629 | 2010

Quentin Botton

Survivable network design with quality of service constraints

Survivable network design with quality of service constraints

Extended formulations and Benders decomposition

Quentin Botton

Septembre 2010

Thèse présentée en vue de l'obtention du grade
de docteur en sciences économiques
et de gestion

Faculté des sciences
économiques, sociales, politiques et
de communication

Université catholique de Louvain

UCL

Université catholique de Louvain
Faculté des sciences économiques, sociales et politiques
Louvain School of Management

Survivable network design with quality of service constraints

Extended formulations and Benders decomposition

Thesis submitted in partial fulfillment of the requirements for the degree of Philosophiæ Doctor in Management Science by

Thèse présentée en vue de l'obtention du titre de Docteur en Sciences Économiques et de Gestion par

Quentin Botton

devant le jury composé de:

Promoteur:	Prof. B. Fortz (ULB)
Co-Promoteur:	Prof. P. Chevalier (UCL)
Président:	Prof. P. Agrell (UCL)
Lecteur:	Prof. M. Labbé (ULB)
Lecteur:	Prof. E. Loute (UCL-FUSL)
Lecteur:	Prof. L. Gouveia (DEIO-Portugal)

Louvain-la-Neuve, Septembre 2010

Quentin Botton: *Survivable network design with quality of service constraints, Extended formulations and Benders decomposition*, Thèse présentée en vue de l'obtention du titre de Docteur en Sciences Économiques et de Gestion, © Septembre 2010.

In memory of Raymond, my lovely grandfather.

"Il ne suffit pas d'engranger les récoltes du savoir, du savoir-faire,
ni de vendanger les fruits du savoir-être et du savoir-devenir,
encore faut il accepter de les offrir pour s'agrandir ensemble."

Jacques Salomé

Acknowledgements

This thesis is the result of a long-term process which started in September 2004. During this long and tumultuous adventure, I had the pleasure and the great honour to meet people who helped me somehow. I could not present this work without mentioning some important persons who have largely contributed to this work.

First of all, I would like to thank and to express my sincere gratitude to my supervisor **Bernard Fortz** for his support and his guidance during all these years. Our collaboration was an enriching experiment from the scientific perspective as well as from the human one. Bernard guided, gave precious advices, his opinion...without putting any pressure. He gave me the opportunity to travel around the world to share our works and to meet people through many renowned conferences. Thank you for all Bernard.

I would also like to thank all members of my Jury. First I thank **Per Agrell** and **Philippe Chevalier** from my department at LSM. Thank you Per for your precious comments as president of the Jury. Thank you Philippe for having accepted to be my co-supervisor and to have read the thesis in depth. I hope I have correctly integrated all your advices to improve the document. It is for me a real pleasure to work during these last 5-6 years with Per and Philippe inside the POMS first and now the CESCM department. Thanks to them and the others professors of the department I learned the rigour needed to achieve this scientific work. I also want to express my gratitude to **Etienne Loute**. Etienne gave me the first notions about the Benders decomposition in his operational research course in 2002. Along the thesis Etienne, as an active member of my monitoring committee, gave me precious advices. **Martine Labbé** another member of this committee welcomed me often at GOM department. I really appreciated Martine's comments, at ULB as well as in conferences. She precisely pointed out details which can make the difference. Last but not least, I deeply thank **Luis Gouveia** who welcomed me six months in his operational department in Lisbon. I met Luis in 2006 in the EURO conference in Iceland and it was an honour and a real pleasure to work with him. An honour because his list of publications is really impressive, and his quality of researcher is well-known all around the world. Luis collaborated in many high-quality publications in network design field. A pleasure because Luis is really someone to meet. I keep lots of souvenirs from my collaboration with Luis.

I would also like to address a special thank to **Yves Pochet** who was the professor who gave me the motivation to start that thesis. I did my master thesis with Yves and as student I tried to attend all Yves's courses. Its pedagogic and scientific skills are really fascistic.

My acknowledgements are also dedicated to all my colleagues from the LSM and from the GOM department: **Jean-Christophe, Mathias, Jean-Sébastien, Géraldine, Jean-Charles, Hakan, Daniele, Vincent, ...** and more especially **Michael Poss** from GOM with who I had the pleasure to collaborate. Thank you Michael for your precious help, for your availability and your kindness.

