l'étape proximale b (7.2.8) nous gardons uniquement les copies des flots qui couplent PSG (6.2.4) et les contraintes qui leurs sont associées. Autrement dit, nous éliminerons le deuxième bloc de contraintes dans (7.3.4) et les variables qui correspondent à tout flot $k \notin \mathcal{K}^e$.

Nous désignons par $A^{e,K}$ la matrice des contraintes associées au flot $k \in \mathcal{K}^e$ en cas de la $e^{\text{ème}}$ panne.

Nous avons:

$$A^{e,k} = \begin{pmatrix} \omega^k & O(1, n^{e,k}) \\ -\Pi_e^k & \omega^{e,k} \end{pmatrix} \quad (7.3.5)$$

Dans le cas où le flot k est affecté par la panne e , les matrices de bases $B^{e,k}$ associées à la matrice $A^{e,k}$ sont définies par :

- si l'une des variables basiques est de reroutage, deux matrices de base sont possibles:

$$- B^{e,k} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \text{ si la variable basique de routage est associée à un chemin (nominal) qui ne passe pas par l'arc en panne } e.$$

$$- B^{e,k} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}, \text{ sinon.}$$

- si les deux variables basiques sont de routage: $B^{e,k} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$.

Nous proposons de résoudre les n problèmes découplés de l'étape proximale (7.3.4) par une méthode de gradient réduit comme pour l'étape proximale b (7.2.8) (voir

Annexe).

Nous désignons par $L(e)$ et $Q(e)$ respectivement la partie linéaire et la partie quadratique du critère de (7.3.4). Nous avons

- $L(e) = \sum_{k=1}^K [C^T \Pi^k - \frac{1}{\Lambda}(x^k(e) + \Lambda y^k(e))]u^k(e) + \sum_{k=1}^K [D^T \Pi^{e,k} - \frac{1}{\Lambda}(z(e) + \Lambda t(e))\Pi^{e,k}]u^{e,k} + [D^T - \frac{1}{\Lambda}(z(e) + \Lambda t(e))]u^{e,0}$.
- $Q(e) = \frac{1}{2}(u(e), u^e)^T H(u(e), u^e)$, avec H définie par (7.2.18).

Contrairement à la partie linéaire, la partie quadratique de (7.3.4) est exactement la même que celle de (7.2.8). Il faut tout de même préciser que si nous ne tenons compte que des copies utiles du routage (section précédente) dans les deux formulations (7.2.8) et (7.3.4), le hessien H de la partie quadratique $Q(e)$ aura une plus petite taille dans le cas de la première stratégie de décomposition (7.2.8).

En effet, cette deuxième stratégie (7.3.2) doit copier plus de variables de routage

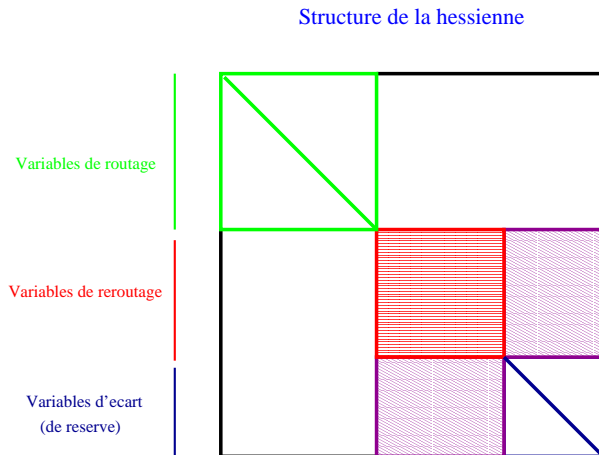


Figure 7.3.3: Structure de la hessienne

que la première (7.2.2) à cause des contraintes de routage. Pour tout chemin nominal associé à un flot $k \in \mathcal{K}^e$, la copie du flux qu'il porte est utile et intervient au moins dans la contrainte de routage. Autrement dit, même si le chemin $i \in I^k$ ne passe pas par l'arc e ($(\Pi_i^k)_e = 0$), si $k \in \mathcal{K}^e$ alors la copie $x_i^k(e)$ (qui était inutile dans (7.3.4)) est utile et peut être même nécessaire pour la contrainte de routage. Le nombre de copies de x_i^k ne dépend plus de la longueur du chemin comme dans (7.2.8) mais du

nombre de pannes qui affectent le routage du flot k .

Mise à part cette différence, toute la partie concernant les copies de la capacité de réserve discutée dans la première stratégie de décomposition reste valable dans le cadre de cette nouvelle stratégie.

7.3.3 Étape de projection

L'étape de projection n'est autre qu'un calcul de moyenne. A la $j^{\text{ème}}$ étape de projection, nous obtenons pour tout $e \in \{1, \dots, n\}$:

$$\begin{cases} x^{j+1}(e) = \frac{\sum_{e=1}^n u^j(e)}{n}, \\ z^{j+1}(e) = \frac{\sum_{e=1}^n u_z^j(e)}{n-1}, \end{cases} \quad (7.3.6)$$

si l'algorithme ne tient pas compte de la réduction du nombre de copies décrite dans le paragraphe précédent.

Dans notre implémentation de cette stratégie, tout comme la première, nous ne copions que les variables susceptibles de coupler. Le vecteur $Z^{j+1} = (z^{j+1}(e'))_{e' \notin \mathcal{P}}$ est définie comme dans le cas de PD1 par (7.2.20). Le vecteur $X^{0j} = (x^j(1), \dots, x^j(n))$ est évidemment défini autrement que dans DP1.

Pour tout flot $k \in \mathcal{K}$, soit \mathcal{E}^k l'ensemble des pannes qui affectent le flot k . Pour tout $e \in \mathcal{P}$ et tout flot $k \in \mathcal{K}$, nous avons alors

$$x^{j+1}(e) = \frac{\sum_{e \in \mathcal{E}^k} u^j(e)}{|\mathcal{E}^k|}. \quad (7.3.7)$$

Les vecteurs Y^{j+1} et T^{j+1} sont déduits grâce à la relation (7.2.11). Toutes les remarques concernant l'optimisation de la place mémoire occupée par les copies et leurs projetés effectuées dans le cas de DP1 restent encore valables pour DP2.

Remarque 7.3.1. La répartition du prix unitaire nominal c sur les copies du routage dépend ici du flot. En effet, à chaque flot k , la formulation (7.3.4) copie $|\mathcal{E}^k|$ fois le vecteur x^k . Puisque la répartition du coût est uniforme, nous attribuons C^k à toute copie de x^k comme prix unitaire :

$$C^k = \frac{c}{|\mathcal{E}^k|}. \quad (7.3.8)$$

Pour les mêmes raisons citées pour DP1, le coût unitaire des copies de capacité de réserve reste égale à $D = \frac{d}{n-1}$ (voir paragraphe 7.2.4). Nous rappelons que nous avons préféré garder ce coût unitaire pour pouvoir retrouver les coûts des chemins de réserve sans devoir les calculer à chaque itération majeure. Cette remarque est importante pour l'implémentation des deux algorithmes présentés dans ce chapitre et la gestion de la base de données.

Pour ne pas encombrer la place mémoire avec les données du problème, nous codons tous les chemins générés de façon qu'un vecteur de \mathbb{R}^{30} soit remplacé par un scalaire. Ce qui facilite le stockage de ces chemins mais les rend inaccessibles sans décodage. Pour ne pas devoir décoder les chemins à chaque fois que l'algorithme s'en sert, nous gardons toutes les informations nécessaires (coût nominal, de réserve, longueur, pannes l'affectant, code) qui les concernent et nous choisissons un modèle qui permet une mise à jour facile de toutes ces informations. D'où le choix du coût D des copies de la capacité de réserve.

Si le coût D était défini par (7.2.19), un arc du réseau de réserve aurait pu ne pas avoir le même prix unitaire au cours de l'algorithme. En effet, en choisissant D comme dans (7.2.19), sa valeur dépendra de \mathcal{P}^j et de son cardinal. Le sous-ensemble \mathcal{P}^j peut changer après la génération des chemins nominaux et par conséquent les coûts des chemins de réserve aussi. Contrairement aux chemins nominaux où les nouveaux coûts des chemins nominaux peuvent se déduire des anciens (multiplication par $\frac{|\mathcal{E}^k|^j}{|\mathcal{E}^{k^{j+1}}|}$), les nouveaux coûts des chemins de réserve ne sont pas évidents à calculer à partir des anciens.

7.3.4 Avantages et inconvénients de la deuxième stratégie de décomposition

Le seul inconvénient de cette stratégie est l'importance du nombre de variables et de contraintes par rapport à la première. Les variables supplémentaires sont dues aux copies $x_i^k(\epsilon)$ pour $k \in \mathcal{K}^\epsilon$ et i tel que $\Pi_i^k = 0$. Le nombre de contraintes dans (7.3.4) est le double de celui dans (7.2.8). L'augmentation du nombre de variables et le dédoublement du nombre de contraintes peuvent présenter des inconvénients. Mais cette stratégie présente aussi des avantages par rapport à la première, comme la réalisabilité des copies du routage. Les problèmes quadratiques (7.3.4) résolus à l'étape proximale sont plus grands que (7.2.8) mais nous espérons que DP2 convergera

plus rapidement que DP1. Intuitivement, elle devrait être plus stable car toutes les copies sont soumises aux contraintes de routage.

7.4 Génération de chemins et garanties d'optimalité

Le problème maître de PSG (6.2.6) peut être résolu par l'algorithme DP1 en suivant la première stratégie de décomposition ou par l'algorithme DP2 en suivant la deuxième. Nous rappelons que le problème maître consiste en un problème de sécurisation globale restreint à des sous-ensembles de chemins nominaux et de réserve. Pour générer de nouveaux chemins, à chaque itération externe l'algorithme appelle des programmes satellites (appelés aussi oracles) qui calculent pour tout flot k un plus court chemin nominal et des plus courts chemins de réserve associés aux pannes qui l'affectent. Les coûts unitaires associés à chaque programme satellite sont donnés par la solution duale du programme maître. Nous utilisons l'algorithme de Dijkstra pour résoudre ces problèmes de plus courts chemins.

Ce schéma algorithmique utilisant la méthode de génération de colonnes et calculant les estimations inférieures et supérieures de la valeur optimale a été largement discuté au chapitre précédent. Nous nous contentons ici de l'appliquer aux deux algorithmes proposés DP1 et DP2. Puisque la résolution du programme maître (6.2.6) peut ne pas être exacte (ne fournit pas forcément une estimation supérieure) nous proposons à la fin de cette section une méthode de calcul de l'estimation supérieure valable même pour les itérations internes de DP1 et DP2.

7.4.1 Calcul des multiplicateurs

Les formulations (7.2.1) et (7.3.2) de PSG sont équivalentes à (6.2.4). Ce n'est pas par hasard que des variables duales de ces formulations portent les mêmes notations mais c'est parce qu'elles désignent les mêmes variables. Nous allons le justifier pour les multiplicateurs α^k , β^k et λ^e . Pour toute panne e le vecteur λ^e est la variable duale au vecteur capacité résiduelle de réserve $x^{e,0}$ et ceci dans les trois formulations (6.2.4), (7.2.1) et (7.3.2). A l'optimalité nous avons alors le même λ^e . En dérivant les trois fonctions Lagrangiennes l , l^1 et l^2 associées respectivement à (6.2.4), (7.2.1) et (7.3.2) nous obtenons (6.2.5.e.k.ii). Pour une même solution optimale nous déduisons

que nous avons les mêmes multiplicateurs β_k^e .

Pour les multiplicateurs α^k nous allons considérer chaque stratégie de décomposition séparément. Nous commençons par (7.2.1).

DP1

Après la dérivation du Lagrangien l^1 associé à (7.2.1) par rapport aux copies du flot k , nous obtenons :

$$\begin{cases} \text{(i)} & \Pi^{kT} c + \omega^{kT} \alpha^k + y^k(0) = s^k(0), \\ e \in \mathcal{E}^k & \left\{ \begin{array}{l} \text{(e.ii)} \\ - \Pi_{e}^k \beta_e^k + y^k(e) \end{array} \right. = s^k(e), \end{cases} \quad (7.4.1)$$

L'addition des équations de (7.4.1) donne :

$$\Pi^{kT} c + \omega^{kT} \alpha^k - \Pi^{kT} \beta^k + \sum_{i=0}^n y^k(i) = \sum_{i=0}^n s^k(i). \quad (7.4.2)$$

Puisque $Y \in \mathcal{A}^-$ à l'optimum, nous avons

$$\sum_{i=0}^n y^k(i) = O(n^k, 1).$$

L'équation (7.4.2) est alors équivalente à (6.2.5.k.i). Par conséquent, α^k désigne la même variable dans (7.2.1) et (6.2.4).

DP2

La deuxième stratégie de décomposition (7.3.1) copie les contraintes de routage et associe à chaque copie un multiplicateur α_e^k . Nous allons montrer le lien entre ces multiplicateurs et ceux de la formulations (6.2.4).

Après dérivation de l par rapport au copies du routage pour tout flot k et toute panne $e \in \mathcal{E}^k$, nous obtenons :

$$\Pi^{kT} C + y^k(e) + \alpha_e^k - \Pi_{e}^k \beta_e^k = s^k(e), \quad (7.4.3)$$

En sommant sur $e \in \mathcal{E}^k$ les équations (7.4.3) pour un flot k donné, nous obtenons

$$\Pi^k{}^T c + \sum_{e=1}^n \alpha_e^k - \Pi^k{}^T \beta^k + \sum_{e=1}^n y^k(e) = \sum_{e=1}^n s^k(e). \quad (7.4.4)$$

En comparant (7.4.4) avec (6.2.5.k.i), grâce à l'unicité des multiplicateurs (associés à une solution), nous avons :

$$\alpha^k = \sum_{e=1}^n \alpha_e^k. \quad (7.4.5)$$

Calcul de λ^e

Les algorithmes DP1 et DP2 ne calculent pas les multiplicateurs $\lambda^e, e \in \{1, \dots, n\}$. A chaque étape proximale l'algorithme DP1 (resp DP2) résout les sous-problèmes (7.2.8) (resp (7.3.4)) par un algorithme de gradient réduit. Les multiplicateurs λ^e ne sont pas explicitement calculés par l'algorithme, puisque nous avons éliminé les contraintes de capacité de réserve. Nous rappelons que λ^e est la variable duale associée à la capacité résiduelle de réserve $x^{e,0}$ à l'optimalité (6.2.5).

Pour tout $e \in \{1, \dots, n\}$ nous introduisons les fonctions Lagrangiennes $l^1(e)$ et $l^2(e)$ respectivement associées aux sous-problèmes (7.2.8) et (7.3.4). En dérivant ces fonctions par rapport à la capacité résiduelle de la copie de la capacité de réserve $u^{e,0}$ nous obtenons :

$$D + (1/\Lambda)(u_z(e) - z(e) - \Lambda t(e)) - \tilde{s}^{e,0}, \quad (7.4.6)$$

où $\tilde{s}^{e,0}$ est la variable duale à $u^{e,0}$.

Grâce à l'unicité des multiplicateurs à l'optimum nous avons :

$$\lambda^e = D + t(e) \quad (7.4.7)$$

7.4.2 Garanties d'optimalité

Au chapitre précédent nous avons exposé une méthode de calcul d'une estimation inférieure de la valeur optimale de PSG (6.2.4). Nous avons également intégré une méthode de perturbation des multiplicateurs pour calculer cette estimation. Toute la partie concernant le calcul de l'estimation inférieure reste valable pour les algorithmes présentés dans ce chapitre. Pour toute panne e , la valeur de λ^e est donnée par (7.4.7).

Cependant, si le programme maître DP1 ou DP2 ne fournit qu'une solution approchée, nous pouvons déduire la valeur de λ^e à partir de (7.4.6) :

$$\lambda^e = D + (1/\Lambda)(u_z(e) - z(e) - \Lambda t(e)).$$

La solution du programme maître (6.2.6) fournit une estimation supérieure à la valeur optimale du PSG (6.2.4). Mais si cette solution est approchée elle peut ne pas être réalisable pour le PSG (6.2.4).

A chaque itération de DP1 et DP2, il est possible de calculer un point réalisable pour le PSG. Il suffit de considérer un routage réalisable $\tilde{x} = (\tilde{x}^1, \dots, \tilde{x}^n)$ [vérifiant les contraintes (6.2.4.i.k) et (6.2.4.ii.k)], et de calculer la capacité de réserve qui assure sa survie en cas de toute panne d'arc :

$$\left\{ \begin{array}{l} \text{Min}_{x^{1,0}, \dots, x^{n,K}, z} d^T z; \\ e \in \mathcal{P} \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = \hat{\Delta}_e z, \\ \omega^{e,k} x^{e,k} = \Pi_e^k \tilde{x}^k, \quad k = 1, \dots, K, \\ x^{e,k} \geq 0, \quad k = 0, \dots, K, \end{array} \right. \\ z \geq 0. \end{array} \right. \quad (7.4.8)$$

Le problème (7.4.8) a été traité dans [86, 39].

La solution de ce problème donne une bonne estimation supérieure mais elle est très coûteuse. Nous devons faire un compromis entre la qualité de l'estimation et la difficulté de son calcul.

Nous avons choisi de calculer à chaque itération interne de DP1 et DP2 un point réalisable au problème (7.4.8). Ce point peut être calculé avec un algorithme inspiré de DP1 et DP2. Il s'agit de résoudre les $|\mathcal{P}|$ problèmes suivants :

$$\left\{ \begin{array}{l} \text{Min}_{x^{e,0}, \dots, x^{e,K}, z(e)} D^T z(e) + (1/2\Lambda) \|z(e) - z^j(e)\|^2; \\ x^{e,0} + \sum_{k=1}^K \Pi^{e,k} x^{e,k} = z(e), \\ \omega^{e,k} x^{e,k} = \Pi_e^k \tilde{x}^k, \quad k = 1, \dots, K, \\ x^{e,k} \geq 0, \quad k = 0, \dots, K, \\ z(e) \geq 0, \end{array} \right. \quad (7.4.9)$$

où

- z^j est la capacité de réserve calculée à la l'itération j ,

- et \tilde{x} dépend de la décomposition choisie (DP1 ou DP2). En effet, nous choisissons \tilde{x}^k comme suit :

$$\tilde{x}^k = \begin{cases} x^{kj}(0), & \text{la copie principale dans le cas de DP1.} \\ x^{kj}(1) = \dots = x^{kj}(n), & \text{le routage moyen dans le cas de DP2.} \end{cases} \quad (7.4.10)$$

Nous pouvons résoudre les problèmes (7.4.9) par la méthode du gradient réduit. Le facteur Λ peut être différent de celui de l'étape proximale de DP1 et de DP2.

Le coût de la capacité z définie par : $z_i = \max_{\epsilon}(z_i(\epsilon))$ pour $i = 1, \dots, n$ est une estimation supérieure de PSG (6.2.4).

7.5 Validation de DP1 et DP2

Nous avons validé les algorithmes DP1 et DP2 sur de petits problèmes générés aléatoirement. L'intérêt de cette section est d'avoir une idée sur le comportement des algorithmes. Nous étudions numériquement l'influence des paramètres des algorithmes DP1 et DP2 sur leurs convergences.

Les paramètres qui nous intéressent sont :

1. le nombre maximal d'itérations internes que nous notons *maxiter*,
2. le paramètre de la décomposition proximale Λ ,
3. le facteur de relaxation ρ .

L'influence de ces paramètres va être étudiée en utilisant l'algorithme DP1. Nous ne reportons que les résultats obtenus avec un réseau complet à 6 nœuds (15 arcs et 15 demandes). Nous avons trouvé ces résultats plus représentatifs.

Nous finirons cette section par une comparaison de DP1 et DP2 sur trois problèmes générés aléatoirement.

7.5.1 Influence du nombre maximal d'itérations mineures

Dans un premier temps nous prenons *maxiter* = 100. Nous obtenons alors la figure (7.5.4) qui représente la variation des estimations supérieure et inférieure. Nous précisons que le facteur de relaxation ρ et le paramètre Λ gardent 1 comme valeur.

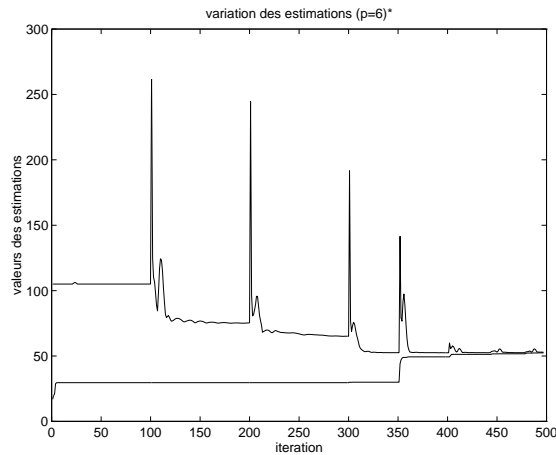


Figure 7.5.4: $p = 6$, $n = 15$ $K = 15$ et $maxiter = 100$

Chaque pic dans la courbe des estimations supérieures correspond à une nouvelle itération majeure, donc à un nouveau programme maître avec plus de chemins. La figure (7.5.5) correspond au cas où $maxiter = 10$.

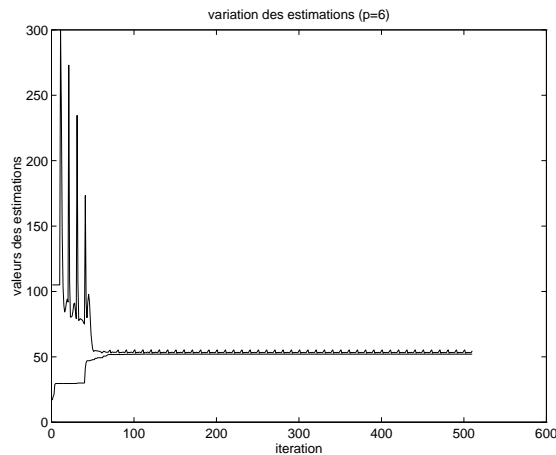


Figure 7.5.5: $p = 6$, $n = 15$ $K = 15$ et $maxiter = 10$

Nous remarquons que la courbe des estimations supérieures est en dents de scie et garde presque la même erreur pendant les 450 dernières itérations. En limitant le nombre d'itérations internes à 10 l'algorithme génère plus de chemins et résout alors de plus grands programmes maîtres.

Pendant les premières itérations internes, le choix qui semble être le plus efficace est $maxiter = 10$ (Figure 7.5.5). Mais la précision atteinte par la valeur $maxiter = 100$ est plus petite que celle obtenue avec $maxiter = 10$ (Figure 7.5.4).

Il serait peut être plus judicieux de limiter le nombre d'itérations internes aux premières itérations externes et d'être ensuite plus exigeant pour la précision vers les dernières itérations. Sur la Figure 7.5.6, l'évolution des écarts relatifs (le rapport de l'écart entre les deux estimations sur l'estimation supérieure) dans les deux cas est plus visible. La courbe qui a la plus grande pente décrit l'écart relatif du cas où $maxiter = 10$.

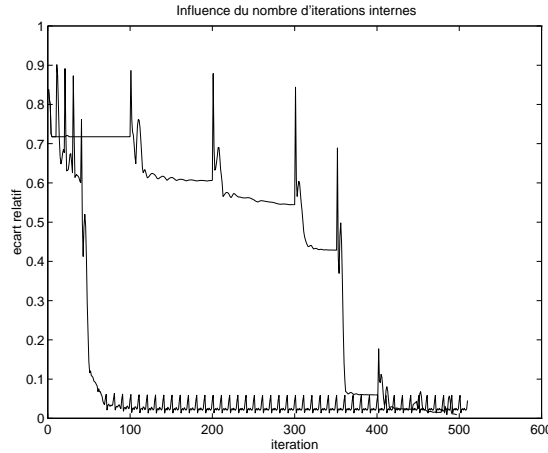


Figure 7.5.6: $p = 6$, $n = 15$, $K = 15$, $maxiter \in \{10, 100\}$

7.5.2 Influence du paramètre de la méthode de décomposition proximale

Le résultat énoncé dans [91] concerne uniquement les opérateurs fortement monotones (La valeur optimale de Λ est l'inverse du coefficient de Lipschitz). En dehors de ce contexte nous devons utiliser des heuristiques pour choisir une valeur de Λ qui accélère la convergence. Eckstein dans [35] s'est intéressé à l'influence du choix du paramètre de la méthode des directions alternées dans le cas de l'optimisation linéaire. D'après ses résultats numériques, l'algorithme est plus efficace pour $\Lambda = \theta c_{max}$ où c_{max} est la composante maximale du coût unitaire et θ est un réel pris entre $1/10$ et $1/3$.

Intuitivement la valeur de Λ doit dépendre du vecteur coût unitaire. Ouorou dans [110] a considéré le même choix que Eckstein pour $\theta \in [0.1, 0.6]$.

Pour l'algorithme DP1 le coût unitaire est le vecteur $D = \frac{d}{n-1}$. Nous avons alors

$$c_{max} = \frac{\max\{d_i, i \in \{1, \dots, n\}\}}{n-1}.$$

Dans le cas de DP2, la valeur de c_{max} dépendra de l'itération majeure. En effet le coût unitaire C associé aux copies de routage (7.3.8) dépend du nombre des ensembles \mathcal{E}^k qui peuvent changer après génération de nouveaux chemins. A chaque itération majeure nous calculons $c_{max} = \max[\max\{D_i, i \in \{1, \dots, n\}, \max\{C_i, i \in \{1, \dots, n_n\}\}]$. Nous considérons un réseau de 30 nœuds et 30 arcs et traversé par 30 flots. Les deux courbes de la figure (7.5.7) représentent la variation de l'écart relatif au cours des itérations internes.

L'algorithme est beaucoup plus rapide pour $\Lambda = 20$. Il converge après 15 itérations mineures contre 85 itérations pour $\Lambda = 1$.

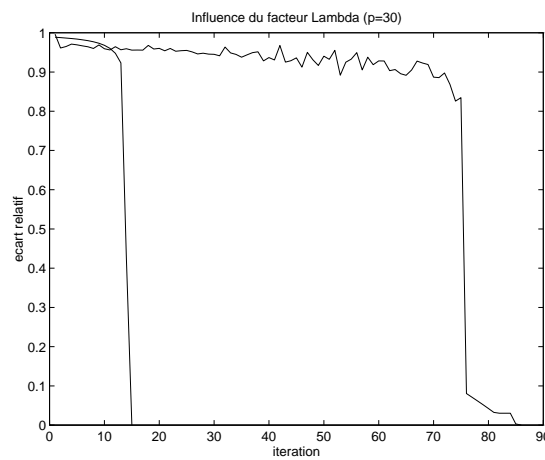


Figure 7.5.7: $p = 30$, $n = 30$ $K = 30$ et $\Lambda \in \{1, 20\}$

La courbe de la Figure 7.5.8 montre l'influence de la valeur de Λ sur le nombre d'itérations que nécessite l'algorithme PD1 pour résoudre un problème de graphe complet à 6 nœuds (15 flots et 15 arcs).

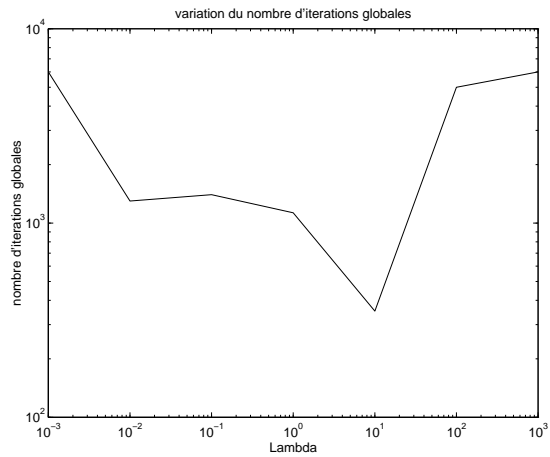


Figure 7.5.8: $p = 6$, $n = 15$ $K = 15$ et $\Lambda \in \{10^{-3}, \dots, 10^3\}$

Pour $\Lambda = 10^{-3}$ et 10^3 nous avons arrêté le programme à cause de sa lenteur.

7.5.3 Influence du facteur de relaxation

Nous rappelons que le facteur de relaxation ρ appartient à $]0, 2[$. Considérons le même réseau complet à 6 nœuds (15 arcs et 15 flots). La figure (7.5.9) représente le nombre d'itérations internes en fonction de ρ .

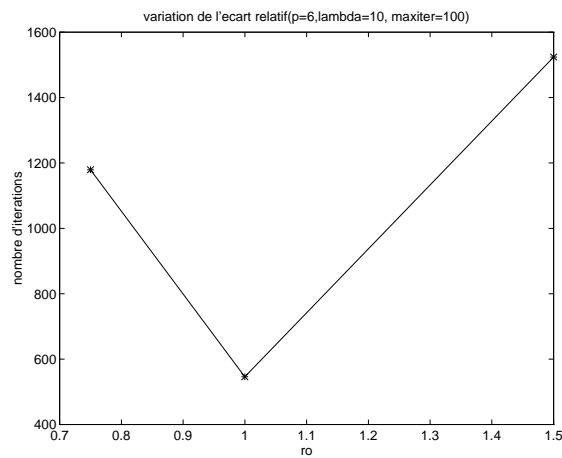


Figure 7.5.9: $p = 6$, $n = 15$ $K = 15$, $maxiter = 100$, $fac = 10$ et $\rho \in \{0.75, 1, 1.5\}$

D'après la courbe de la figure (7.5.9), il semblerait que la valeur la plus judicieuse de ρ est 1. Il faudrait plusieurs tests pour confirmer que ce choix est le meilleur.

7.5.4 Comparaison entre DP1 et DP2

Le choix du sous espace \mathcal{A} est très important. La comparaison des deux algorithmes DP1 et DP2 montre l'influence de ce choix sur la vitesse de convergence de la méthode de décomposition proximale.

L'algorithme DP2 met en jeu plus de variables, soumises au double des contraintes que dans DP1. Mais les essais numériques montrent que l'algorithme DP2 est extrêmement plus rapide que DP1. Ce résultat n'est pas surprenant et il est même prévisible. En effet dans DP1 la seule copie de routage qui satisfait les contraintes de routage est la copie principale. Par conséquent, à l'étape proximale (7.2.8), l'algorithme DP1 reroute des routages non réalisables (pour la contrainte de routage).

Les résultats numériques illustrés dans les figures (7.5.10), (7.5.11) et (7.5.12) confirment la supériorité numérique de la deuxième stratégie de décomposition (DP2).

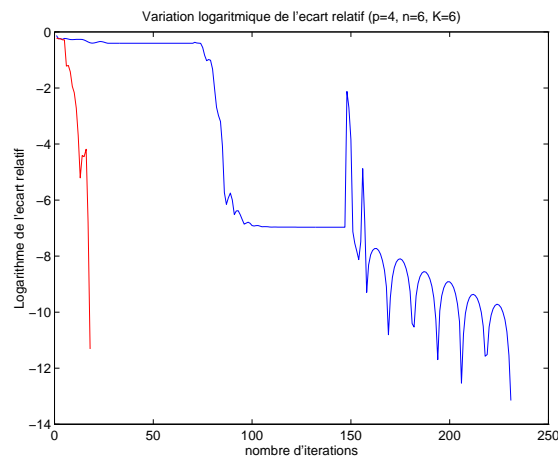
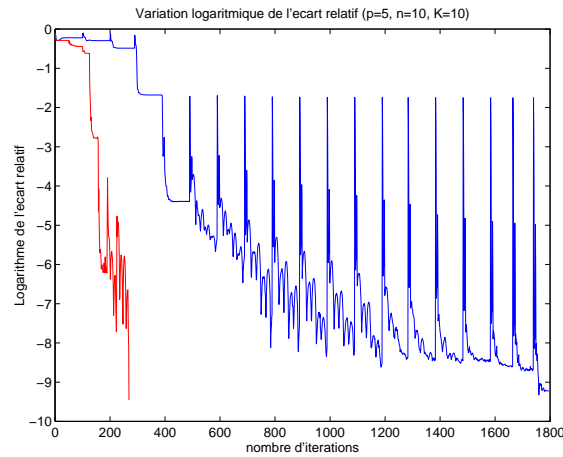
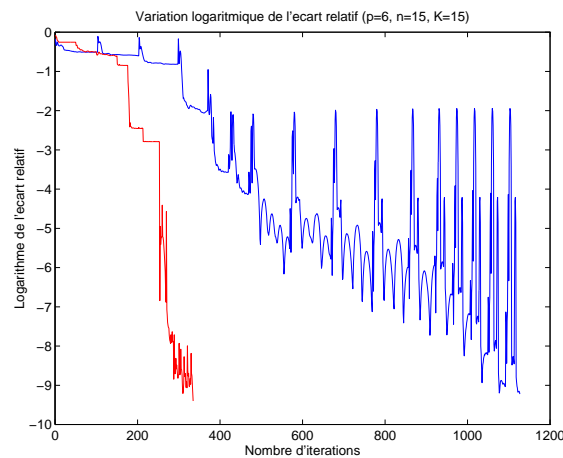


Figure 7.5.10: $p = 4$, $n = 6$, $K = 6$, $\lambda = 1$ et $\rho = 1$

Figure 7.5.11: $p = 5$, $n = 10$, $K = 10$, $\lambda = 1$ et $\rho = 1$ Figure 7.5.12: $p = 6$, $n = 15$, $K = 15$, $\lambda = 1$ et $\rho = 1$

7.5.5 Conclusion

Dans le but d'améliorer l'algorithme de résolution de problème de sécurisation globale nous avons étudié numériquement l'influence de la variation des paramètres de l'algorithme de décomposition proximale (Λ, ρ), ainsi que le nombre maximal d'itérations internes (*maxiter*). Nous avons observé l'importance du paramètre de l'algorithme de décomposition proximale Λ ainsi que le nombre *maxiter*. L'algorithme DP2 a donné de meilleurs résultats que l'algorithme DP1.

8. Sécurisation par une méthode de points intérieurs

8.1 Introduction

L'usage de méthodes de points intérieurs de type prédicteur-correcteur permet la résolution de problèmes de multiflots en exploitant la structure de décomposition du problème, et se montre numériquement robuste.

L'étude exposée dans ce chapitre est inspirée de l'algorithme (APF) proposé pour la résolution du problème de routage (chapitre 4).

Suivant le même principe de décomposition nous adaptons l'algorithme pour résoudre le problème de sécurisation globale. Comme dans le cas du problème de routage nous optons pour une spécialisation de la méthode prédicteur-correcteur à PSG.

Nous commençons par observer la structure du système de déplacement de Newton que l'algorithme prédicteur-correcteur doit résoudre aux étapes affine et de centralisation et ceci à chaque itération.

La résolution de ces systèmes constitue la partie la plus coûteuse de l'algorithme. Un algorithme efficace doit donc exploiter la structure de ces systèmes. En utilisant une technique de décomposition-coordination au niveau du calcul du système de déplacement de Newton, nous obtenons un système réduit dont la résolution découple le système en plusieurs sous-systèmes indépendants de plus petite taille.

L'étude de complexité de cette méthode de décomposition nous amène à distinguer deux types de problèmes de sécurisation. Le premier type est caractérisé par un graphe complet. Dans ce cas, la décomposition est inutile puisque nous obtenons le même ordre de complexité ($O(n^6)$) avec la méthode directe, où n est le nombre d'arcs du graphe. En plus la méthode directe a une matrice d'incidence creuse.

Le deuxième type de problème est caractérisé par un graphe très peu maillé. La complexité de calcul du système de Newton dans la méthode directe peut, dans ce cas, être de l'ordre de n^9 . Par contre la méthode avec décomposition garde le même ordre de complexité (n^6).

Pour trancher entre ces deux méthodes, il suffit donc de connaître la densité du graphe.

Rappelons la formulation de PSG :

$$\left\{ \begin{array}{l} \text{Min}_{x^1, \dots, x^K, x^{1,1}, \dots, x^{e,K}, z} c^T \sum_{k=1}^K \pi^k x^k + d^T z; \\ k \in \{1, \dots, K\} \left\{ \begin{array}{l} \omega^k x^k = r^k, \\ x^k \geq 0, \end{array} \right. \begin{array}{l} (\alpha^k) \\ (s^k) \end{array} \\ e \in E \left\{ \begin{array}{l} x^{e,0} + \sum_{k=1}^K \pi^{e,k} x^{e,k} = \hat{\Delta}_e z, \\ x^{e,0} \geq 0, \end{array} \right. \begin{array}{l} (\lambda^e) \\ (s^{e,0}) \end{array} \\ k \in \{1, \dots, K\} \left\{ \begin{array}{l} \omega^{e,k} x^{e,k} = \pi_e^k x^k, \\ x^{e,k} \geq 0, \end{array} \right. \begin{array}{l} (\beta_e^k) \\ (s^{e,k}) \end{array} \\ z \geq 0, \end{array} \right. \quad (8.1.1) \quad (t)$$

où

- $\hat{\Delta}_e$ est la matrice diagonale d'ordre n définie par :
 $\hat{\Delta}_e = \text{diag}(\mathbf{1} - \delta_e)$, avec δ_e est un vecteur de \mathbb{R}^n qui a toutes ses composantes nulles en excluant la $e^{\text{ème}}$ qui vaut 1.
- ω^k : vecteur ligne unité de taille n^k ,
- $\omega^{e,k}$: vecteur ligne unité de taille $n^{e,k}$.