The "**Communauté française de Belgique**" as well as the "**ADONET program**" (Algorithmic Discrete Optimization Network, A Consortium of European Research Institutes on Discrete Optimization) are also thanked because these two organizations selected me for a grant to collaborate with the department of Luis Gouveia.

I want to thank my family for its permanent support, **my mother, my father and my two brothers (Gilles and Gérald)**. I know that it is not easy to motivate someone when it is difficult to really understand the subject of his work. But you offer me an adequate environment which comforted me as well as shaked when needed. Thank also to **Mr and Mrs Demoulin, Ben & Gé, Véro & Denis, Gégé & Miki and their children**.

I could not have achieved and presented this work if I had not met you in 2006. I have no words to express all things you bring me. You are the small bird on my shoulder which gives me each day the pleasure of life. An huge thank to you **Sébastienne**.

Finally, all my gratitude goes to all my friends: **Sam, Caro, Oli, Marie, Marc, Denis, Kévin, Barbara, Max, ...** thank you to be the branch of the my tree.

Enjoy your reading.

Abstract

In this thesis we study the survivable network design problem applied to the telecommunications industry. For obvious reasons, networks have to be survivable, in other words, they have to work even in case of failure (*edge or node failure*).

In a first part of this thesis, the objective is to determine for each edge of the network the optimal capacity to install allowing connectivity for each commodity. To ensure survivability, we assume that each edge of the given network, represented by a graph $G = (V, E)$, can break down independently and we impose that at most one edge can be broken at the same time. For each edge $ij \in E$ we consider an independent failure scenario in which we have to ensure connectivity in a new graph where the given broken edge ij is removed from the edge set E. We propose a formulation for this problem and a cutting plane algorithm based on the Benders decomposition to improve the time needed to reach optimality.

Then in a second part of this thesis, we consider the uncapacitated design problem in which the main objective is to determine the optimal topology (*the minimum cost topology*) of the network by choosing among all the potential edges, the ones on which to install capacity in order to send flows for a predetermined set of commodities. This time, to ensure survivability, we propose to find for each distinct commodity at least K edge-disjoint paths. This ensures that if $(K-1)$ edges fail, the network keeps the possibility to send demands on the remaining path.

Nowadays, due to the big expansion of telecommunications applications like audio, video, IP telephony,... survivability is no more the only key criteria for customers. New considerations linked to the quality of the service become more and more important. For these applications, the delay, which is closely related to the length of the paths used for routing, plays an important role. That is the reason why we propose to add new constraints bounding the length of the paths. Consequently we decide to analyze, in this thesis, the survivable network design problem with hop constraints.

Based on an alternative representation of the graph (*also called the layered graph*) we give an extended formulation for this kind of problem and we formulate and prove some nice properties for the single commodity case. Given the nice structure of the problem and the huge number of variables and constraints, we study also the multicommodity case using the Benders decomposition, some branch-and-cut algorithms and an heuristic to reduce the CPU time. Finally we study a variation of the problem where we consider two groups of edges, the

natural and the reliable edges. We show how this new problem becomes more complicated than the original one, we apply also the Benders decomposition and we conclude with an economical analysis of the impact of this new class of reliable edges.

Résumé

Au travers de cette thèse nous étudions le problème de conception de réseaux robustes appliqué aux réseaux de télécommunications. Concevoir un réseau robuste c'est définir parmi l'ensemble des arêtes potentielles, celles sur lesquelles l'installation de capacité permettra de véhiculer l'information à moindre coût et ce, y compris en cas de pannes potentielles.

Dans une première partie de cette thèse, l'objectif est de définir pour chaque arête du réseau quelle sera la capacité à installer permettant de véhiculer l'entièreté des demandes, et ce même en cas de pannes. Pour assurer la robustesse, nous proposons d'utiliser l'approche suivante : Pour un réseau donné, représenté par un graphe $G = (V, E)$, nous créons un scénario de panne pour chaque arête du réseau dans lequel l'arête en panne ne peut être utilisée. Pour chaque scénario de panne, nous créons donc un réseau fictif dans lequel il manque une arête (chaque fois différente). Nous proposons un modèle utilisant deux groupes de variables et étant donné le grand nombre de celles-ci, nous proposons une approche de plans coupants basée sur la décomposition de Benders afin d'améliorer le temps calcul.