8.2 Une approche par décomposition d'une méthode de points intérieurs

Comme dans [15] nous utilisons la méthode prédicteur-correcteur introduite par Mizuno, Todd et Ye dans [102]. Nous nous intéressons à la version de grands voisinages non réalisable [15]. La méthode prédicteur-correcteur a l'avantage de posséder à la fois une complexité en $O(\sqrt{n}L)$ (en nombre d'itérations dans le cas réalisable) et une convergence quadratique [16]. Elle consiste à alterner des pas de centralisation, de façons à se rapprocher de la trajectoire centrale et les pas affines (les plus grands pas permettant de rester dans le grand voisinage). La méthode prédicteur-correcteur suit la trajectoire centrale associée à (8.1.1) :

$$\left\{ \begin{array}{l}
 k \in \{1, \dots, K\} \left\{ \begin{array}{l}
 x^k s^k = \mu \mathbf{1}, \\
 \omega^k x^k = r^k, \\
 \pi^{kT} c + \omega^{kT} \alpha^k - \pi^{kT} \beta^k = s^k, \\
 x^k, s^k \geq 0,
 \end{array} \right. \\
 \\
 e \in E \left\{ \begin{array}{l}
 x^{e,0} s^{e,0} = \mu \mathbf{1}, \\
 x^{e,0} + \sum_{k=1}^K \pi^{e,k} x^{e,k} = \hat{\Delta}_e z, \\
 \lambda^e = s^{e,0}, \\
 x^{e,0}, s^{e,0} \geq 0, \\
 \\
 k \in \mathcal{K} \left\{ \begin{array}{l}
 x^{e,k} s^{e,k} = \mu \mathbf{1}, \\
 \omega^{e,k} x^{e,k} = \pi_e^k x^k, \\
 \omega^{e,kT} \beta_e^k + \pi^{e,kT} \lambda^e = s^{e,k} \\
 x^{e,k}, s^{e,k} \geq 0,
 \end{array} \right.
 \end{array} \right. \\
 \\
 tz = \mu \mathbf{1}, \\
 d - \sum_{\epsilon=1}^n \hat{\Delta}_\epsilon^T \lambda^\epsilon = t, \\
 z, t \geq 0,
 \end{array} \right. \quad (8.2.1)$$

A chaque itération, l'algorithme prédicteur-correcteur résout deux systèmes de déplacement de Newton :

$$\left\{ \begin{array}{l}
 \left. \begin{array}{l}
 x^k v^k + s^k u^k = f^k, \\
 \omega^k u^k = 0, \\
 \omega^{kT} \delta_{\alpha^k} - \pi^{kT} \delta_{\beta^k} = v^k,
 \end{array} \right\} k \in \mathcal{K} \text{ (I.k)} \\
 \\
 e \in E \left\{ \begin{array}{l}
 \left. \begin{array}{l}
 x^{e,k} v^{e,k} + s^{e,k} u^{e,k} = f^{e,k}, \\
 \omega^{e,k} u^{e,k} = \pi_e^k u^k, \\
 \omega^{e,kT} \delta_{\beta_e^k} + \pi^{e,kT} \delta_{\lambda^e} = v^{e,k}
 \end{array} \right\} k \in \mathcal{K}^e \text{ (II.e.k)} \\
 \\
 \left. \begin{array}{l}
 x^{e,0} v^{e,0} + s^{e,0} u^{e,0} = f^{e,0}, \\
 \delta_{\lambda^e} = v^{e,0},
 \end{array} \right\} \text{(III.e)} \\
 \\
 \left. \begin{array}{l}
 z \delta_t + t \delta_z = g, \\
 u^{e,0} + \sum_{k=1}^K \pi^{e,k} u^{e,k} = \hat{\Delta}_e \delta_z, \\
 - \sum_{\epsilon=1}^n \hat{\Delta}_\epsilon^T \delta_{\lambda^\epsilon} = t,
 \end{array} \right\} \text{(IV)}
 \end{array} \right. \quad (8.2.2)$$

où

- $(u^k, v^k, \delta_{\alpha^k}, \delta_{\beta^k})$ est le vecteur déplacement de $(x^k, s^k, \alpha^k, \beta^k)$, pour tout $k \in \mathcal{K}$,

- $(u^{e,k}, v^{e,k})$ est le vecteur de déplacement associé à $(x^{e,k}, s^{e,k})$, pour tout $e \in E$ et $k \in \mathcal{K}^e$,
- δ_{λ^e} est le déplacement associé au multiplicateur λ^e , pour tout $e \in E$,
- (δ_z, δ_t) est le vecteur de déplacement associé à (z, t) ,
- $f^k, f^{e,k}, f^{e,0}$ et g dépendent de la nature des pas calculés.

Nous avons

$$f^{i,l} = -x^{i,l} s^{i,l} + (\gamma - 1)\mu \mathbf{1},$$

pour tout $i \in \{0, \dots, n\}$ et $l \in \{0, \dots, K\}$ et de même

$$g = -zt + (1 - \gamma)\mu \mathbf{1}.$$

Le paramètre γ prend les valeurs 0 et 1 pour calculer respectivement le pas affine et le pas de centralisation.

Le système (8.2.2) peut atteindre une très grande taille même pour de petits réseaux. Nous nous inspirons de [15] et proposons un schéma de décomposition analogue.

8.2.1 Schéma de décomposition du système de déplacement de Newton

Il s'agit alors de sélectionner un sous-système de (8.2.2), que nous appellerons aussi le *système réduit*, dont la résolution découple les autres sous-systèmes de (8.2.2). Nous rappelons que pour le problème de routage le système réduit correspond aux contraintes de capacité et les variables couplantes ne sont autres que les déplacements des multiplicateurs associés à ces contraintes. Par analogie, nous pouvons déduire que le système réduit est, dans ce cas, formé par toutes les contraintes de capacité de réserve. Les variables couplantes sont alors les déplacements des multiplicateurs δ_{λ^e} pour $e \in \mathcal{E}_a$.

Les K sous-systèmes (8.2.2.I) correspondent au routage dans le réseau nominal. Pour chaque flot k , si la valeur δ_{β^k} était donnée, le sous-système (8.2.2.I) ne serait autre qu'un système de déplacement de Newton correspondant à un problème de flot simple avec un vecteur coût unitaire égal à $-\delta_{\beta^k}$. Le nombre de sous-systèmes de type (8.2.2.II.e.k) dépend du nombre de couples panne-flot affecté. Au pire des cas ce

nombre pourrait être égal à $n \times K$, si tous les flots sont affectés par toutes les pannes. Un sous-système (8.2.2.II.e.k) peut être considéré comme un système de déplacement de Newton associé à un problème de flot simple avec un coût unitaire égal à δ_{λ^e} , si le déplacement du routage u^k et le multiplicateur δ_{β^k} sont connus.

Les sous-systèmes (8.2.2.III.e) concernent les capacités de réserve résiduelles en cas de chaque panne. La deuxième équation d'un sous-système du type (8.2.2.III.e) correspond aux contraintes de capacité associées à la panne de l'arc e . Cette partie de (8.2.2) est à l'origine de son importante taille. En effet, si toutes les pannes d'arcs sont actives (ce qui est le cas en général), nous avons n^2 contraintes de capacités dans (8.2.2.III).

Pour expliciter le système réduit, nous avons besoin d'introduire quelques notations :

Pour tout flot k dans $\{1, \dots, K\}$

- $d^k = \sqrt{x^k / s^k}$,
- $D^k = \text{diag}(d^k)$,
- $\bar{\omega}^k = \omega^k D^k$,
- $\bar{\pi}^k = \pi^k D^k$,
- $\bar{u}^k = u^k / d^k$,
- $\bar{v}^k = d^k v^k$,
- $\bar{f}^k = f^k / \sqrt{x^k s^k}$,
- pour toute panne $e \in E$:
 - $d^{e,k} = \sqrt{(x^{e,k} / s^{e,k})}$,
 - $D^{e,k} = \text{diag}(d^{e,k})$,
 - $\bar{\omega}^{e,k} = \omega^{e,k} D^{e,k}$,
 - $\bar{\pi}^{e,k} = \pi^{e,k} D^{e,k}$,
 - $\bar{u}^{e,k} = u^{e,k} / d^{e,k}$,
 - $\bar{v}^{e,k} = d^{e,k} v^{e,k}$,
 - $\bar{f}^{e,k} = f^{e,k} / \sqrt{x^{e,k} s^{e,k}}$.

Le système (8.2.3) est obtenu après une mise à l'échelle du système (8.2.2.II.e.k):

$$\begin{cases} \overline{v}^{e,k} + \overline{u}^{e,k} = \overline{f}^{e,k}, \\ \overline{\omega}^{e,k} \overline{u}^{e,k} = \pi_e^k u^k, \\ \overline{\omega}^{e,k T} \delta_{\beta_e^k} + \overline{\pi}^{e,k T} \delta_{\lambda^e} = \overline{v}^{e,k}. \end{cases} \quad (8.2.3)$$

Partant du système (8.2.3), nous exprimons $\delta_{\beta_e^k}$ en fonction de δ_{λ^e} et de u^k . En effet, en multipliant la troisième équation du système (8.2.3) par $\overline{\omega}^{e,k}$ nous obtenons :

$$\delta_{\beta_e^k} = M_{\beta_e^k, \lambda^e} \delta_{\lambda^e} + M_{\beta_e^k, u^k} u^k + C_{\beta_e^k}, \quad (8.2.4)$$

où

- $M_{\beta_e^k, \lambda^e} = \frac{-\overline{\omega}^{e,k} \pi_e^k T}{\|d^{e,k}\|^2},$
- $M_{\beta_e^k, u^k} = \frac{-\pi_e^k}{\|d^{e,k}\|^2},$
- $C_{\beta_e^k} = \frac{\overline{\omega}^{e,k} \overline{f}^{e,k}}{\|d^{e,k}\|^2}.$

Nous introduisons les matrices $M_{\beta^k, \lambda}$ et M_{β^k, u^k} et le vecteur C_{β^k} définis comme suit :

$$\bullet M_{\beta^k, \lambda} = \begin{pmatrix} M_{\beta_1^k, \lambda^1} & 0 & \cdots & 0 \\ 0 & M_{\beta_2^k, \lambda^2} & & \vdots \\ & & \ddots & \\ & & & M_{\beta_n^k, \lambda^n} \end{pmatrix},$$

$$\bullet M_{\beta^k, u^k} = \begin{pmatrix} M_{\beta_1^k, u^k} \\ M_{\beta_2^k, u^k} \\ \vdots \\ M_{\beta_n^k, u^k} \end{pmatrix},$$

$$\bullet C_{\beta^k} = \begin{pmatrix} C_{\beta_1^k} \\ C_{\beta_2^k} \\ \vdots \\ C_{\beta_n^k} \end{pmatrix}.$$

De (8.2.4), nous pouvons déduire que les matrices $M_{\beta^k, \lambda}$, et M_{β^k, u^k} et le vecteur C_{β^k} vérifient :

$$\delta_{\beta^k} = M_{\beta^k, \lambda} \delta_{\lambda} + M_{\beta^k, u^k} u^k + C_{\beta^k}, \quad (8.2.5)$$

Après une mise à l'échelle du système (8.2.2.I.k) nous obtenons le système équivalent suivant :

$$\begin{cases} \bar{v}^k + \bar{u}^k = \bar{f}^k, \\ \bar{\omega}^k \bar{u}^k = 0, \\ \bar{\omega}^{kT} \delta_{\alpha^k} - \bar{\pi}^{kT} \delta_{\beta^k} = \bar{v}^k, \end{cases} \quad (8.2.6)$$

Du système (8.2.6) nous pouvons déduire la relation suivante

$$\delta_{\alpha^k} = \frac{\bar{\omega}^k \bar{\pi}^{kT}}{\|d^k\|^2} \delta_{\beta^k} + \frac{\bar{\omega}^k \bar{f}^k}{\|d^k\|^2}.$$

Ce qui implique que

$$\bar{v}^k = \left[\frac{\bar{\omega}^{kT} \bar{\omega}^k \bar{\pi}^{kT}}{\|d^k\|^2} - \bar{\pi}^{kT} \right] \delta_{\beta^k} + \frac{\bar{\omega}^{kT} \bar{\omega}^k \bar{f}^k}{\|d^k\|^2}$$

Posons $R^k = Id - \frac{\bar{\omega}^{kT} \bar{\omega}^k}{\|d^k\|^2}$.

Nous exprimons u^k en fonction de δ_{β^k} :

$$u^k = R^k \bar{\pi}^{kT} \delta_{\beta^k} + R^k \bar{f}^k$$

En utilisant l'équation (8.2.5), nous pouvons exprimer u^k en fonction de δ_{λ} :

$$u^k = M_{u^k, \lambda} \delta_{\lambda} + C_{u^k}, \quad (8.2.7)$$

avec

- $M_{u^k, \lambda} = [Id - R^k \bar{\pi}^{kT} M_{\beta^k, u^k}]^{-1} R^k \bar{\pi}^{kT} M_{\beta^k, \lambda}$
- $C_{u^k} = [Id - R^k \bar{\pi}^{kT} M_{\beta^k, u^k}]^{-1} [R^k \bar{\pi}^{kT} C_{\beta^k} + R^k \bar{f}^k]$

De la relation (8.2.4) et du système (8.2.3), nous pouvons déduire que

$$\bar{v}^{e,k} = M_{v^{e,k},\lambda^e} \delta_{\lambda^e} + M_{v^{e,k},u^k} u^k + C_{v^{e,k}}$$

avec

- $M_{v^{e,k},\lambda^e} = \bar{\pi}^{e,kT} + \bar{\omega}^{e,kT} M_{\beta_{\xi}^k, \lambda^e},$
- $M_{v^{e,k},u^k} = \bar{\omega}^{e,kT} M_{\beta_{\xi}^k, u^k},$
- $C_{v^{e,k}} = \bar{\omega}^{e,kT} C_{\beta_{\xi}^k}.$

En tenant compte de la relation précédente et du système (8.2.3), nous avons :

$$u^{e,k} = M_{u^{e,k},\lambda^e} \delta_{\lambda^e} + M_{u^{e,k},u^k} u^k + C_{u^{e,k}}, \quad (8.2.8)$$

où

- $M_{u^{e,k},\lambda^e} = -D^{e,k} M_{v^{e,k},\lambda^e},$
- $M_{u^{e,k},u^k} = -D^{e,k} M_{v^{e,k},u^k},$
- $C_{u^{e,k}} = D^{e,k} \bar{f}^{e,k} - D^{e,k} C_{v^{e,k}}.$

Grâce à (8.2.7), nous exprimons u^k en fonction de δ_{λ} . Les vecteurs $u^{e,k}$ peuvent alors s'exprimer uniquement en fonction de δ_{λ} et (8.2.8) devient :

$$u^{e,k} = M_{u^{e,k},\lambda} \delta_{\lambda} + C^{e,k}, \quad (8.2.9)$$

où

- $M_{u^{e,k},\lambda} = M_{u^{e,k},u^k} M_{u^k,\lambda} + [0, \dots, 0, M_{u^{e,k},\lambda^e}, \dots, 0],$
- $C^{e,k} = C_{u^{e,k}} + M_{u^{e,k},u^k} C_{u^k}.$

D'après le système (8.2.2.III.e), nous avons la relation suivante :

$$u^{e,0} = M_{u^{e,0},\lambda^e} \delta_{\lambda^e} + C^{e,0}, \quad (8.2.10)$$

où

- $M_{u^{e,0},\lambda^e} = -D_2^{e,0}$,
- $D_2^{e,0} = \text{diag}(x^{e,0}/s^{e,0})$,
- $C^{e,k} = f^{e,0}/s^{e,0}$.

Il ne nous reste plus qu'à exprimer le déplacement de la capacité de réserve δ_z en fonction de δ_λ .

Pour ce faire nous utilisons les première et troisième équations du système (8.2.2.IV).

Nous obtenons :

$$\delta_z = M_{z,\lambda} \delta_\lambda + C^z, \quad (8.2.11)$$

où

- $M_{z,\lambda} = (D^z \hat{\Delta}_1^T \quad D^z \hat{\Delta}_2^T \quad \cdots \quad D^z \hat{\Delta}_n^T)$,
- $D^z = \text{diag}(z/t)$,
- $C^{e,k} = g/t$.

Nous avons réussi à exprimer les vecteurs $u^{e,i}$ (pour $i \in \{0, \dots, K$ et $e \in \mathcal{E}$) en fonction du vecteur $\delta_\lambda \in \mathbb{R}^{n^2}$ (8.2.10), (8.2.8). La complexité de ce calcul est présentée dans la suite de ce chapitre.

8.2.2 Système réduit

Le système réduit associé au système de déplacement de Newton (8.2.2) d'un problème de sécurisation globale (8.1.1) s'écrit en fonction du vecteur des variables couplantes

$$\delta_\lambda = (\delta_{\lambda^1}, \dots, \delta_{\lambda^n}).$$

$$\begin{cases} u^{1,0}(\delta_\lambda) + \sum_{k=1}^K \pi^{1,k} u^{1,k}(\delta_\lambda) = \hat{\Delta}_1 \delta_z, \\ u^{2,0}(\delta_\lambda) + \sum_{k=1}^K \pi^{2,k} u^{2,k}(\delta_\lambda) = \hat{\Delta}_2 \delta_z, \\ \vdots \\ u^{n,0}(\delta_\lambda) + \sum_{k=1}^K \pi^{n,k} u^{n,k}(\delta_\lambda) = \hat{\Delta}_n \delta_z, \end{cases} \quad (8.2.12)$$

Si toutes les pannes sont actives le système réduit est de taille n^2 (n =nombre d'arcs). En réalité, le système (8.2.12) ne comporte que $n^2 - n$ équations lorsque toutes les pannes sont actives.

Le système réduit (8.2.12) peut être écrit comme suit:

$$(M^0 + \sum_{k=1}^K \Pi^k M^k - M_r) \delta_\lambda = Cst \quad (8.2.13)$$

avec

$$\begin{aligned} \bullet M^0 &= \begin{pmatrix} M_{u^{1,0},\lambda^1} & 0 & \cdots & 0 \\ 0 & M_{u^{2,0},\lambda^2} & & \vdots \\ & & \ddots & \\ & & & M_{u^{n,0},\lambda^n} \end{pmatrix}, \\ \bullet \Pi^k &= \begin{pmatrix} \pi^{1,k} & 0 & \cdots & 0 \\ 0 & \pi^{2,k} & & \vdots \\ & & \ddots & \\ & & & \pi^{n,k} \end{pmatrix}, \\ \bullet M^k &= \begin{pmatrix} M_{u^{1,k},\lambda} \\ M_{u^{2,k},\lambda} \\ \vdots \\ M_{u^{n,k},\lambda} \end{pmatrix}, \\ \bullet M_r &= \begin{pmatrix} \hat{\Delta}_1 D^z \hat{\Delta}_1 & \hat{\Delta}_1 D^z \hat{\Delta}_2 & \cdots & \hat{\Delta}_1 D^z \hat{\Delta}_n \\ \hat{\Delta}_2 D^z \hat{\Delta}_1 & \hat{\Delta}_2 D^z \hat{\Delta}_2 & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\Delta}_n D^z \hat{\Delta}_1 & \hat{\Delta}_n D^z \hat{\Delta}_2 & \cdots & \hat{\Delta}_n D^z \hat{\Delta}_n \end{pmatrix}, \\ \bullet Cst &= \begin{pmatrix} -C^{1,0} - \sum_{k=1}^K \pi^{1,k} C^{e,k} + \hat{\Delta}_1 C^z \\ -C^{2,0} - \sum_{k=1}^K \pi^{2,k} C^{2,k} + \hat{\Delta}_2 C^z \\ \vdots \\ -C^{n,0} - \sum_{k=1}^K \pi^{n,k} C^{n,k} + \hat{\Delta}_n C^z \end{pmatrix}. \end{aligned}$$

8.3 Étude de complexité

Nous rappelons d'abord les notations que nous utiliserons dans la suite :

- n^k et $n^{e,k}$ sont respectivement le nombre de chemins nominaux associés au flot k et celui des chemins de réserve associés à la panne e et au flot k .

- l'ensemble \mathcal{K}^e est formé par les flots affectés par la panne de l'arc e .
- $E^{\epsilon,k}$ est l'ensemble des arcs qui portent les chemins de réserve associés au flot k en cas de la panne e et $n(e,k)$ est son cardinal.
- E^k est l'ensemble des pannes qui affectent le flot k et $n(k)$ représente son cardinal.
- E_a est l'ensemble de pannes actives et n_a est le nombre de ces pannes.

Nous supposons que pour tout flot k :

- le nombre de chemins nominaux n^k est de l'ordre de 1.
- Le nombre $n^{\epsilon,k}$ de chemins de réserve pour la panne e est de l'ordre de 1 pour tout $e \in E$.

8.3.1 Complexité de la construction de la matrice M

En suivant la méthode de décomposition, l'algorithme aurait à construire une matrice pleine (M) d'ordre n^2 et à résoudre le système linéaire correspondant.

$$M = \begin{pmatrix} M_{u^1,0,\lambda^1} & \cdots & 0 \\ \vdots & \ddots & \\ 0 & & M_{u^n,0,\lambda^n} \end{pmatrix} + \sum_{k=1}^K \begin{pmatrix} \pi^{1,k} M_{u^1,k,\lambda} \\ \vdots \\ \pi^{n,k} M_{u^n,k,\lambda} \end{pmatrix} - \begin{pmatrix} \hat{\Delta}_1 D^z \hat{\Delta}_1 & \cdots & \hat{\Delta}_1 D^z \hat{\Delta}_n \\ \vdots & \ddots & \vdots \\ \hat{\Delta}_n D^z \hat{\Delta}_1 & \cdots & \hat{\Delta}_n D^z \hat{\Delta}_n \end{pmatrix}.$$

où

$$\begin{aligned} \pi^{\epsilon,k} M_{u^{\epsilon,k},\lambda} &= \pi^{\epsilon,k} \frac{(d^{\epsilon,k})^2}{\|d^{\epsilon,k}\|^2} \pi^k [I(n^k) - R^k (\bar{\pi}^k)^T M_{\beta^k,u^k}]^{-1} R^k (\pi^k)^T M_{\beta^k,\lambda} \\ &+ [0(n^{\epsilon,k}, n), \dots, -\bar{\pi}^{\epsilon,k} (\bar{\pi}^{\epsilon,k})^T + \pi^{\epsilon,k} (d^{\epsilon,k})^2 [(d^{\epsilon,k})^2]^T (\pi^{\epsilon,k})^T, \dots, 0(n^{\epsilon,k}, n)]. \end{aligned}$$

Le calcul des matrices $\pi^{\epsilon,k} M_{u^{\epsilon,k},\lambda}$ est la partie la plus coûteuse de la construction de la matrice M . En effet, le nombre d'opérations élémentaires nécessaire pour le calcul d'une matrice $\pi^{\epsilon,k} M_{u^{\epsilon,k},\lambda}$ est de l'ordre de n^3 (exactement $n_a^2 n(e,k)$). La complexité de la construction de la matrice M est donc de l'ordre de $n_a^2 \sum_{e \in \mathcal{E}_a} \sum_{k \in \mathcal{K}^e} n(e,k)$.

Les calculs des matrices M^0 et M_r nécessitent respectivement $n_a \times n$ et n opérations élémentaires.

Une fois les matrices du système réduit calculées l'algorithme aura à résoudre un système linéaire à n^2 variables et contraintes.

8.3.2 Comparaison avec l'approche directe

L'importance de la taille du système réduit (8.2.12), qui peut atteindre n^2 , et la complexité de la construction de sa matrice, nous ont amené à comparer l'approche par décomposition avec une méthode directe.

La complexité de la résolution d'un système réduit (8.2.12) serait donc de l'ordre de n^6 dans le pire des cas. Rappelons que pour le problème de routage [15], nous avons utilisé la même approche par décomposition. La taille du système réduit était dans ce cas égal au nombre d'arcs du réseau (n). L'importance du nombre de flot K devant la taille du système réduit est la raison principale du succès de la méthode dans le cas du routage.

Nous remettons en question ici, l'efficacité de cette méthode uniquement pour PSG (8.1.1).

En adoptant la méthode directe, l'algorithme prédicteur-correcteur résout un problème d'optimisation linéaire de grande taille qui a pour matrice de contraintes la matrice

\hat{A} :

$$\left(\begin{array}{cccccccc} \omega^1 & & & & & & & \\ & \ddots & & & & & & \\ & & \omega^K & & & & & \\ \pi_1^1 & & & \omega^{1,1} & & & & \\ & \ddots & & & \ddots & & & \\ & & \pi_1^K & & & \omega^{1,K} & & \\ & & & \pi^{1,1} & \dots & \pi^{1,K} & Id(n) & -\hat{\Delta}_1 \\ \vdots & & \vdots & & & & \ddots & \vdots \\ \pi_n^1 & & & & & & \omega^{n,1} & \\ & \ddots & & & & & & \ddots \\ & & \pi_n^K & & & & \omega^{n,K} & \\ & & & \pi^{n,1} & \dots & \pi^{n,K} & Id(n) & -\hat{\Delta}_n \end{array} \right)$$

Cette matrice est de taille

$$\hat{p} \times \hat{n} = \left[K + \sum_{k=1}^K n(k) + n \times n_a \right] \times \left[\sum_{k=1}^K n^k + \sum_{e \in E^k} n^{e,k} + n(n_a + 1) \right]$$

($=O(n^4 + n^2 \sum_{k=1}^K n(k))$) mais elle est très creuse. En effet, elle a au plus

$$\hat{m} = \sum_{k=1}^K n^k + \sum_{e=1}^n \sum_{k \in \mathcal{K}^e} (n^{e,k} + n^k + n(e, k) \times n^{e,k}) + 2n \times n_a - n_a$$

($=O(n^2 + n \sum_{k=1}^K n(k))$) éléments non nuls ($\in \{1, -1\}$).

L'ordre de grandeur des valeurs de \hat{p} , \hat{n} et \hat{m} dépend de celui de n et K .

Au cours de cette étude nous évoquons les deux problèmes types suivants :

1. Le problème de sécurisation d'un réseau à graphe complet où à toute paire de nœuds nous associons une demande ($K = n = p(p-1)/2$). Dans ce cas la taille de la matrice des contraintes \hat{A} et la valeur de \hat{m} sont respectivement de l'ordre de n^4 et $n \sum_{k=1}^K n(k)$ ($O(n^2 \sqrt{n})$). Nous appellerons ce problème PB1.
2. Dans le deuxième type de problème, le graphe est très peu maillé et est parcouru par un grand nombre de flots. Un exemple de ce genre de problème est celui où $n = p$ et $K = p(p-1)/2$. Nous appellerons ce problème PB2. Pour PB2, l'ordre de grandeur de la taille de la matrice des contraintes est n^5 et celui de \hat{m} est n^4 .

Pour PB2 la complexité de calcul de la matrice M est de l'ordre de n^6 .

	Directe	Par décomposition
taille du système notée N	$K + 2 \sum_{k=1}^K n^k +$ $\sum_{e \in \mathcal{E}_a} \sum_{k \in \mathcal{K}^e} (2n^{e,k} + 1) +$ $2(n_a + 1) \times n$	$n_a \times n$
Complexité	$O(N^3)$	$O(N^3) +$ $\sum_{e \in \mathcal{E}_a} \sum_{k \in \mathcal{K}^e} n(e, k) \times n_a^2$
Creux de la matrice noté \hat{m}	$O(n^2 + n \sum_{k=1}^K n(k))$	pleine $O(n^4)$
PB1	$N = O(n_a \times n)$ $\hat{m} = O(n^2 \sqrt{n})$ à suivre	$N = O(n_a \times n)$ à éviter
PB2	$N = O(n^3)$ $\hat{m} = O(n^4)$	$N = O(n_a \times n)$

Table 8.3.1: Comparaison des deux approches

Dans le cas des problèmes de type PB1 la décomposition est inutile puisque nous obtenons le même ordre de complexité ($O(n^6)$) avec la méthode directe. En plus la méthode directe a une matrice de contraintes creuse. Pour les problèmes où le graphe est moins maillé la complexité de calcul du système de Newton dans la méthode directe peut être de l'ordre de n^9 . Par contre la méthode avec décomposition garde le même ordre de complexité (n^6).

Notre décision d'utiliser l'une de ces deux méthodes dépend du type de problèmes à résoudre (voir table (8.3.1)).

8.4 Réduction de la taille du système réduit

La résolution des systèmes réduits (8.2.12) constitue la partie la plus coûteuse de l'algorithme prédicteur-correcteur.

La complexité du calcul de la matrice du système réduit ($O(n^6)$) et la taille de ce système (d'ordre n^2) nous ont amené à intégrer une méthode de génération de colonnes pour réduire la taille du système réduit.

L'idée est de ne tenir compte que des contraintes de capacités de réserve actives et de rajouter au fur et à mesure les contraintes les plus violées.

L'intérêt de cette méthode par rapport à une méthode directe (simplexe) réside dans la possibilité de remplacer les n^2 contraintes de capacités de réserve par n contraintes. En effet, nous associons à chaque arc (i) du réseau la panne (ϵ) qui le sature ($x_i^{\epsilon,0} = 0$). Cette nouvelle gestion des arcs du réseau pourrait imposer l'utilisation de la formulation sommets-arcs.

Nous allons tout de même présenter cette technique de réduction du système réduit sous formulation arcs-chemins.

Rappelons le système d'optimalité associé au problème de sécurisation globale :

$$\left\{ \begin{array}{l}
 k \in \{1, \dots, K\} \left\{ \begin{array}{l}
 x^k s^k = 0, \\
 \omega^k x^k = r^k, \\
 \pi^{kT} c + \omega^{kT} \alpha^k - \pi^{kT} \beta^k = s^k, \\
 x^k, s^k \geq 0,
 \end{array} \right. \\
 \\
 e \in E \left\{ \begin{array}{l}
 x^{\epsilon,0} s^{\epsilon,0} = 0, \\
 x^{\epsilon,0} + \sum_{k=1}^K \pi^{\epsilon,k} x^{\epsilon,k} = \hat{\Delta}_e z, \\
 \lambda^\epsilon = s^{\epsilon,0}, \\
 x^{\epsilon,0}, s^{\epsilon,0} \geq 0,
 \end{array} \right. \quad (*) \text{ Système réduit} \\
 \\
 k \in \{1, \dots, K\} \left\{ \begin{array}{l}
 x^{\epsilon,k} s^{\epsilon,k} = 0, \\
 \omega^{\epsilon,k} x^{\epsilon,k} = \pi_e^k x^k, \\
 \omega^{\epsilon,kT} \beta_e^k + \pi^{\epsilon,kT} \lambda^\epsilon = s^{\epsilon,k} \\
 x^{\epsilon,k}, s^{\epsilon,k} \geq 0,
 \end{array} \right. \\
 \\
 tz = 0, \\
 d - \sum_{\epsilon=1}^n \hat{\Delta}_e^T \lambda^\epsilon = t, \\
 z, t \geq 0.
 \end{array} \right. \quad (8.4.1)$$

A toute panne d'arc active ϵ et tout arc du graphe i nous associons respectivement les ensembles E^ϵ et E_i définis par :

- E^e : l'ensemble des arcs saturés par le reroutage des demandes créées par la panne de l'arc e à l'optimum.
- E_i : l'ensemble des pannes qui saturent l'arc i pendant le reroutage à l'optimum.

le système d'optimalité (8.4.1) est alors équivalent au système suivant :

$$\left\{ \begin{array}{l}
 k \in \{1, \dots, K\} \left\{ \begin{array}{l}
 x^k s^k = 0, \\
 \omega^k x^k = r^k, \\
 \pi^{kT} c + \omega^{kT} \alpha^k - \pi^{kT} \beta^k = s^k, \\
 x^k, s^k \geq 0,
 \end{array} \right. \\
 \\
 e \in E \left\{ \begin{array}{l}
 \left. \begin{array}{l}
 x_i^{e,0} = 0, \\
 \sum_{k=1}^K \pi_i^{e,k} x^{e,k} = z_i, \\
 \lambda_i^e \geq 0,
 \end{array} \right\} (*) \quad i \in E^e \\
 \left. \begin{array}{l}
 x_i^{e,0} > 0, \\
 x_i^{e,0} + \sum_{k=1}^K \pi_i^{e,k} x^{e,k} = z_i, \\
 \lambda_i^e = 0,
 \end{array} \right\} (**) \quad i \notin E^e \\
 \\
 k \in \{1, \dots, K\} \left\{ \begin{array}{l}
 x^{e,k} s^{e,k} = 0, \\
 \omega^{e,k} x^{e,k} = \pi_e^k x^k, \\
 \omega^{e,kT} \beta_e^k + \sum_{i \in E^e} \pi_i^{e,kT} \lambda_i^e = s^{e,k} \\
 x^{e,k}, s^{e,k} \geq 0,
 \end{array} \right. \\
 \\
 tz = 0, \\
 d_i - \sum_{e \in E_i} \lambda_i^e = t_i, i \in E \\
 z, t \geq 0,
 \end{array} \right. \quad (8.4.2)$$

Le sous-système (8.4.1.*) est à l'origine de l'importance de la taille du système réduit. La formulation (8.4.2) distingue entre les contraintes de capacité de secours actives et inactives et remplace (8.4.1.*) par deux sous-systèmes (8.4.2.*) et (8.4.2.**). Pour réduire encore la taille du système réduit, nous proposons d'éliminer les contraintes (8.4.2.**). Mais comme les ensembles E_i , $i \in \{1, \dots, n\}$ et E^e , $e \in \{1, \dots, n\}$

ne sont pas connus, nous devons munir l'algorithme d'une structure de programme maître et de programmes satellites. Le programme maître est consacré à la résolution d'un problème de sécurisation globale tenant compte uniquement des contraintes de type (8.4.2.*). L'ensemble de ces contraintes est remis à jour grâce aux programmes satellites. La solution du $j^{\text{ème}}$ programme maître vérifie le système suivant :

$$\left\{ \begin{array}{l}
 k \in \{1, \dots, K\} \left\{ \begin{array}{l}
 x^k s^k = 0, \\
 \omega^k x^k = r^k, \\
 \pi^{kT} c + \omega^{kT} \alpha^k - \pi^{kT} \beta^k = s^k, \\
 x^k, s^k \geq 0,
 \end{array} \right. \\
 \\
 e \in E \left\{ \begin{array}{l}
 \left. \begin{array}{l}
 x_i^{e,0} \lambda_i^e = 0, \\
 x_i^{e,0} + \sum_{k=1}^K \pi_i^{e,k} x^{e,k} = z_i, \\
 x_i^{e,0}, \lambda_i^e \geq 0,
 \end{array} \right\} (*i \in E_j^e \\
 \\
 k \in \{1, \dots, K\} \left\{ \begin{array}{l}
 x^{e,k} s^{e,k} = 0, \\
 \omega^{e,k} x^{e,k} = \pi_e^k x^k, \\
 \omega^{e,kT} \beta_e^k + \sum_{i \in E_j^e} \pi_i^{e,kT} \lambda_i^e = s^{e,k} \\
 x^{e,k}, s^{e,k} \geq 0,
 \end{array} \right. \\
 \\
 tz = 0, \\
 d_i - \sum_{e \in E_{i_j}} \lambda_i^e = t_i, i \in E \\
 z, t \geq 0,
 \end{array} \right. \quad (8.4.3)$$

L'optimalité du problème de sécurisation globale est atteinte si en plus, pour toute panne $e \in E$ et pour tout arc $i \notin E_j^e$ nous avons la positivité de $x_i^{e,0} = z_i - \sum_{k=1}^K \pi_i^{e,k} x^{e,k}$. Si l'un des arcs $i \notin E_j^e$ ne remplit pas cette condition, la contrainte de capacité de réserve correspondante doit être prise en compte.

8.5 Conclusion

Les résultats de complexité obtenus montrent l'efficacité de la méthode directe dans le cas des graphes complets ($n = p(p - 1)/2$). La méthode avec décomposition n'est pas conseillée pour ce type de problème mais elle peut être intéressante dans le cas des réseaux peu maillés et traversés par un nombre important de flots.

Cette approche par décomposition de la méthode prédicteur-correcteur n'a pas été implémenté. Les données du Cnet sont du type PB1. Nous avons préféré utiliser une méthode directe qui tient compte du creux de la matrice des contraintes. Pour résoudre les systèmes de déplacement de Newton nous avons besoin d'un programme de Cholesky creux. Ceci est disponible dans la bibliothèque SPARSEPACK de NETLIB. Il existe déjà plusieurs codes de méthodes de points intérieurs utilisant un programme de Cholesky creux. Nous citons par exemple :

- **PCx** [26]
<http://www-fp.mcs.anl.gov/otc/Tools/PCx>,
- **BDMPD** [97, 96]
<http://www.sztaki.hu/meszaros/bmpd>,
- **HOPDM** [47, 48, 49]
<http://www.maths.ed.ac.uk/gondzio/software/hopdm.html>.

Dans le chapitre suivant, nous mettons en œuvre le code HOPDM pour résoudre les programmes maîtres.

9. Sécurisation par HOPDM

9.1 Motivation

D'après l'étude de complexité réalisée au chapitre précédent, la résolution directe du programme maître a le même ordre de complexité que la résolution avec décomposition (décrite au chapitre précédent), dans le cas des problèmes de réseau à graphe complet. Le code HOPDM (Higher Order Primal-Dual Method) implémente une méthode de points intérieurs primale-duale avec de multiples pas de correction. De plus, HOPDM offre plusieurs options utiles pour choisir un bon point de départ et pour exploiter le creux de la matrice des contraintes. Les problèmes satellites sont des problèmes de recherche de plus court chemin tout comme dans le cas des autres méthodes proposées.

Dans [48] Gondzio introduit une méthode de points intérieurs qui utilise des corrections multiples. Cette méthode cherche une direction de correction qui améliore la centralité de l'itéré courant et augmente le pas dans les espaces primal et dual. La méthode présentée par Gondzio traduit l'idée de Jansen, Roos, Terlaky et Vial dans [70]. Atteindre un pas plus grand dans les espaces primal et dual permet de réduire plus rapidement la non réalisabilité primale et duale et accélère la convergence. L'amélioration de la centralisation de l'itéré courant augmente les chances d'un grand pas d'être pris à l'itération suivante.

Le nombre maximal conseillé de corrections dépend du rapport de l'effort de résolution et de la factorisation du système Karush-Kuhn-Tucker. Le code HOPDM est muni d'une heuristique qui calcule ce rapport avant de lancer le processus d'optimisation de tout problème linéaire. L'utilisation des corrections multiples réduit le nombre d'itérations en moyenne de 25% à 40% en comparaison avec le prédicteur-correcteur du second ordre [93]. Cette réduction se traduit, d'après les essais numériques de [48], par un gain de 20% à 30% de temps cpu. Le code HOPDM est également muni de techniques de pré-résolution pour les problèmes linéaires creux de très grande taille.

9.2 Techniques de pré-résolution

Dans [49], Gondzio distingue trois groupes de techniques de pré-résolution.

Les techniques du premier groupe analysent répétitivement le problème linéaire. Elles éliminent les colonnes et les lignes vides ou singletons, cherchent les contraintes dominantes ou forçantes, serrent les bornes des variables et des prix cachés ou relâchent les bornes pour trouver les variables libres impliquées.