Dans une seconde étude de cette thèse, pour assurer la robustesse, nous imposons que la topologie optimale du réseau doit contenir pour chaque paire origine-destination au moins K chemins arête-disjoints. Deux chemins sont, par définition, arête-disjoints lorsqu'ils ne partagent aucune arête en commun; il est dès lors possible de toujours router l'information du noeud d'origine au noeud de destination si un maximum de $(K-1)$ arêtes ne fonctionnaient plus.

De nos jours, et étant donné le large panel de services proposés aux utilisateurs (*téléphonie fixe ou mobile, télévision numérique, multimédia, transferts de données quelconques,...*) la fiabilité des réseaux n'est plus le seul critère pertinent aux yeux des consommateurs. D'autres considérations notamment liées à la qualité (*entre autres la rapidité*) du service, entrent en ligne de compte. Il est certain que de bénéficier d'un réseau fiable mais qui véhicule l'information avec des délais d'attente trop longs n'est pas avantageux pour le client. Il est dès lors primordial pour les gestionnaires de réseaux d'intégrer cet aspect lié au "délai d'attente" dans leur processus de conception de réseaux.

Au travers de cette thèse, nous analysons et modèlisons le problème de conception de réseaux fiables avec contraintes de sauts (*Qualité du Service*) en utilisant une description alternative du graphe original appelée "représentation en couches". Nous proposons et démontrons quelques propriétés intéressantes pour le cas simple commodité. Ensuite, les modèles pro-

posés utilisant un grand nombre de variables et de contraintes, nous étudions la performance de la décomposition de Benders ainsi que de divers algorithmes de type "branch-and-cut" pour le cas multi-commodités. Enfin nous étudions une variante du problème dans laquelle un sous-ensemble d'arêtes dites "fiables" sont intégrées au graphe d'origine. Nous montrons que ce nouveau problème est beaucoup plus compliqué que le problème original et nous concluons par une analyse économique à propos de l'impact de la présence de cette nouvelle catégorie d'arêtes sur la topologie optimale.

List of Publications

International Journal Papers

1. Q. Botton, B. Fortz, L. Gouveia and M. Poss, Benders decomposition for the hop-constrainted survivable network design problem, *Informs Journal on Computing*. Submitted.

Technical Reports

2. Q. Botton and B. Fortz, Decomposition Methods for the optimization of survivable telecommunications networks, *IAG Working Paper* 135/05, Université catholique de Louvain, 2005.

3. Q. Botton, B. Fortz, L. Gouveia and M. Poss, Benders decomposition for the hop-constrainted survivable network design problem, *CORE Discussion Paper*, March 2010. Submitted.

Forthcoming Papers

4. Q. Botton, B. Fortz and L. Gouveia, On the K edge (arc) disjoint L-paths problem: Properties of a layered extended formulation.

5. Q. Botton, B. Fortz, L. Gouveia and M. Poss, On the hop-constrained survivable network design problem with reliable edges.

Refereed International Conference Proceedings

6. Q. Botton and B. Fortz, Extended node-arc formulation for the hop-constrained network design problem. In *INOC 2007: International Network Optimization Conference*, Spa, Belgium, 2007.

International Conferences

7. Q. Botton and B. Fortz, Decomposition methods for the optimization of survivable telecommunications networks. In *ORBEL 20: Belgian National Operations Research Conference*, Gent, Belgium, 2006.

8. Q. Botton, and B. Fortz, Extended node-arc formulation for the hop-constrained network design problem. In *EURO XXI: 21st European Conference on Operational Research*, Reykjavik, Iceland, 2006.

9. Q. Botton, B. Fortz, Extended node-arc formulations for the K edge-disjoint hop-constrained network design problem. In *ORBEL 21: Belgian National Operations Research Conference*, Luxembourg, 2007.

10. Q. Botton, B. Fortz and L. Gouveia, Extended node-arc formulations for the K edge-disjoint hop-constrained network design problem. In Optimization 2007, sixth International Conference on Optimization, Porto, Portugal, 2007.

11. Q. Botton, B. Fortz and L. Gouveia, The K edge 3-hop constrained network design polyhedron. In *ORBEL 22: Belgian National Operations Research Conference*, Brussels, Belgium, 2008.

12. Q. Botton, B. Fortz and L. Gouveia, The K edge 3-hop constrained network design polyhedron. In *9th Informs Telecommunication Conference*, Baltimore, USA, 2008.

13. Q. Botton, B. Fortz and L. Gouveia, The K edge 3-hop constrained network design polyhedron. In *ORBEL 23: Belgian National Operations Research Conference*, Leuven, Belgium, 2009.

14. Q. Botton, B. Fortz and L. Gouveia, Properties of a layered extended graph formulation for designing K edge (arc)-disjoint L-paths. In *EURO XXIII, 23rd European Conference on Operational Research*, Bonn, Germany, 2009.

15. Q. Botton, B. Fortz and L. Gouveia, Properties of a layered extended graph formulation for designing K edge (arc)-disjoint L-paths. In *ISMP 2009: 20th International Symposium on Mathematical Programming*, Chicago, USA, 2009.

16. Q. Botton, B. Fortz, L. Gouveia and M. Poss, Benders decomposition for the hop-constrainted survivable network design problem. In *EURO XXIV: 24th European Conference on Operational Research*, Lisbon, Portugal, 2010.

List of Figures

2.1 Example of a multicommodity network with 2 demands 10
2.2 Solution without failure 11
2.3 Solution with one edge failure at a time 11
2.4 Optimal solution with failures 12
2.5 Example of network to design. 21
2.6 Optimal solution of the independent approach problem. 21
2.7 Optimal solution of the global approach problem. 22
2.8 Illustration of the testing phase. 23

3.1 Graph G (left), feasible stable set (centre), maximum stable set (right). . . . 28
3.2 Odd hole graph with 5 nodes on the left and the associated odd anti-hole graph on the right. 29
3.3 Representation of a claw. 30
3.4 Representation of a graph G and its line representation $L(G)$. 30
3.5 Seven intervals of the real line and the corresponding seven vertices interval graph G. 31
3.6 Example of Circular interval graph. 31
3.7 Original graph G (on the left) and its complementary graph $G^{'}$ (on the right). 32

4.1 Basic network (a) and its Layered representation (b) when $L = 3$ 39
4.2 Feasible solution for $K = 2$ and $L = 3$ in the layered graph $G^{'}$. 40
4.3 Feasible solution for $K = 2$ and $L = 3$ in the original graph G. 40
4.4 Feasible flow in the layered graph $G^{'}$. 41
4.5 Feasible flow in the layered graph is not systematically a feasible flow in the original graph. 42
4.6 Feasible arc-disjoint routing (a.) vs feasible edge-disjoint routing (b.) for $L = 3$ 44
4.7 Feasible arc-disjoint routing (a.) vs feasible edge-disjoint routing (b.) for $L = 4$ 44
4.8 Feasible arc-disjoint routing (a.) vs unfeasible edge-disjoint routing (b.) for $L = 5$ 45
4.9 Characteristics of paths when $L = 3$ 46
4.10 Layered representation when $L = 3$ 47
4.11 Feasible arc-disjoint solution (a) and feasible arc or edge disjoint solution (b). 49
4.12 Illustration of the first fractional routing case: loop between 2 nodes. 50
4.13 Illustration of the second fractional routing case: loop between 3 nodes. . . . 51

4.14 Arcs included in the inequality (4.13) 51
4.15 Illustration of the feasible configurations for inequality (4.13). 52
4.16 Layered graph $G^{'}$ and the reduced layered graph $G^{''}$ 53
4.17 Conflict graph G^c. 55
4.18 Neighbourhood of node ij in the alternative graph G^c. 55
4.19 Integer design with fractional routing. 61
4.20 Integer edge-disjoint routing $\neq$ integer edge-disjoint design with $c_{ij} \leq 0$. . . . 62
4.21 Fractional counter example when $L = 4$. 64
4.22 Different fractional paths when $L = 4$. 64

5.1 Basic network (a) and its layered representation (b) when $L = 4$ 70
5.2 Graph obtained from $\overline{z}$ 76
5.3 Fractional and integer minimum cost routing when $L = 4$ and $K = 2$. 77
5.4 Support graph a two-layered 4-path-cut inequality, inspired by Figure 4 from [62]. 78
5.5 Infeasible solutions for the edge THPP with $L = 5$, inspired by Figure 6 from [62]. 81
5.6 New inequality for $L = 5$. 81
5.7 `bc-n` depth parameter tuning by average CPU time (sec.) 84
5.8 Performance profile comparing methods on the entire test set. 85