Le deuxième type d'analyse a pour but de réduire l'effort de calcul du facteur de Cholesky de la matrice de l'équation normale associée au problème. Gondzio propose une heuristique qui trouve la ligne pivot de la matrice des contraintes \hat{A} dont le modèle de creux est sous-ensemble de celui de toutes les autres. La ligne pivot est ensuite utilisée pour éliminer les composantes non nulles de toute ligne avec un sur-ensemble du modèle du creux. Cette heuristique est incapable de trouver 2 lignes avec des modèles de creux différents, telles que leur combinaison linéaire limite le remplissage et en même temps supprime plus de composantes non nulles. Cependant les résultats obtenus avec cette heuristique sont comparables à ceux des algorithmes plus compliqués.

Une colonne pleine dans la matrice \hat{A} crée un bloc complètement dense dans la matrice de l'équation normale ($\hat{A}^T D^2 \hat{A}$) et ses facteurs de Cholesky. Gondzio propose de décomposer une telle colonne en plusieurs petits morceaux et de remplacer le grand bloc dense par plusieurs blocs plus petits qui peuvent être facilement adaptés par la décomposition de Cholesky. L'applicabilité de la décomposition de colonnes pleines est réduite au cas des matrices qui ont un petit nombre de colonnes denses.

Le troisième groupe d'analyse de pré-résolution pose le problème de trouver les dépendances linéaires dans la matrice \hat{A} . La présence de lignes linéairement dépendantes cause une déficience du rang de la matrice $\hat{A}^T D^2 \hat{A}$. La dépendance linéaire des colonnes est en général moins dangereuse. L'heuristique que Gondzio propose pour rendre la matrice \hat{A} plus creuse, traite en même temps les lignes dupliquées. Les colonnes dupliquées sont identifiées après un effort d'analyse modéré (pour plus de détails voir [49]). Toutes ces techniques d'analyse ont été incorporées dans HOPDM. L'utilisateur peut choisir le niveau de l'analyse qui précède la résolution de son problème grâce à un paramètre LEVPRS. Pour plus de détails sur HOPDM nous faisons référence à la page web :

9.3.1 Gestion de chemins

PSG est un problème linéaire avec K contraintes de routage, n^2 contraintes de capacité de réserve et au plus $n \times K$ contraintes de reroutage (6.2.4).

Si l'ensemble des solutions de PSG n'est pas vide, alors PSG admet une solution dont au plus \tilde{p} ($\tilde{p} = K + n^2 + n \times K$) de ses composantes sont non nulles (corollaire 15.3 [13]).

Rappelons que parmi les composantes de cette solution, nous avons au plus n vecteurs de capacité résiduelle $x^{0,e} \in \mathbb{R}^n$.

Comme la capacité de réserve sur un arc donné doit être atteinte au moins pour une panne, nous savons qu'au moins n composantes correspondantes à des capacités de réserve résiduelles sont nulles. Le nombre de chemins utilisés à l'optimum est alors compris entre $\tilde{p} - n^2$ et $\tilde{p} - n$.

La résolution de PSG, nécessite au moins la génération de $\tilde{p} - n^2$ chemins. Cette estimation du nombre de chemins peut donner une idée sur les limites numériques de cette méthode.

Dans certains exemples, la croissance rapide du nombre de chemins générés représente un handicap pour cette méthode (voir les essais numériques).

Certains chemins générés aux premières itérations peuvent devenir inutiles. L'élimination de ces colonnes permet de libérer de la place mémoire.

Nous avons établi une méthode d'élimination de colonnes basée sur l'analyse de la solution. Nous éliminons les colonnes qui ont de "très grands" coûts réduits. Nous déclenchons cette procédure d'élimination chaque fois que le nombre de chemins générés dépasse une valeur critique, multiple de $\tilde{p} - n^2$.

Une colonne de la matrice des contraintes peut correspondre à un chemin nominal ou à un chemin de réserve ou à une composante de la capacité résiduelle de réserve associée à une panne. Il existe au plus n^2 colonnes correspondant au dernier type. Nous nous intéressons uniquement à la génération de chemins. L'élimination peut entraîner de grands changements dans la base de données. En effet, si le chemin est nominal, son élimination peut engendrer la désactivation d'une panne ou la non affectation d'un flot par un certain nombre de pannes. Si le chemin est de réserve, la procédure d'élimination est plus simple. Il suffit d'enlever le chemin de la liste des chemins de réserve.

Cette procédure nous a permis de résoudre des problèmes avec graphe complet de

plus de 12 nœuds.

9.3.2 Techniques d'élimination de colonnes

Pour préserver le creux de la matrice de Cholesky, calculée par la méthode de points intérieurs, il est essentiel de garder le nombre de colonnes le plus petit possible. Pour ce faire, nous pouvons utiliser une technique d'élimination de coupes inutiles (de colonnes inactives). Les travaux effectués dans ce cadre reposent sur des heuristiques. Un recueil de ces méthodes est présenté dans ce paragraphe.

Certaines colonnes générées aux premières itérations peuvent devenir inutiles et doivent être éliminées. Mais suivant quels critères et à quelle fréquence ? Les réponses à ces questions ne sont pas faciles.

En introduisant un terme quadratique de régularisation, comme dans la méthode de faisceaux [13], nous pouvons remplacer les colonnes éliminées par une colonne agrégée qui garde l'information nécessaire pour garantir la convergence.

Dans [43] Goffin et Vial proposent d'effectuer une seule élimination sévère (de coupes dans ACCPM) après un tiers du nombre total d'itérations estimé.

Dans [33] du Merle propose un test qui permet de détecter les coupes inutiles basé sur les ellipsoïdes inscrits (inclus dans l'ensemble de localisation) et exinscrits (englobant l'ensemble de localisation). Ce test consiste à estimer la position des coupes par rapport à la réduction homothétique de l'ellipsoïde exinscrit. L'étude numérique menée dans [33] montre que ce test induit souvent plus d'itérations extérieures et n'apporte en échange qu'une diminution négligeable du temps de calcul.

Dans [51], Gondzio et Sarkissian présentent une approche inexacte de la méthode de la génération de colonnes qui s'inspire de ACCPM [42] en utilisant les prix centraux. Leur approche utilise pour résoudre le programme maître (réduit) une méthode de points intérieurs primale duale, non réalisable qui emploie la notion du μ -centre pour contrôler la distance à l'optimalité. L'élimination dans [51], repose sur une analyse de la solution centrale donnée par le programme maître. Tous les produits de complémentarité $x_i s_i$ de cette solution restent proches du paramètre barrière final μ . Ils identifient les colonnes i telles que $x_i \leq \nu \mu^{1/2}$ et $s_i \geq (1/\nu) \mu^{1/2}$, avec $\nu < 1$ et supposent que les variables correspondantes x_i approchent zéro à l'optimum.

Cette technique d'élimination de colonnes peut être incorporée dans n'importe quelle

méthode de plans coupants centraux. En particulier, il est possible de l'incorporer dans la méthode de génération de colonne primale duale qui utilise HOPDM pour la résolution du programme maître [51].

9.4 Essais Numériques

Dans la suite de ce mémoire nous nous référons à l'algorithme décrit dans ce chapitre par DIP (Direct Interior Point). Nous recueillons dans cette section quelques résultats numériques qui résument le comportement de l'algorithme DIP.

9.4.1 Données générées

Pour une suite de graphes complets ($n = K = p/2 \times (p-1)$), nous observons l'évolution de la taille de la matrice des contraintes du dernier problème linéaire maître que HOPDM résout :

p	n	K	N	M	NZA
8	28	28	2699	1041	7697
9	36	36	4488	1653	12934
10	45	45	6342	2454	18184
11	55	55	9188	3550	26607

Table 9.4.1: Taille du dernier programme maître

Dans le tableau 11.2.6 N, M et NZA désignent respectivement le nombre de colonnes, de lignes et de composantes non nulles de la matrice des contraintes (du dernier programme maître que HOPDM résout).

Pour un graphe plein à 12 nœuds, l'algorithme s'arrête avec une précision (garantie relative d'optimalité de la solution) de 0.187 suite à la saturation de la mémoire. Pour le dernier programme maître, nous avons $N = 11852$, $M = 5249$ et $NZA = 33553$. Ceci nous a conduit à étudier et implémenter des stratégies de suppression de chemins. En éliminant les chemins qui ont un "très grand" coût réduit (supérieur à la moyenne des coûts), nous avons pu traiter des problèmes de sécurisation de graphe

complet à 12, 13 et 14 nœuds avec des précisions respectives de 10^{-10} , 10^{-9} et 10^{-4} . La courbe ?? décrit la précision obtenue par l'algorithme DIP sans activer la procédure d'élimination de chemins.

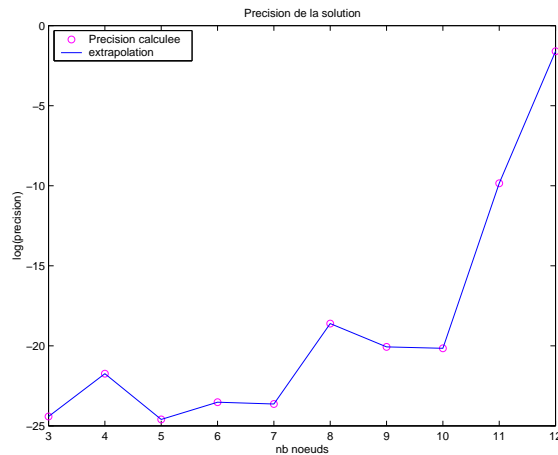


Figure 9.4.1: La précision obtenue avec DIP sans élimination de chemins

A partir de 15 nœuds, on ne peut avoir que des solutions approchées. Pour le graphe complet nous avons obtenu une précision de 6.10^{-2} . Nous avons également essayé de résoudre un problème de sécurisation d'un graphe à 20 nœuds, 190 arcs et 190 demandes. La matrice du dernier programme maître a 39401 colonnes, 36747 lignes et 81498 composantes non nulles. La précision que l'algorithme a pu réaliser avant la saturation de la mémoire est de 0.2.

9.4.2 Données du CNET

Nous présentons dans la suite les résultats numériques obtenus en testant DIP sur un problème réel fourni par le CNET *res-8*.

Il s'agit de la sécurisation d'un réseau à 8 nœuds, 28 arcs et traversé par 28 demandes. Les premières courbes sont obtenues en utilisant la routine de pré-résolution [49] (*lev-prs=6*). Le temps de calcul est, dans ce cas, 44 minutes. Le nombre total d'itérations internes est 260. Le nombre d'itérations externes est 20.

La figure 9.4.2 décrit la variation du coût calculé par HOPDM en fonction du nombre d'itérations majeures.

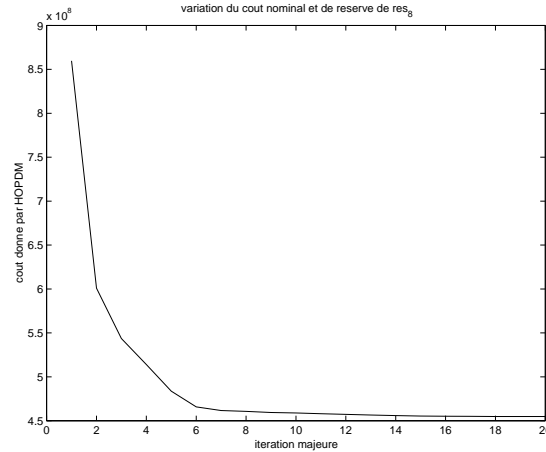


Figure 9.4.2: Variation de la valeur optimale du programme maître

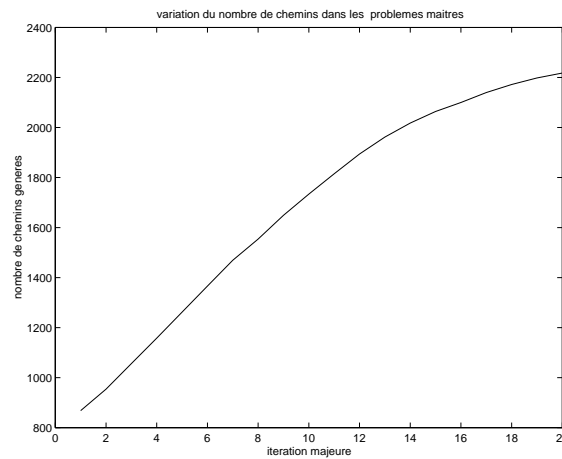


Figure 9.4.3: Variation du nombre de colonnes de la matrice des contraintes

Les figures 9.4.3 et 9.4.4 représentent respectivement la variation du nombre de colonnes et celle du nombre de lignes de la matrice des contraintes \hat{A} en fonction de celui des itérations majeures.

Nous remarquons que le nombre de contraintes se stabilise bien avant la convergence de l'algorithme (figure 9.4.4). Alors que le nombre de colonnes continue à croître.

La figure 9.4.5 confirme le creux de la matrice. La matrice des contraintes du dernier problème maître admet $N = 2218$ colonnes et $M = 939$ lignes et n'a que $NZA = 6809$ composantes non nulles.

Un autre avantage de cette méthode réside dans le fait que le nombre d'itérations

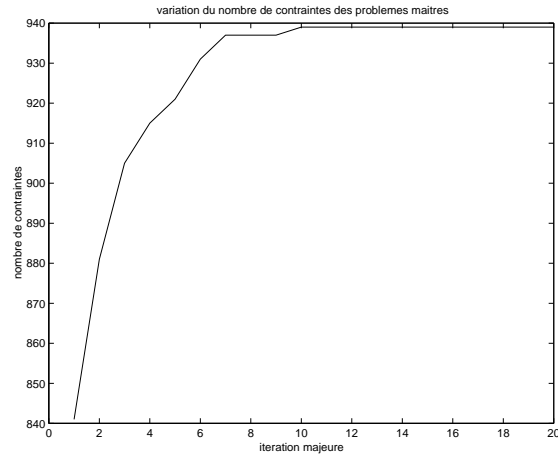


Figure 9.4.4: Variation du nombre de lignes de la matrice des contraintes

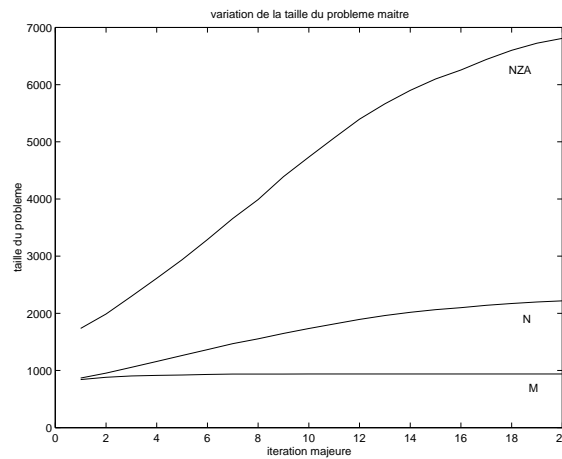


Figure 9.4.5: Variation de la taille du problème maître

effectuées par le programme maître reste petit. Pour ce problème nous avons en moyenne 13 itérations internes à chaque appel de HOPDM (figure 9.4.6).

Nous avons testé l'algorithme DIP sur le même problème (res-8), sans appeler la procédure de pré-résolution dans HOPDM (levprs=0).

Le nombre d'itérations a augmenté mais le temps de calcul était divisé par 3. En effet, l'algorithme DIP a convergé en 13 minutes (contre 44 au premier test) après 23 itérations externes (contre 20) et 301 itérations internes totales (contre 260).

La figure 9.4.7 représente les courbes de variation du coût calculé par HOPDM dans les deux cas (levprs=0 et Levprs=6).

Nous remarquons que les deux courbes de la figure 9.4.7 se superposent uniquement

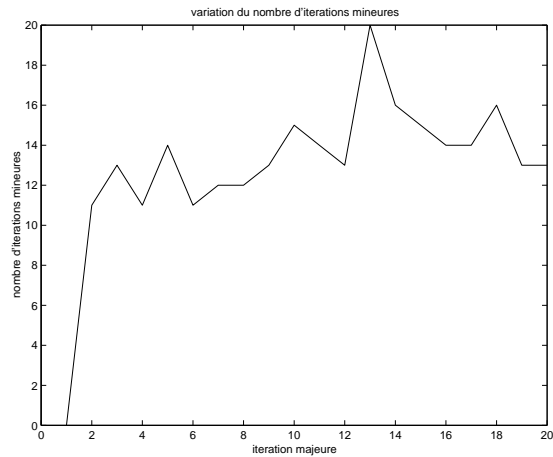


Figure 9.4.6: Variation du nombre d'itérations internes

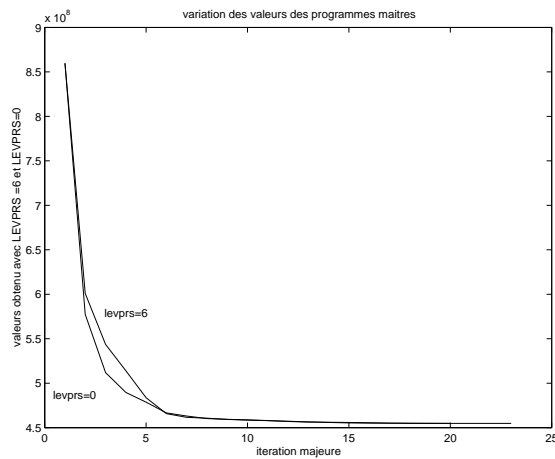


Figure 9.4.7: Comparaison de la variation du coût

aux premières et dernières itérations. En dehors de ces itérations, les valeurs calculées par HOPDM diffèrent. Ceci est dû à la sensibilité du centre analytique à la formulation du problème. En effet HOPDM traite deux formulations différentes du problème res-8. La première formulation obtenue après la procédure de pré-résolution est une version simplifiée de la deuxième. Les deux formulations restent tout de même équivalentes mais pas leurs sous-problèmes résolus par HOPDM.

Les multiplicateurs calculés à une même itération externe par HOPDM dépendent de la formulation de res-8. Par conséquent, les sous-ensembles de chemins générés peuvent être également différents.

La figure 9.4.9 confirme la différence des sous-ensembles de chemins générés dans

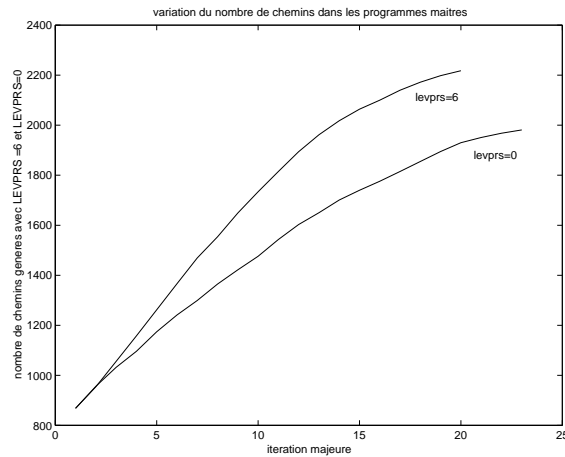


Figure 9.4.8: Comparaison de la variation du nombre de chemins

les deux tests.

La procédure de pré-résolution a augmenté la taille de la matrice des contraintes des

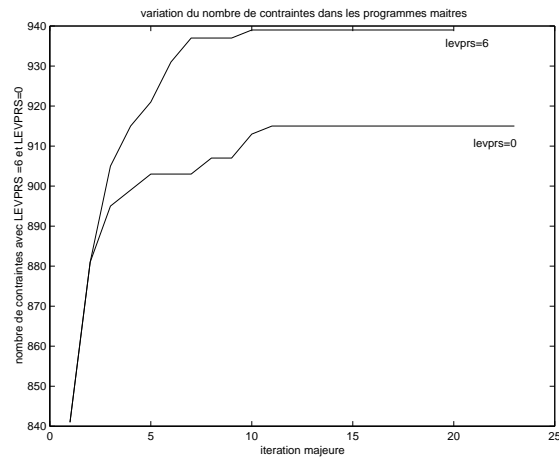


Figure 9.4.9: Comparaison de la variation du nombre de contraintes

programmes maîtres.

Sans pré-résolution le dernier programme maître admet $N = 1981$ variables (contre $N = 2218$ dans le premier) et $M = 915$ contraintes (contre $M = 939$). De plus le nombre de composantes non nulles de la matrice des contraintes correspondante est passé de $NZA = 6809$ au premier test à $NZA = 5970$ dans le second.

Nous finissons cette comparaison par la variation du nombre d'itérations internes (figure 9.4.10). Le nombre moyen d'itérations internes (effectuées par HOPDM) reste indifférent au niveau de pré-résolution (=13).

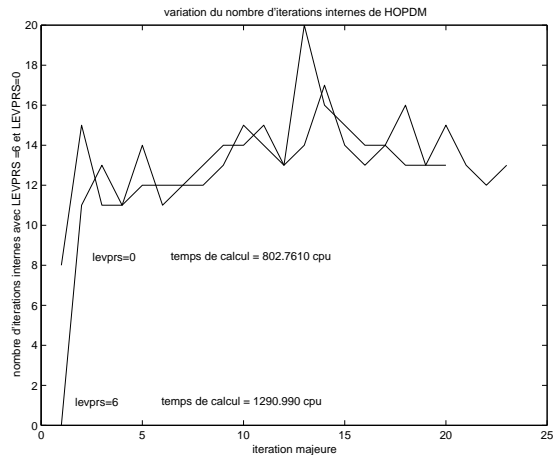


Figure 9.4.10: Comparaison de la variation du nombre d'itérations internes

Nous avons reporté ces tests uniquement pour montrer l'influence de la formulation sur la vitesse de convergence. La procédure de pré-résolution peut être très efficace sur d'autres problèmes, comme à titre d'exemple les réseaux symétriques.

9.5 Conclusion

Dans ce chapitre nous avons proposé une mise en œuvre de HOPDM pour la résolution de PSG. La croissance rapide du nombre de chemins générés nous a amené à intégrer une procédure délimitation de colonnes à DIP. Malgré ces efforts, cet algorithme générale n'a pas été capable de résoudre le problème PB15 (15 nœuds, 105 arcs et 105 demandes). L'importance de l'exploitation de la structure du problème pour sécuriser des réseaux de plus grande taille est indéniable.

10. Analyse numérique

Les algorithmes de sécurisation proposés dans ce mémoire ont été implémentés en FORTRAN 77. Chaque programme a été analysé numériquement dans le chapitre qui lui a été consacré.

Dans ce chapitre nous sélectionnons quelques résultats numériques obtenus avec DP2 et DIP. Nous verrons que la comparaison numérique de ces deux algorithmes conduit aux mêmes conclusions que celles effectuées pour le problème convexe de routage traité dans [111].

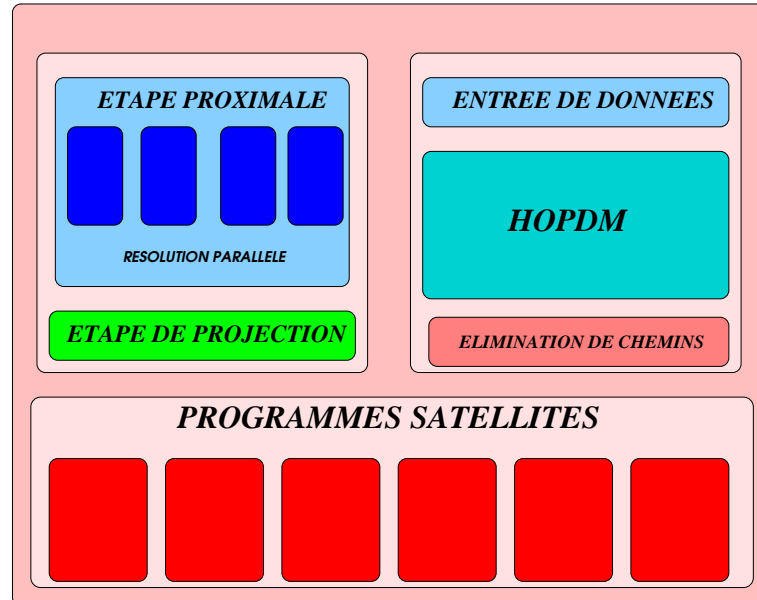


Figure 10.0.1: Organigramme

10.1 Précision de la solution

Pour un réseau symétrique à 12 nœuds, nous étudions numériquement l'influence du paramètre de la décomposition proximale Λ sur la précision de la solution fournie par DP2. Le nombre maximal d'itérations majeures *maxrouter* est fixé à 400. Pour toutes les valeurs de Λ testées, le programme DP2 s'arrête parce qu'il atteint 400 itérations externes. La précision demandée est de l'ordre de 10^{-6} . Nous rappelons que $\Lambda^{-1} =$

$\theta \times c_{max}$, avec $c_{max} = \max[\max\{D_i, i \in \{1, \dots, n\}\}, \max\{C_i, i \in \{1, \dots, n_n\}\}]$.

$10^3 \times \theta$	Λ	précision
3	33.3	0.115
3.5	28.57	0.101
4	25	8.06×10^{-2}
4.5	22.2	6.10×10^{-2}
5	20	3.7×10^{-2}
5.5	18.18	1.26×10^{-2}
6	16.66	1.43×10^{-5}
6.5	15.38	1.24×10^{-5}
7	14.28	1.62×10^{-5}
8	12.5	1.01×10^{-5}
9	11.11	1.29×10^{-5}

Table 10.1.1: Précision

La précision obtenue pour PB12 par DIP est de l'ordre de 10^{-9} . Le tableau 10.1.1 traduit bien l'importance du paramètre Λ . Une estimation du bon choix de ce paramètre dans le cas strictement convexe a été fournie par Mahey et al dans [91]. Le cas linéaire est encore une question ouverte.

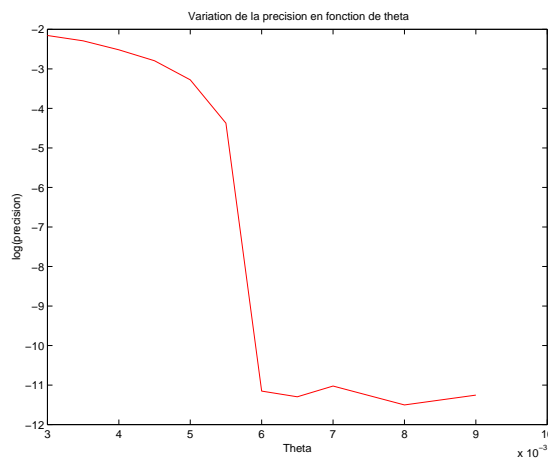


Figure 10.1.2: Variation de la précision en fonction de $\log(\theta)$

Nom	PB8	PB9	PB10	PB11	PB12	PB13	PB14	PB15
<i>DIP</i>	1.8E-9	1.3E-9	1.19E-11	3.47E-9	1.35E-9	7.16E-12	7.77E-10	0.000253
<i>DP2</i>	9.31E-7	9.83E-5	1.59E-6	1.59E-6	1.01E-5	1.16E-5	5.78E-6	7.49E-6

Table 10.1.2: Précision de la solution

Le tableau 10.1.2 compare les précisions obtenues par les deux algorithmes. De ce point de vue *DIP* était plus performant que *DP2*.

10.2 Génération de chemins

Pour une suite de données nous relevons le nombre de chemins générés dans *DIP* et *DP2*.

PB	DIP	DP2
8	623	196
9	1043	607
10	1485	5167
11	2270	878
12	4001	726
13	4295	936
14	3765	1183
15	> 4797	2809
16	X	1803

Table 10.2.3: Nombre de chemins générés : mip

Chaque *PB_i* correspond à un problème de réseau complet à i nœuds et $i(i-1)/2$ demandes.

Dans le problème de sécurisation, comme dans celui du routage, les chemins élémentaires qui reconstituent un flot ne sont pas uniques. Moins ces chemins sont nombreux, mieux l'algorithme se comporte. Dans [111], la méthode de décomposition proximale s'est montrée plus performante pour trouver une solution qui utilise peu de chemins en comparaison avec ACCPM. Nous avons abouti à la même conclusion empirique avec *DP2* (voir tableau 10.2.3).

A partir du problème PB12, DIP doit faire appel à une heuristique d'élimination de colonnes.

Les tableaux 10.2.4 et 10.2.5 donnent respectivement les tailles de IPP et IBFP. Pour plus de précision sur ces tableaux nous renvoyons le lecteur au chapitre suivant.

Nom	PB8	PB9	PB10	PB11	PB12	PB13	PB14	PB15
<i>DIP</i>	2634	4368	5986	8677	13265	13336	9767	11871
<i>DP2</i>	392	1214	10334	1756	1452	1872	2366	5618

Table 10.2.4: Taille du tableau des copies des chemins : mipp

Nom	PB8	PB9	PB10	PB11	PB12	PB13	PB14	PB15
<i>DIP</i>	228	320	383	469	850	1009	465	507
<i>DP2</i>	28	36	45	55	66	78	91	105

Table 10.2.5: Taille du tableau des copies des demandes : mibfp

Même si les tableaux de la structure de données (détaillée au chapitre suivant) sont de plus grande taille pour DIP, ceci n'explique pas l'encombrement de la mémoire pour le problème PB12. En effet, HOPDM utilise une factorisation de Cholesky qui est à l'origine de cet encombrement de mémoire. Cependant, HOPDM utilise une heuristique "Minimum degree ordering" pour réduire le plein du facteur de Cholesky. Pour PB12 la taille du facteur de Cholesky est 5.431.502 et pour PB15 il est de taille 6.847.216.

L'élimination de colonne peut préserver le creux du facteur de Cholesky jusqu'à la résolution de PB15.

10.3 Itérations

La comparaison du nombre d'itérations internes des deux algorithmes n'a pas de sens, vu la différence des opérations effectuées. Une itération externe (ou majeure) correspond à un appel du programme maître et à une nouvelle série de générations de chemins.

Nom	PB8	PB9	PB10	PB11	PB12	PB13	PB14	PB15
<i>DIP</i>	17	21	21	26	35	37	47	59
<i>DP2</i>	66	17	28	18	270	192	12	49

Table 10.3.6: Nombre d'itérations externes

Nom	PB8	PB9	PB10	PB11	PB12	PB13	PB14	PB15
<i>DIP</i>	150	204	214	283	349	389	469	544
<i>DP2</i>	573	2299	4233	1688	377	302	135	223

Table 10.3.7: Nombre total d'itérations internes

10.4 Temps de calcul

L'algorithme DP2 était plus rapide pour résoudre les problèmes testés mais moins précis.

Le tableau 10.4.8 compare les temps de calcul des deux algorithmes.

Nom	<i>DIP</i>	<i>DP2</i>
PB8	132.315	25.253
PB9	893.001	255.916
PB10	3190.309	2870.345
PB11	15882.353	199.788
PB12	28934.772	171.917
PB13	30360.794	231.636
PB14	23216.124	156.055
PB15	54402.129	353.194

Table 10.4.8: Temps de calcul en secondes

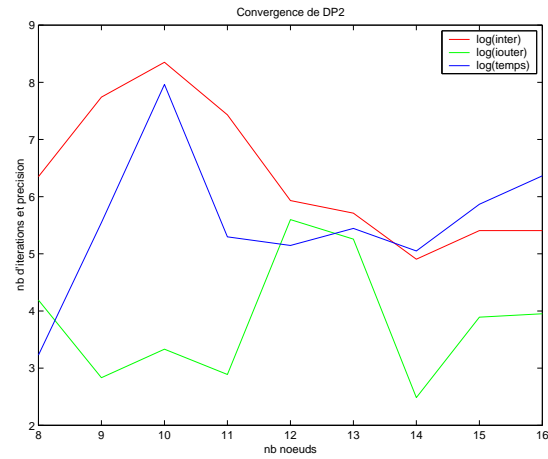


Figure 10.4.3: Variation de la vitesse de convergence de DP2 en fonction de la taille du problème

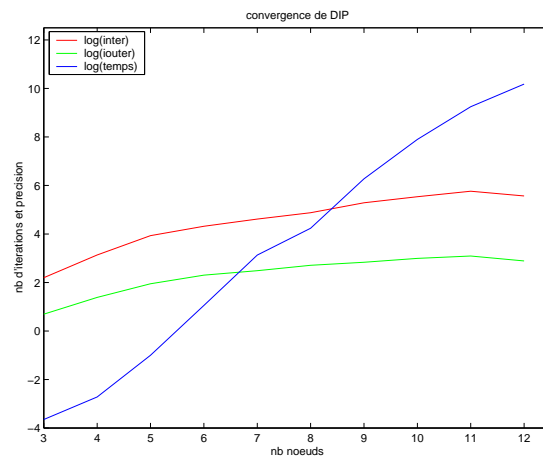


Figure 10.4.4: Variation de la vitesse de convergence de DIP en fonction de la taille du problème

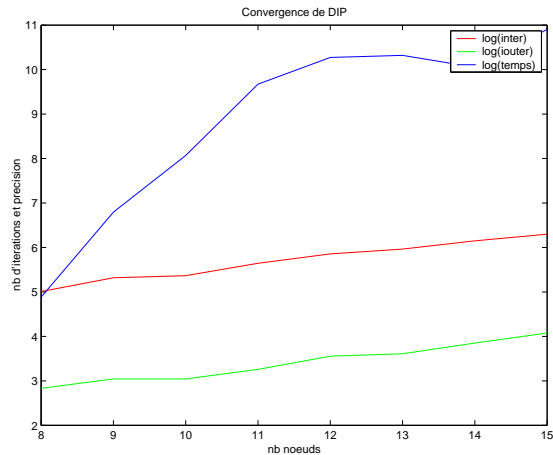


Figure 10.4.5: Variation de la vitesse de convergence de DIP avec élimination de chemins

Le tableau 10.4.9 résume les résultats obtenus pour avoir une vue plus globale sur les performances des deux algorithmes.

PB	mip1	mip2	temps1	temps2	prec1	prec2
8	623	196	132.315	25.253	1.82E-009	9.31E-007
9	1043	607	893.001	255.916	1.37E-009	9.83E-005
10	1485	5167	3190.309	2870.345	1.19E-011	1.59E-006
11	2270	878	15882.353	199.788	3.47E-009	1.59e-06
12	4001	726	28934.772	171.917	1.35E-009	1.01E-005
13	4295	936	30360.794	231.636	7.16E-012	1.16E-005
14	3765	1183	23216.124	156.055	7.77E-010	5.78E-006
15	> 4797	2809	54402.129	353.194	0.0002530792	7.49E-006
16	X	1803	X	580.797	X	8.02E-006

Table 10.4.9: Comparaison des performances de DIP et DP2

Dans le tableau 10.4.9 :

- mip1 : le nombre de chemins générés par DIP.
- mip2 : le nombre de chemins générés par DP2.
- temps1 : temps de calcul de DIP.

- temps2 : temps de calcul de DP2.
- prec1 : précision de la solution calculée par DIP.
- prec2 : précision de la solution calculée par DP2.

10.5 Conclusion

Ce chapitre constitue un recueil des essais numériques effectués avec les algorithmes DIP et DP2. L'algorithme DIP s'est montré plus précis que DP2, mais il ne peut sécuriser un réseau complet à 12 nœuds sans avoir recours à une procédure d'éliminant de chemins. Même en intégrant une heuristique d'éliminations de chemins, DIP n'a pas pu résoudre le problème PB15.

L'algorithme DP2 profite de la structure très particulière du problème, ce qui constitue un grand avantage par rapport à la méthode générale HOPDM.

D'après nos essais numériques, PD2 converge plus rapidement (avec un bon choix du Λ) et DIP est plus précis.

11. Annexe

11.1 Introduction

Ce chapitre peut être considéré comme un guide des programmes présentés dans la thèse.

Afin de gérer d'une façon optimale la base de données nous avons muni le programme d'une structure particulière. La génération, le classement et l'ajout d'un chemin par panne ou par flot doivent être rapides et négligeables par rapport aux autres tâches effectuées par le programme. L'idée est de répartir les informations de façon à y accéder facilement.

Nous détaillons dans la suite la fonction des tableaux suivants :

1. **ip(mcod+1,mipt)** : path,
2. **ipp(2,mippt)** : path-path,
3. **ifp(4,K)** : flow-path,
4. **ibfp(5,mibfpt)** : break-flow-path,
5. **ibf(n)**: break-flow,
6. **varx(2,mipt)** : routage moyen,
7. **vary(mippt)** : copies de routage et de reroutage,
8. **varz(nb+2,n)**: capacité moyenne et les copies de capacité,
9. **costp1(mipt)** : coût nominal,
10. **costp2(mipt)** :coût de réserve,
11. **lzlam(nk+1)** : pointeur sur les tableaux des multiplicateurs,
12. **izlam(mibft)** : lie les pannes au flot,
13. **izlam2(mibft)** : lie les contraintes au flot,

14. **zlam(mibft)** : les valeurs des multiplicateurs de Lagrange,

avec

- **n** est le nombre d'arcs du réseau,
- **p** est le nombre nœuds,
- **K** est le nombre de flots qui traversent le réseau,
- **nb** est le nombre de pannes ($nb \leq n$),
- **mipt** est le nombre maximal de chemins générés,
- **mippt** est le nombre maximal de copies de chemins nécessaires,
- et **mibfpt** est le nombre maximal de couples “panne-flot affecté”.

Le paramètre *mcod* sera défini dans la suite.

Les tableaux *varx*, *vary* et *varz* seront utilisés uniquement dans le cas de la décomposition proximale (DP1 et DP2).

11.2 Structure de données

11.2.1 Codage des chemins

Les codes DP1, DP2 et DIP utilisent la formulation arcs-chemins. L'ensemble des chemins générés constitue une information très importante qui doit être accessible. Le stockage des chemins en tant que vecteurs de \mathbb{R}^n est encombrant pour la mémoire. Afin de garder les informations importantes fournies par les chemins sans occuper beaucoup de place mémoire, nous proposons de coder ces chemins comme suit.