6.1 Example of a bi-layered architecture. 92
6.2 Links between the various layers of a bi-layered network architecture. 93
6.3 Bi-layered architecture and impact of reliable edges. 95
6.4 Basic Network (a) and its Layered Representation (b) when $L = 4$ 96
6.5 Edges sets for variant HopR1 (left) and variant HopR2 (right). 98
6.6 Example of a network to connect: 1 source node and 5 destination nodes. . . 99
6.7 Example of optimal topology with model Hop: Optimal design cost: 262 €. . 99
6.8 Example of optimal topology with model HopR1: Optimal design cost: 216,4 €.100
6.9 Example of optimal topology with model HopR2: Optimal design cost: 212 €. 101
6.10 An homogeneous normal network G and the optimal solution with $L = 3$ and $K = 2$. 102
6.11 An homogeneous reliable network G_R and a feasible solution with $L = 3$ and $K = 2$. 102
6.12 Optimal solution for the homogeneous reliable network G_R with $L = 3$ and $K = 2$. 103
6.13 Feasible solutions subsets depending of the model. 103
6.14 Example of a network to design. 104
6.15 Illustration of feasible solutions for each model. 104
6.16 Representation of graph G (a.) and graph $G^{'}$ (b.) with one reliable edge ac. . 110
6.17 Representation of associated layered graph $G^{''}$. 111
6.18 Evolution of the gap (%) in function of the percentage of reliable edges. . . . 114
6.19 Performance profile comparing methods on the *SNDlib* test set. 116
6.20 Evolution of the average gap (%) depending on the scale economy factor f. . 117

List of Tables

1.1 World Internet usage and population statistics. 2

2.1 Details of the instances used. 20
2.2 Cost of the optimal solution. 22
2.3 CPU time in sec needed to find the optimal solution. 24
2.4 Comparison of number of variables used in the principal models and approaches. 24

4.1 H' matrix for the example with 5 nodes 48
4.2 Feasible configurations for inequality (4.13). 51
4.3 Summary of the different design models. 59
4.4 Fractional optimal solutions when $L > 3$. 65
4.5 Different cases and associated contributions of this chapter. 66

5.1 Values of $\overline{\pi}$ for the *generalized jump inequality* for $L = 4$ and $r = 2$. 79
5.2 Values of $\overline{\pi}$ for nodes in W for the *two-layered 4-path-cut inequality*. 79
5.3 Values of $\overline{\pi}$ for cut (5.14). 81
5.4 Instances description. 83
5.5 Geometric average of gap $\left(\frac{IP^* - LP^*}{IP^*} * 100\right)$ for all instances. 83
5.6 Number of instances (out of 213) unsolved within one hour. 85
5.7 Average CPU times for all approaches. 87
5.8 LP Relaxation, Integer optimal solution and heuristic performances for the entire test set. 87
5.9 Average numbers of iterations. 88
5.10 Number of cuts generated and of nodes visited for the entire test set. 88
5.11 CPU time spent for solving Benders subproblems. 89

6.1 Instances description. 113
6.2 Average CPU time, problem not solved and final gap for *SNDlib* test set. . . 114
6.3 Average gap (%) depending of the model and the number of commodities. . . 114
6.4 Average gap (%) depending on the number of edge-disjoint paths K. 115
6.5 Average gap (%) depending on the hop limit L. 115
6.6 Average gap (%) depending on the process used to generate reliable subset of edges. 116
6.7 Impact on the gap of the scale economy factor f. 117
6.8 Impact on the IP CPU Time of the scale economy factor f. 117

6.9 Economy proposed by the utilization of reliable edges. 118

A.1 CPU Time (sec.)/Gap on TC test set. 136
A.2 CPU Time (sec.)/Gap on TC test set. 137
A.3 CPU Time (sec.)/Gap on TE test set. 138
A.4 CPU Time (sec.)/Gap on TE test set. 139
A.5 CPU Time (sec.)/Gap on SNDlib test set. 140
A.6 LP Relaxation. Integer optimal solution and heuristic performances on TC test set. 141
A.7 LP Relaxation. Integer optimal solution and heuristic performances on TC test set. 142
A.8 LP Relaxation. Integer optimal solution and heuristic performances on TE test set. 143
A.9 LP Relaxation. Integer optimal solution and heuristic performances on TE test set. 144
A.10 LP Relaxation. Integer optimal solution and heuristic performances on SNDlib test set. 145
A.11 Number of cuts generated and of nodes visited for the TC test set. 146
A.12 Number of cuts generated and of nodes visited for the TC test set. 147
A.13 Number of cuts generated and of nodes visited for the TE test set. 148
A.14 Number of cuts generated and of nodes visited for the TE test set. 149
A.15 Number of cuts generated and of nodes visited for the SNDlib test set. 150
A.16 Number of iterations on TC test set. 151
A.17 Number of iterations on TC test set. 152
A.18 Number of iterations on TE test set. 153
A.19 Number of iterations on TE test set. 154
A.20 Number of iterations on SNDlib test set. 155
A.21 CPU time (sec.) spent for solving Benders subproblems on TC test set. . . . 156
A.22 CPU time (sec.) spent for solving Benders subproblems on TC test set. . . . 157
A.23 CPU time (sec.) spent for solving Benders subproblems on TE test set. . . . 158
A.24 CPU time (sec.) spent for solving Benders subproblems on TE test set. . . . 159
A.25 CPU time (sec.) spent for solving Benders subproblems on SNDlib test set. . 160