Un vecteur π de \mathbb{R}^n caractérisant un chemin du graphe \mathcal{G} est défini par :

$$(\pi)_e = \begin{cases} 1 & \text{si } e \text{ appartient au chemin,} \\ 0 & \text{autrement.} \end{cases} \quad (11.2.1)$$

Nous introduisons le paramètre *mcod* défini par

$$mcod = 1 + PE[(n - 1)/30]. \quad (11.2.2)$$

où $PE(q)$ désigne la partie entière de q . Nous associons à chaque chemin π un vecteur $icod$ de taille $mcod$ défini par:

$$icod_j = \sum_{i=30(j-1)+1}^{30j} 2^{i-1}. \quad (11.2.3)$$

A titre d'exemple, dans un graphe à 30 arcs, un chemin est identifié par un seul entier. Les $mcod$ premières lignes du tableau ip sont consacrées aux codes des différents chemins générés.

icod	vecteur code du chemin
nbc	nombre d'arcs qui forment le chemin

Table 11.2.1: $ip(mcod+1, mip)$, path

Comme le vecteur chemin n'est pas accessible sans décodage, nous gardons en mémoire :

- sa longueur, en dernière ligne de ip .
- son coût (nominal ou/et de réserve), dans un vecteur de taille mip ($costp1$ ou/et $costp2$) géré comme ip .

cost1	coût unitaire du chemin nominal
--------------	---------------------------------

Table 11.2.2: $costp1(mip)$, coût nominal

cost2	coût unitaire du chemin de réserve
--------------	------------------------------------

Table 11.2.3: $costp2(mip)$, coût de réserve

Nous classons les chemins dans le tableau ipp par ordre lexicographique. La première ligne de ce tableau donne la colonne correspondant au chemin dans le tableau ip . La deuxième ligne de ipp pointe sur le successeur de ce chemin dans ipp .

np	adresse du chemin dans le tableau ip
lnp	adresse de son successeur dans ipp

Table 11.2.4: $ipp(2, mipp)$: path-path

11.2.2 Classement des chemins par flot

Pour ajouter ou éventuellement éliminer un chemin, il faut avoir accès à la liste des autres chemins associés au même flot. Le tableau `ifp` fournit cette information. Il donne accès au premier chemin nominal et au premier chemin de réserve. Grâce au tableau `ipp` la suite de la liste pourrait être reconstituée. Le nombre de pannes affectant un flot et le nombre total de ces chemins nominaux sont aussi donnés par le tableau `ifp`. Le tableau `ifp` lie les flots aux chemins qui les portent.

lpn	adresse du 1 ^{er} chemin nominal associé au flot (dans <code>ipp</code>)
lpr	adresse du 1 ^{er} chemin de réserve associé au flot (dans <code>ipp</code>)
nbk	nombre de pannes qui affectent le routage du flot
ncn	nombre total de chemins nominaux générés

Table 11.2.5: `ifp(4,nk)`: flow-path

11.2.3 Classement des chemins par panne

Pour classer les chemins par panne, nous avons besoin de 2 tableaux en plus de `ipp`. Nous notons le premier `ibf` et le second `ibfp`.

Le tableau `ibf` lie les pannes aux flots qu'elles affectent. A chaque panne le tableau `ibf` associe l'adresse dans le tableau `ibfp` du premier flot affecté. Ce dernier fait le lien entre les flots affectés et les chemins correspondants dans `ipp`.

if	adresse du 1 ^{er} flot concerné par la panne <code>b</code> (dans <code>ibfp</code>)
nf	nombre de flots affectés par la panne <code>b</code>

Table 11.2.6: `ibf(2,n)`: break-flow

f	flot
lfs	adresse du flot successeur (dans ibfp)
lpn1	adresse du 1 ^{er} chemin nominal affecté (dans ipp)
lpr	adresse du 1 ^{er} chemin de réserve associé au flot f (dans ipp)
lpn2	adresse du 1 ^{er} chemin nominal non affecté (dans ipp)

Table 11.2.7: ibfp(5,mibfp): break-flow-path

11.2.4 Multiplicateurs de Lagrange

Les multiplicateurs de Lagrange calculés par le programme maître sont nécessaires au calcul de l'estimation inférieure et à la génération de chemins. L'accès à ces multiplicateurs doit se faire aussi bien par flot que par panne. C'est pour cette raison que nous utilisons 4 tableaux pour garder ces vecteurs en mémoire.

lzlam(k)	pointeur sur la première panne qui affecte le flot k
-----------------	--

Table 11.2.8: lzlam(nk+1), pointeur sur les tableaux des multiplicateurs

Pour un flot $k \in \mathcal{K}$ le tableau lzlam associe deux adresses : $l0=lzlam(k)$ et $le=lzlam(k+1)-1$ qui délimitent la partie qui concerne ce flot dans les autres tableaux :

- $izlam(mibft)$ lie les pannes aux flots.
- $izlam2(mibft)$ lie les contraintes aux flots.
- $zlam(mibft)$ donne les valeurs des multiplicateurs de Lagrange.

izlam(l0)	première panne affectant le flot k
izlam(le)	dernière panne affectant le flot k

Table 11.2.9: izlam(mibft), lie les pannes aux flots

izlam2(l0)	première contrainte de reroutage associée au flot k
izlam2(le)	dernière contrainte de reroutage associée au flot k

Table 11.2.10: izlam2(mibft), lie les contraintes aux flots

zlam(l0)	premier multiplicateur associé au flot k
zlam(le)	dernier multiplicateur associé au flot k

Table 11.2.11: zlam(mibft): lie les multiplicateurs aux flots

Ce choix de base de données nous facilite la gestion des chemins (l'ajout et l'élimination) ainsi que les pannes (active ou inactives).

11.3 Décomposition proximale

L'algorithme de décomposition principale alterne une étape proximale et une étape de projection :

- Étape proximale \rightarrow une solution $U = (u_x, u_z)$.
- Étape de projection :
 - Projection sur $\mathcal{A} \rightarrow (x, z)$.
 - Projection sur $\mathcal{A}^- \rightarrow (y, t)$.

A chaque itération l'algorithme de décomposition proximale garde en mémoire :

- le routage moyen x .
- $x + \lambda y$.
- la capacité moyenne de réserve z .
- $z + \lambda t$.

$varx(2, \text{mipt})$: projection du routage sur \mathcal{A}

Le calcul et le stockage du routage moyen se font grâce au tableau $varx$.

Comme on doit garder l'ancienne valeur du routage moyen, ce tableau comporte 2 lignes.

Le choix de la ligne de travail dépend de la parité de l'itération majeure.

Si j est pair, on modifie la ligne 1. La ligne 2 représente dans ce cas un multiple de la valeur du routage moyen à l'itéré précédent.

l'algorithme met à jour $varx$ à chaque calcul d'une copie du routage.

$2(n+1)\bar{x}$	$2 \times$ la somme des $(n+1)$ copies du routage
$2(n+1)x^{iter-1}$	$2(n+1) \times$ la valeur du routage à l'itéré précédent

Table 11.3.12: $varx(2, \text{mipt})$, projection du routage sur \mathcal{A}

La gestion du tableau $varx$ est similaire à celle du tableau ip .

 $vary(1, \text{mippt})$: projection du routage sur \mathcal{A}^-

On désigne par \bar{x} le projeté de la solution sur \mathcal{A} : $\bar{x} = \mathcal{P}_{\mathcal{A}}(u)$.

On a $v = y^{j-1} + \frac{x^{j-1} - u}{\lambda}$.

Soit \bar{y} le projeté de v sur \mathcal{A}^- : $\bar{y} = \mathcal{P}_{\mathcal{A}^-}(v)$.

On a alors $\lambda\bar{y} - \bar{x} = \lambda y^{iter-1} - u$.

L'égalité précédente justifie la méthode que nous avons adoptée pour stocker le projeté sur \mathcal{A}^- .

Cette mise à jour de $\lambda\bar{y} - \bar{x}$ peut se faire partiellement sans connaître la valeur de \bar{x} .

En effet : $\lambda\bar{y} - \bar{x} = \lambda y - x + x + u$.

$\lambda y - x$	projection sur \mathcal{A}^-
-----------------	--------------------------------

Table 11.3.13: $vary(1, \text{mippt})$, projection du routage sur \mathcal{A}^-

La gestion du tableau $vary$ est similaire à celle du tableau ipp .

$varz(nb+2,n)$: tableau des capacités de réserve

Les deux premières lignes sont équivalentes à $varx$.

Elles représentent les projetés des vecteurs de copies de la capacité de réserve sur \mathcal{A} aux itérés courant et précédent (selon la parité de l'itération).

Les nb lignes suivantes représentent l'équivalent du tableau $vary$.

Pour les mêmes raisons que pour $vary$, nous gardons ici les valeurs de $(\lambda t - z)$.

Nous rappelons que le nombre de pannes nb peut être inférieur au nombre d'arcs n .

$2(n-1)\bar{z}$	$2 \times$ la somme des n copies de capacité de réserve
$2(n-1)z^{iter-1}$	$2(n-1) \times$ la capacité de réserve à l'itéré précédent
$(\lambda t - z)(1)$	projection de la 1 ^{ère} copie de la capacité sur \mathcal{A}^-
\vdots	\vdots
$(\lambda t - z)(nb)$	projection de la $nb^{ième}$ copie de la capacité sur \mathcal{A}^-

Table 11.3.14: $varz(nb+2,n)$: tableau des capacités de réserve

11.4 Méthode du gradient réduit

Considérons le problème d'optimisation :

$$\begin{cases} \text{Min}_{x \in \mathbb{R}^n} f(x); \\ Ax = b, \\ x \geq 0, \end{cases} \quad (11.4.1)$$

où A est une matrice de taille $p \times n$, ($p \leq n$) et f est une fonction de \mathbb{R}^n à valeur dans \mathbb{R} .

On suppose que $\text{rang}(A) = p$.

On fait l'hypothèse que le problème (11.4.1) admet au moins une solution.

11.4.1 Principe de la méthode

Supposons que les variables x_i sont ordonnées de telle sorte que la matrice A peut s'écrire $A = [A_B A_N]$ avec A_B matrice d'ordre p inversible.

En respectant cette partition, le problème (11.4.1) peut s'écrire comme suit :

$$\left\{ \begin{array}{l} \text{Min}_{(x_B, x_N) \in \mathbb{R}^p \times \mathbb{R}^{n-p}} f(x_B, x_N); \\ \text{(i)} A_B x_B + A_N x_N = b, \\ \text{(ii)} x_B \geq 0, \\ \text{(iii)} x_N = 0, \end{array} \right. \quad (11.4.2)$$

On peut considérer x_N comme un vecteur de variables indépendantes et x_B comme un vecteur de variables dépendantes. En effet, lorsque x_N est fixé, x_B est unique et est donné par (11.4.2.i). Le principe de la méthode du gradient réduit est de considérer le problème (11.4.2) uniquement en terme de variables indépendantes x_N .

On a $Ax = b \iff A_B x_B + A_N x_N = b$.

Soit $x_B(x_N) = A_B^{-1}b - A_B^{-1}A_N x_N$.

On résout localement le problème :

$$\text{Min}_{x \in \mathbb{R}^n} g(x_N); \quad x_N \geq 0, \quad (11.4.3)$$

avec $g(x_N) = f(x_B(x_N), x_N)$.

On appelle *gradient réduit* le gradient de g :

$$\nabla g(x_N) = \nabla_N f(x_B(x_N), x_N) - (A_B^{-1}A_N)^t \nabla_B f(x_B(x_N), x_N).$$

11.4.2 Algorithme du gradient réduit

1. Initialisation : choix d'une base initiale et d'un vecteur initial $x^0 = (x_B^0, x_N^0)$ tel que $x_B^0 \geq 0$, $k = 0$.
2. Calcul du gradient réduit :
 - calcul de $\lambda (= \lambda(x_N))$ solution de :

$$\nabla_B f(x_B(x_N), x_N) + A_B^t \lambda = 0.$$

- On en déduit la valeur du gradient réduit :

$$\nabla g(x_N) = \nabla_N f(x_B(x_N), x_N) + A_N^t \lambda.$$

3. Calcul de la direction de descente : gradient réduit projeté

$$d_N^i = \begin{cases} -\nabla g^i(x_N) & \text{si } x_N^i \geq 0, \\ [-\nabla g^i(x_N)]^+ & \text{si } x_N^i = 0. \end{cases} \quad d_B = -A_B^{-1} A_N d_N$$

4. Recherche linéaire :

$$\text{Choisir } \rho^k \in]0, \rho_M[\text{ avec } \rho_M = \text{Min} \{ \rho_M^i / i \in \{1, \dots, n\} \} \quad \rho_M^i = \begin{cases} \infty & \text{si } d^i \geq 0, \\ -\frac{x^i}{d^i} & \text{sinon.} \end{cases}$$

5. Pivotage si c'est nécessaire.

6. $x^{k+1} = x^k + d^k$.

7. test d'arrêt et retour éventuel en [1].

Remarque 11.4.1. La méthode du gradient réduit est une extension de l'algorithme du simplexe aux problèmes quadratiques, qui répartit les variables en trois ensembles :

- les **variables de base**, assurant le respect des contraintes linéaires,
- les **variables hors-base**, bloquées à une borne,
- les **variables super-basiques** sont le reste des variables qui ne sont ni de base, ni bloquées à une borne.

Nous avons présenté la méthode du gradient réduit d'une façon sommaire. Pour plus de détails sur la convergence de cette méthode nous renvoyons le lecteur à [88].

Si on ne dispose pas d'une prédiction de l'ensemble des contraintes actives à l'optimum, la méthode du gradient réduit peut effectuer un grand nombre d'itérations. Dans la section suivante, nous proposons un point initial de l'algorithme qui exploite l'itéré précédent.

11.5 Initialisation du gradient réduit

La matrice des contraintes associée au problème quadratique correspondant à une panne donnée (active) est bloc-diagonale. Chaque bloc correspond aux deux contraintes (routage et reroutage) associées à un flot affecté par la panne.

Nous distinguons trois matrices de base possibles associées à un bloc de la matrice

des contraintes : $B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $B' = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$ et $B'' = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$.

Pour initialiser ces matrices de base, nous considérons comme première variable de base de chaque flot k touché par la panne, la variable qui correspond au chemin nominal qui porte le plus grand poids xn^k . Ensuite nous comparons cette première variable de base avec la variable qui correspond au chemin de réserve qui porte le plus grand poids xr^k :

- si xr^k est supérieur à xn^k la matrice de base est égale à
 - B si le chemin nominal qui porte xn^k ne passe pas par l'arc en panne,
 - B' sinon.
- si xr^k est inférieur strictement à xn^k , nous calculons $xn2^k$ qui est le plus grand poids porté par un chemin nominal de nature différente de celle du chemin qui porte xn^k . Autrement dit si le premier chemin passe par l'arc en panne, le deuxième doit l'éviter et inversement. Nous comparons ensuite les valeurs xr^k et $xn2^k$ pour déduire la base correspondante au bloc du flot k :
 - si xr^k est supérieur à $xn2^k$, la matrice de base est égale à
 - * B si le chemin nominal qui porte $xn2^k$ ne passe pas par l'arc en panne,
 - * B' sinon.
 - si xr^k est strictement inférieur à $xn2^k$, alors la matrice de base est B'' et les deux variables de base sont des variables de routage.

En suivant cette simple procédure d'initialisation de base nous exploitons l'information de l'itéré précédent et nous garantissons un bon point de départ pour l'algorithme de gradient réduit (opscq).

12. Conclusion

La première partie de cette thèse concerne une étude de robustesse des algorithmes de points intérieurs *prédicteurs-correcteurs*, ainsi qu'une approche par décomposition de cette méthode pour la résolution de *problèmes de multiflot*.

Dans la deuxième partie, nous nous sommes intéressés au *Problème de Sécurisation Globale* dont l'objectif est de déterminer le routage nominal et l'investissement de moindre coût en capacités *nominale* et *de réserve* qui garantit la survie de ce routage. Nous adoptons la stratégie globale pour rerouter le trafic interrompu.

Dans notre modèle les routages et les capacités peuvent être fractionnés. PSG se formule alors comme un problème linéaire de très grande taille avec plusieurs niveaux de couplage. Sa structure particulière fait appel à l'emploi d'algorithmes de décompositions.

Nous avons proposé quatre méthodes utilisant la technique de *générations de colonnes*. Les deux premières sont basées sur *les techniques proximales*. Leur tâche principale consiste en la résolution de sous-problèmes quadratiques indépendants.

Le troisième algorithme s'inspire de l'approche de points intérieurs décrite dans la première partie. Pour finir, nous avons intégré une procédure *d'élimination de chemins* dans une adaptation d'un solveur de points intérieurs HOPDM.

Nous avons utilisé le troisième algorithme pour résoudre des problèmes réels de routage fournis par le CNET.

L'implémentation de cet algorithme pour le problème de sécurisation est envisagée si le réseau est peu maillé.

Nous avons implémenté les trois autres algorithmes proposés dans cette thèse pour la résolution de PSG.

Une analyse numérique montre l'avantage de la méthode de décomposition proximale lorsqu'elle tire profit de la structure du problème.

Parmi les deux algorithmes basés sur cette méthode, seul DP2 s'est montré numériquement performant.

Cependant, cette performance est dépendante de la valeur du facteur de la décomposition proximale Λ . Nous avons tenté d'estimer la valeur optimale de ce paramètre en suivant une heuristique. La prévision théorique de ce paramètre en optimisation linéaire reste un problème ouvert.

DP2 se prête bien au parallélisme. Une mise en œuvre de DP2 sur des moyens de calcul parallèle est envisagée.

L'algorithme utilisant HOPDM, appelé DIP, était limité à cause de l'encombrement de la place mémoire. Nous espérons que ces limites de taille seront repoussées avec le développement croissant des moyens informatiques.

Cette thèse présente des méthodes prometteuses en optimisation de réseau de télécommunications. Nous espérons les étendre à d'autres problèmes. Une des extensions possibles de ce travail est la sécurisation globale dans le cas de pannes élémentaires de nœuds.

Références

- [1] Damage to fiber cable hinders phone service. Newark Star Ledger, 22 September 1987.
- [2] Fire in fiber gateway sparks flight delays, problems at brokerages. Wall Street Journal, 11 May 1988.
- [3] Phone snafu isolates New Jersey - long distance cable severed. Newark Star Ledger, 19 November 1988.
- [4] Ilan Adler, G. C. Resende, Geraldo Veiga, and Narendra Karmarkar. An implementation of Karmarkar's algorithm for linear programming. *Math. Programming*, 44(3 (Ser. A)):297–335, 1989.
- [5] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows*. Prentice Hall Inc., Englewood Cliffs, NJ, 1993. Theory, algorithms, and applications.
- [6] D. Alevras, M. Grötschel, and R. Wessäly. Capacity and survivability models for telecommunication networks. Technical Report SC-97-24, Konard-Zuse-Zentrum für Informationstechnik, Berlin, June 1997.
- [7] A. Ali, R.V. Helgason, J.L. Kennington, and H. Lall. Computational comparison among three multicommodity network flow algorithms. *Operations Research*, 28:995–1000, 1980.
- [8] E.D. Andersen, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In ed. T. Terlaky, editor, *Interior point methods in mathematical programming*, pages 189–252. Kluwer academic publishers, Dordrecht, the Netherlands, 1996.
- [9] A. Assad. Multicommodity network flows-a survey. *Networks*, 8:37–91, 1978.
- [10] F. Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6(3):823–837, 1996.