Contents

Acknowledgements v

Abstract vii

Résumé ix

List of Publications xi

List of Figures xiii

List of Tables xv

Contents xvii

1 Introduction 1
- 1.1 Telecommunications industry . . . 1
- 1.2 Network design and survivability . . . 3
  - 1.2.1 Network design problems . . . 3
  - 1.2.2 Survivability . . . 4
- 1.3 Graph theory key concepts . . . 6
- 1.4 Outline of the thesis . . . 7

2 Survivability & decomposition methods: first contacts 9
- 2.1 Introduction . . . 9
- 2.2 Research objectives . . . 9
- 2.3 Literature review about decomposition methods . . . 12
- 2.4 Formulations & models . . . 14
  - 2.4.1 Assumptions . . . 15
  - 2.4.2 Model 1: Independent approach . . . 15
  - 2.4.3 Model 2: Global approach . . . 16
  - 2.4.4 Benders approach . . . 17
  - 2.4.5 Model 3: Aggregated approach . . . 19
- 2.5 Numerical experiments . . . 20
- 2.6 Summary . . . 25

3 On the stable set of size 2 polyhedron 27

3.1 Introduction . . . 27
3.2 Literature review . . . 27
3.3 Formulation . . . 32
3.4 Facets . . . 33
3.5 Summary . . . 34

4 On the K edge (arc)-disjoint L-paths problem: Properties of a layered extended formulation 35
4.1 Introduction . . . 35
4.2 Towards an hop-indexed formulation . . . 38
4.2.1 Formulation in the space of the design variables z . . . 38
4.2.2 Alternative layered graph . . . 38
4.2.3 Formulation in the space of the flow variables x . . . 40
4.3 Arc-disjoint versus Edge-disjoint . . . 44
4.4 Properties of the Hop-indexed formulation . . . 45
4.5 Routing problem versus Design problem . . . 57
4.6 Extension to $L \geq 4$ cases . . . 63
4.7 Summary . . . 65

5 Benders decomposition for the hop-constrained survivable network design problem 67
5.1 Introduction . . . 67
5.2 Problem description . . . 69
5.3 Benders decomposition . . . 71
5.3.1 Reformulation . . . 71
5.3.2 Algorithmic approach . . . 73
5.3.3 Heuristic . . . 75
5.4 Feasibility problem . . . 76
5.5 Computational results . . . 81
5.5.1 Implementation details . . . 82
5.5.2 Instances details . . . 83
5.5.3 Results . . . 83
5.6 Summary . . . 85

6 On the hop-constrained survivable network design problem with reliable edges 91
6.1 Introduction . . . 91
6.2 Problem statement . . . 96
6.2.1 Extended formulation for the survivable network design problem with hop constraints . . . 97
6.2.2 Extended formulation for the survivable network design problem with hop constraints and reliable edges. . . . 97
6.2.3 Properties of the reliable models HopR1 and HopR2 . . . 101
6.3 Benders reformulation . . . 105
6.3.1 Master problem of model HopR1 . . . 105
6.3.2 Master problem of model HopR2 . . . 106
6.3.3 Subproblem of model HopR1 and HopR2 . . . 106
6.4 Algorithms . . . 107
6.5 Feasibility problem . . . 109

6.6 Computational experiments 112
6.6.1 Implementation details 112
6.6.2 Instances details 112
6.6.3 Complexity of the problem and quality of Hop models 113
6.6.4 Economic advantages of reliable edges 118
6.7 Summary 118