-
- [11] D. Bienstock, S. Chopra, O. Günlück, and C.Y. Tsai. Minimum cost capacity installation for multicommodity network flows. *Mathematical programming*, 1995. to appear.
- [12] D. Bienstock and O. Günlück. Computational experience with a difficult mixed-integer multicommodity flow problem. *Mathematical Programming*, 68:213–237, 1995.
- [13] J. Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia Sagastizábal. *Optimisation numérique*. Springer-Verlag, Berlin, 1997. Aspects théoriques et pratiques. [Theoretical and applied aspects].
- [14] J. Frédéric Bonnans and Clovis C. Gonzaga. Convergence of interior point algorithms for the monotone linear complementarity problem. *Math. Oper. Res.*, 21(1):1–25, 1996.
- [15] J.F. Bonnans, M. Haddou, A. Lisser, and R. Rébaï. Interior point methods with decomposition for multicommodity flow problems. Technical Report RR-3852, INRIA, Inria-rocquencourt, Janvier 2000. Telecom/BD/NT/Cnet/6573.
- [16] J.F. Bonnans, C. Pola, and R. Rébaï. Perturbed path following interior point algorithms. *OMS*, 2000. To appear.
- [17] J.F. Bonnans and F.A. Potra. Infeasible path following algorithms for linear complementarity problems. *Mathematics of Operations Research*, 22(2):378–407, 1997.
- [18] G. Bradley, G. Brown, and G. Graves. Design and implementation of large scale primal transshipment algorithms. *Management Science*, 24:1–34, 1977.
- [19] H. Brézis. *Opérateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert*, volume 5. Math, Studies, North-Holland, 1973.
- [20] J. Castro. A specialized interior point algorithm for multicommodity network flows. *SIAM J. Optimization*, to appear.
- [21] P. Chardaire and A. Lisser. Simplex and interior point specialized algorithms for solving non-oriented multicommodity flow problems. Technical report, CNET, 1997.

-
- [22] J. Chifflet, P. Mahey, and V. Reynier. Proximal decomposition for multicommodity flow problems with convex costs. *Telecommunication Systems*, 3:1–10, 1994.
- [23] In Chan Choi and Donald Goldfarb. Solving multicommodity network flow problems by an interior point method. In *Large-scale numerical optimization (Ithaca, NY, 1989)*, pages 58–69. SIAM, Philadelphia, PA, 1990.
- [24] N. Christophides and C.A. Whitlock. Network synthesis with connectivity constraint: A survey. *Operational Research*, pages 705–723, 1981.
- [25] S. Cosares, N.D. Deutch, I. Saniee, and O.J. Wasem. Sonet toolkit: A decision support system for designing robust and cost-effective fiber-optic networks. *Interfaces*, 25:20–40, 1995.
- [26] J. Czyzyk, S. Mehrotra, and S. Wright. Pcx user guide. Technical Report OTC 96/01, Optimization Technology Center, May 1996.
- [27] G. Dahl and M. Stoer. Multisun- mathematical model and algorithms. Technical Report TF R 46/92, Televerkets Forskningsinstitut, 1992.
- [28] G. Dahl and M. Stoer. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68:149–167, 1994.
- [29] G.B. Dantzig. The decomposition algorithm for linear programming. *Econometrica*, 29:767–778, 1961.
- [30] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programming. *Operations Research*, 8:101–111, 1960.
- [31] George B. Dantzig. Application of the simplex method to a transportation problem. In *Activity Analysis of Production and Allocation*, pages 359–373. John Wiley & Sons Inc., New York, N. Y., 1951. Cowles Commission Monograph No. 13.
- [32] George B. Dantzig. *Linear programming and extensions*. Princeton University Press, Princeton, NJ, corrected edition, 1998.

-
- [33] O. du Merle. *Points intérieurs et plans coupants : mise en œuvre et développement d'une méthode pour l'optimisation convexe et la programmation linéaire structuré de grande taille*. PhD thesis, Université de Genève, Faculté des Sciences Economiques et Sociales, 1995.
- [34] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. The Clarendon Press Oxford University Press, New York, second edition, 1989. Oxford Science Publications.
- [35] J.E. Eckstein. Parallel alternating direction multiplier decomposition of convex programs. *J. Optim. Theory Appl.*, 1:39–62, 1994.
- [36] L.R. Ford and D.R. Fulkerson. A suggested computation for maximal multi-commodity network flow. *Manag.Sci.*, 5:97–101, 1958.
- [37] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
- [38] B. Fortz, M. Labbe, and F. Maffioli. Two-connected network with bounded meshes. Technical Report IS-MG 96/8, Université Libre de Bruxelles, Campus de la plaine CP 210/01 Boulevard du triomphe B -1050 Bruxelles, Belgique, August 1996.
- [39] A. Frangioni. *Dual ascent methods and multicommodity flows*. Dip. di informatica, Univ. di Pisa, 1997.
- [40] Fred Glover and Darwin Klingman. The simplex SON algorithm for LP/embedded network problems. *Math. Programming Stud.*, (15):148–176, 1981. Network models and associated applications.
- [41] J.-L. Goffin, J. Gondzio, R. Sarkissian, and J.-P. Vial. Solving nonlinear multi-commodity flow problems by the analytic center cutting plane method. *Math. Programming*, 76(1, Ser. B):131–154, 1997. Interior point methods in theory and practice (Iowa City, IA, 1994).
- [42] J.-L. Goffin, A. Haurie, and J.-P. Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 38:284–302, 1992.

-
- [43] J.-L. Goffin and J.-P. Vial. Convex nondifferentiable optimization: a survey focussed on the analytic center cutting plane method. Technical Report 99.02, Logilab, Management Studies, University of Geneva, February 1999.
- [44] A.J. Goldman and A.W. Tucker. Polyhedral convex cones. In H.W. Kuhn and A.W. Tucker eds, editors, *Linear inequalities and related systems*, pages 19–40. Princeton University Press, Princeton, 1956.
- [45] R. E. Gomory and T. C. Hu. An application of generalized linear programming to network flows. *J. Soc. Indust. Appl. Math.*, 10:260–283, 1962.
- [46] M. Gondran and M. Minoux. *Graphes et Algorithmes*. Eyrolles, 1979.
- [47] J. Gondzio. Hopdm (version 2.12) - a fast lp solver based on a primal-dual interior point method,. *European Journal of Operational Research*, 85:221–225, 1995).
- [48] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6:137–156, 1996.
- [49] J. Gondzio. Presolve analysis of linear programs prior to applying an interior point method. *INFORMS Journal on Computing*, 9(1):73–91, Winter 1997.
- [50] J. Gondzio and M. Makowski. HOPDM modular solver for LP problems user’s guide to version 2.12. International Institute for Applied Systems Analysis, June 1995.
- [51] J. Gondzio and R. Sarkissian. Column generation with a primal-dual method. Technical Report 96.6, Logilab, Department of Management Studie, University of Geneva, Switzerland, June 1996.
- [52] C.C. Gonzaga. Path following methods for linear programming. *SIAM Review*, 34:167–227, 1992.
- [53] C.C. Gonzaga and R.A. Tapia. On the convergence of the Mizuno-Todd-Ye algorithm to the analytic center of the solution set. *SIAM J. Optim.*, 7(1):47–65, 1997.

-
- [54] C.C. Gonzaga and R.A. Tapia. On the quadratic convergence of the simplified Mizuno-Todd-Ye algorithm for linear programming. *SIAM J. Optim.*, 7(1):66–85, 1997.
- [55] M. D. Grigoriadis. An efficient implementation of the network simplex method. *Math. Programming Stud.*, (26):83–111, 1986. Netflow at Pisa (Pisa, 1983).
- [56] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In *Network models*, pages 617–672. North-Holland, Amsterdam, 1995.
- [57] M. Grötschel and C.L. Monma. Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM J. Discrete Math.*, 3:502–524, 1990.
- [58] M. Grötschel, C.L. Monma, and M. Stoer. Polyhedral approaches to network survivability. In *Reliability of Computer and Communication Networks*, volume 5 of *Discrete Mathematics and Computer Science*, pages 121–141. AMS/ACM, 1991.
- [59] M. Grötschel, C.L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):1992, 1992.
- [60] M. Grötschel, C.L. Monma, and M. Stoer. Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM Journal on Optimization*, 2(3):474–504, 1992.
- [61] G.M. Guisewite. *Handbook of global optimization*, volume 2 of *Nonconvex Optim. Appl.*, chapter Network problems, pages 609–648. Kluwer Acad. Publ., Dordrecht, 1995.
- [62] G.M. Guisewite and P.M. Pardalos. Minimum concave-cost network flow problems: applications, complexity, and algorithms. *Ann. Oper. Res.*, 25(1-4):75–99, 1990.
- [63] O. Güler. On the convergence of the proximal point algorithm for convex minimization. *SIAM J. Control Optim.*, 29(2):403–419, 1991.

-
- [64] O. Güler. New proximal point algorithms for convex minimization. *SIAM J. Optim.*, 2(4):649–664, 1992.
- [65] O. Güler. Ergodic convergence in proximal point algorithms with Bergman functions. In *Advances in optimization and approximation*, volume 1 of *Nonconvex Optim. Appl.*, pages 155–165. Kluwer Acad. Publ, Dordrecht, 1994.
- [66] M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.
- [67] F. Hitchcock. The distribution of a product from several sources to numerous localities. *J. Math. Phys. Mass. Inst. Tech.*, 20,:224–230, 1941.
- [68] S. Ibaraki and M. Fukushima. Primal-dual proximal point algorithm for multicommodity network flow problems. *J. Oper. Res. Soc. Japan*, 37(4):297–309, 1994.
- [69] A.N. Iusem. Some properties of generalized proximal point methods for quadratic and linear programming. *J. Optim. Theory Appl.*, 85(3):593–612, 1995.
- [70] B. Jansen, C. Roos, T. Terlaky, and J.-P. Vial. Long-step primal-dual target-following algorithms for linear programming. *Math. Methods Oper. Res.*, 44(1):11–30, 1996.
- [71] W.S. Jewell. Optimal flow through networks. Technical report, Massachusetts Institute of Technology, 1958.
- [72] K.L. Jones, I.L. Lustig, J.M. Farvolden, and W.B. Powel. Multicommodity network flows: The impact of formulation on decomposition. *Math. Prog.*, 62:95–117, 1993.
- [73] A.P. Kamath, N.K. Karmarkar, and K.G Ramakrishnan. Computational ans complexity results for an interior point algorithm on multicommodity flow problems. Technical Report TR-21/93, Dip. di Informatica, Univ. di Pisa, Italy, 1993. pp. 116-122.
- [74] S. Kapoor and P.M. Vaidya. Speeding up Karmarkar’s algorithm for multicommodity flows. *Mathematical programming*, 73:111–127, 1996.

- [75] N. K. Karmarkar and K. G. Ramakrishnan. Computational results of an interior point algorithm for large scale linear programming. *Math. Programming*, 52(3, Ser. B):555–586 (1992), 1991. Interior point methods for linear programming: theory and practice (Scheveningen, 1990).
- [76] J.E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.
- [77] Jeff L. Kennington and Richard V. Helgason. *Algorithms for network programming*. John Wiley & Sons, New York-Brisbane-Chichester, 1980. With a foreword by Fred Glover and Darwin Klingman, A Wiley-Interscience Publication.
- [78] J.F. Kennington. A survey of linear cost multicommodity network flows. *Oper.Res*, 2(26):209–236, 1978.
- [79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244(5):1093–1096, 1979.
- [80] Victor Klee and George J. Minty. How good is the simplex algorithm? In *Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)*, pages 159–175. Academic Press, New York, 1972.
- [81] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise. *A unified approach to interior point algorithms for linear complementarity problems*. Springer-Verlag, Berlin, 1991.
- [82] K.O. Kortanek, F. Potra, and Y. Ye. On some efficient interior point methods for nonlinear convex programming. *Linear algebra and its applications*, 152:169–189, 1991.
- [83] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956.
- [84] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In Neyman J, editor, *Proceeding of the second Berkeley symposium on mathematical statistics*

- and probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press.
- [85] C. Lemaréchal and J.-B. Hiriart-Urruty. *Convex analysis and minimization algorithms II : Advanced theory and bundle methods*, volume 306 of *Comprehensive studies in mathematics*. Springer-Verlag, Berlin, 1993.
- [86] A. Lisser, R. Sarkissian, and J.P. Vial. Optimal joint synthesis of base and reserve telecommunication networks. Technical report, CNET, Novembre 1995.
- [87] A. Lisser, R. Sarkissian, and J.P. Vial. Survivability in transmission telecommunication networks. Technical report, CNET, April 1995.
- [88] David G. Luenberger. *Introduction to linear and nonlinear programming*. Addison-Wesley., 1984.
- [89] Zhi-Quan Luo and Yinyu Ye. A genuine quadratically convergent polynomial interior point algorithm for linear programming. In *Advances in optimization and approximation*, pages 235–246. Kluwer Acad. Publ., Dordrecht, 1994.
- [90] T.L. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–156, 1995.
- [91] P. Mahey, S. Oualibouch, and P.D. Tao. Proximal decomposition on the graph of a maximal monotone operator. *SIAM J. Optimization*, 5(2):454–466, May 1995.
- [92] Jean-François Maurras and Yann Vaxès. Multicommodity network flow with jump constraints. *Discrete Math.*, 165/166:481–486, 1997. Graphs and combinatorics (Marseille, 1995).
- [93] S. Mehrotra. On the implementation of a (primal-dual) interior point method. *SIAM J. Optimization*, 4(2):575–601, 1992.
- [94] S. Mehrotra. Quadratic convergence in a primal-dual method. *Mathematics of Operations Research*, 18:741–751, 1993.

-
- [95] Sanjay Mehrotra and Jen-Shan Wang. Conjugate gradient based implementation of interior point methods for network flow problems. In *Linear and nonlinear conjugate gradient-related methods (Seattle, WA, 1995)*, pages 124–142. SIAM, Philadelphia, PA, 1996.
- [96] C. Mészáros. *The efficient implementation of interior point methods for linear programming and their applications*. PhD thesis, Eötvös Loránd university of sciences, 1996.
- [97] C. Mészáros. Fast Cholesky factorization for interior point methods of linear programming. *Comput. Math. Appl.*, 31(4-5):49–54, 1996. Selected topics in numerical methods (Miskolc, 1994).
- [98] M. Minoux. Synthèse optimale d’un réseau de télécommunication avec contraintes de sécurité. *Annales des télécommunications*, 36:211–230, 1981.
- [99] M. Minoux. Network synthesis and optimum network design problems: models, solution methods and applications. *Networks*, 19(3):313–360, 1989.
- [100] G.J. Minty. Monotone (nonlinear) operators in hilbert space. *Duke Math. J.*, 29:341–346, 1962.
- [101] G.J. Minty. On the monotonicity of the gradient of a convex function. *Pacific J. Math*, 14:243–247, 1964.
- [102] S. Mizuno, M.J. Todd, and Y. Ye. On adaptive-step primal-dual interior-point algorithms for linear programming. *Mathematics of Operations Research*, 18:964–981, 1993.
- [103] Shinji Mizuno. A new polynomial time method for a linear complementarity problem. *Math. Programming*, 56(1, Ser. A):31–43, 1992.
- [104] Shinji Mizuno, F. Jarre, and J. Stoer. A unified approach to infeasible-interior-point algorithms via geometrical linear complementarity problems. *Appl. Math. Optim.*, 33(3):315–341, 1996.
- [105] Shinji Mizuno and Florian Jarre. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Math. Program.*, 84(1, Ser. A):105–122, 1999.

-
- [106] C.L. Monma, B.S. Munson, and W.R. Pulleyblank. Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46:153–171, 1990.
- [107] C.L. Monma and D.F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4):153–171, 1989.
- [108] J.M. Mulvey. Testing of a large network optimization program. *Math. Programming*, 15:291–315, 1978.
- [109] Y.S. Myung, H.J. Kim, and D.W. Tcha. Design of communication networks with survivability constraints. *Management Science*, 45(2):238–252, February 1999.
- [110] A. Ouorou. *Décomposition proximale des problèmes de multiflot à critère convexe. Application aux problèmes de routage dans les réseaux de communication*. PhD thesis, Université Blaise Pascal, Ecole doctorale des sciences pour l'ingénieur de Clermont-Ferrand, Clermont-Ferrand, 1995.
- [111] A. Ouorou, P. Mahey, and J.Ph. Vial. A survey of algorithms for convex multi-commodity flow problems. *Management Science*, 46(1):126–147, January 2000.
- [112] P. Pardalos and H. Wolkowicz. Quadratic assignment and related problems. In P. Pardalos and H. Wolkowicz, editors, *Papers from the workshop held at Rutgers University*, volume 16 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, New Brunswick, New Jersey, 1994.
- [113] Panos M. Pardalos and Dingzhu Du, editors. *Network design: connectivity and facilities location*. American Mathematical Society, Providence, RI, 1998. Papers from the workshop held as part of the DIMACS Special Year on Networks at Princeton University, Princeton, NJ, April 28–30, 1997.
- [114] L. Portugal, F. Bastos, J. Júdice, J. Paixão, and T. Terlaky. An investigation of interior-point algorithms for the linear transportation problem. *SIAM J. Sci. Comput.*, 17(5):1202–1223, 1996.
- [115] L.-F. Portugal, M.G.C. Resende, G. Veiga, and J.J. Judice. A truncated primal-infeasible dual-feasible network interior point method. Preprint, to appear.

-
- [116] Luis Portugal, Luís Fernandes, and Joaquim Júdice. A truncated Newton interior-point algorithm for the solution of a multicommodity spatial equilibrium model. In *Complementarity and variational problems (Baltimore, MD, 1995)*, pages 315–344. SIAM, Philadelphia, PA, 1997.
- [117] Florian A. Potra. An $O(nL)$ infeasible-interior-point algorithm for LCP with quadratic convergence. *Ann. Oper. Res.*, 62:81–102, 1996. Interior point methods in mathematical programming.
- [118] R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.
- [119] R. Rébaï. *Optimisation de réseaux de télécommunications avec sécurisation*. PhD thesis, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny 75775 Paris cedex 16, 10 Février 2000.
- [120] Mauricio G. C. Resende and Panos M. Pardalos. Interior point algorithms for network flow problems. In *Advances in linear and integer programming*, pages 145–185. Oxford Univ. Press, New York, 1996.
- [121] R.T. Rockafellar. *Convex Analysis*. Princeton Univ. Press, 1970.
- [122] R.T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control and Optimization*, 14(5), August 1976.
- [123] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and algorithms for linear optimization*. John Wiley & Sons Ltd., Chichester, 1997. An interior point approach.
- [124] R. Sarkissian. *Telecommunications Networks: Routing and survivability optimization using a central cutting plane method*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, 1997.
- [125] J.E. Spingarn. Partial inverse of a monotone operator. *Appl. Math. Optim.*, 10:247–265, 1983.
- [126] K. Steiglitz, P. Weiner, and D. J. Kleitman. The design of minimum-cost survivable networks. *IEEE Trans. Circuit Theory*, CT-16:455–460, 1969.
- [127] M. Stoer. *Design of Survivable Networks*, volume Lecture Notes in Mathematics. Springer, 1992.

-
- [128] S.J. Wright. *Primal-dual interior-point methods*. SIAM, Philadelphia, PA, 1997.
- [129] T. Wu. *Fiber Network Survivability*. Artech House, Boston MA, 1992.
- [130] Y. Ye. On the q-order of convergence of interior-point algorithms for linear programming. In Wu Fang, editor, *Proceedings Symposium on Applied Mathematics*. Institute of Applied Mathematics, Chinese Academy of Sciences, 1992.
- [131] Y. Ye, O. Güler, R.A. Tapia, and Y.Y. Zhang. A quadratically convergent $O(\sqrt{n}L)$ iteration algorithm for linear programming. *Mathematical Programming*, 59:151–162, 1993.
- [132] S. Zenios. A smooth penalty function algorithm for network-structured problems. *European Journal of Operational Research*, 83:220–236, 1995.
- [133] G. Zorpette. Keeping the phone lines open. *IEEE Spectrum*, pages 32–36, June 1989.