7 Conclusion **121**
7.1 Contributions 121
7.2 Limits and future work 124

Bibliography **127**

Appendices **133**

A Benders decomposition for the hop-constrainted survivable network design problem: Detailed Results **135**

Chapter 1

Introduction

THis research focuses on the survivable network design problem. In this introduction chapter we motivate our choice to study this specific problem so important for the telecommunications industry nowadays. In Section 1.1 we analyze roughly the telecommunications industry and we justify why it is so important now for operators to build survivable networks at lower cost. Because telecommunications are founded on a large variety of networks, we make, in Section 1.2, a review of the literature first about the large field of research concerning network design and then about survivability. Finally, in Section 1.3, we conclude by a small note about some basic graph theory tools. This chapter ends with a general outline of the thesis.

1.1 Telecommunications industry

Telecommunication can be defined as the transmission of messages from a transmitter to a receiver via a transmission medium over significant distances for the purpose of communication. Since the beginning, things changed considerably. Indeed, in earlier times, to communicate people used either visual signals (such as smoke, semaphore telegraphs, signal flags, optical heliographs,...), or audio messages (such as coded drumbeats, lung-blown horns, loud whistles,...). Since the 18th century, with the control of the electricity and later on of electronics, the world of telecommunications knew a first important upheaval in their development. Indeed telecommunications has typically involved the use of electric means such as the telegraph, the telephone, the teletype, the use of microwave communications, the use of fiber optics and their associated electronics, and since 50 years, the use of the Internet. Now, due to, amongst other things, the development of wireless techniques, people can send emails, read newspapers, watch television or video, listen radio, book a flight or buy tickets for the new show of their favourite artist,..., and all these things in real time and simply from a smart phone as big as a card deck.

No other industry touches as many technology-related business sectors as telecommunications, which, by definition, encompasses not only the traditional areas of local and long-distance telephone service, but also advanced technology-based services including wireless communications, the Internet, fiber-optics and satellites. Telecom is also deeply intertwined with entertainment of all types, including cable TV systems, since cable

companies are now aggressively offering local exchange service and high-speed Internet access. Telecommunications is everywhere and touches everything, it has a significant social, cultural and economic impact on modern society. Indeed, the role that telecommunications has played in social relations has become increasingly important with the high popularity of social networking sites and with technologies like SMS and the telephone. In cultural terms, telecommunications has also increased the ability to access music, video, virtual visit of museums... Last but not least, from an economical point of view, telecommunications play an important role in the world economy as the worldwide telecommunications industry's revenue was estimated to be 3.16 trillion € in 2008 (These figures include equipment and related services, as well as subscriber revenues[1].) which represents 3% of the gross world products. The service revenue of the global telecommunications industry was estimated to be 1.39 trillion € in 2008, and is expected to reach 2.21 trillion € by 2013. The International telecommunications Union (ITU) estimates the number of mobile subscribers worldwide at 4.01 billion at the end of 2008, up from 1.41 billion in 2003. This is a compound annual growth rate of 23.2%, and represents 59.3% of world population. Concerning the Internet, as illustrated on Table 1.1[2], in 2010 around 25% of the world population are Internet users, in 2000, this percentage reached only approximately 5%. Global wireless subscribers will grow to over 5.5 billion by 2011 to 2012.

World Regions	Population	Internet users		Penetration	Growth
		(2000)	(2010)	(% population)	(2000-2010)
Africa	991,002,342	4,514,400	67,371,700	6.8%	1,392.4%
Asia	3,808,070,503	114,304,000	738,257,230	19.4%	545.9%
Europe	803,850,858	105,096,093	418,029,796	52.0%	297.8%
Middle East	202,687,005	3,284,800	57,425,046	28.3%	1,648.2%
North America	340,831,831	108,096,800	252,908,000	74.2%	134.0%
Latin America	586,662,468	18,068,919	179,031,479	30.5%	890.8%
Oceania	34,700,201	7,620,480	20,970,490	60.4%	175.2%
WORLD TOTAL	6,767,805,208	360,985,492	1,733,993,741	25.6%	380.3%

Table 1.1: World Internet usage and population statistics.

For a telecommunications operator, the international business has changed almost beyond recognition. Regulatory and technological changes have transformed a once cosy cartel of monopoly carriers into a fiercely competitive market (AT&T, Telefonica, Deutshe Telecom, Vodafone, Verizon, France Telecom, ...) characterized by plummeting prices and razor-thin margins. The shock of market liberalization (particularly the European "Big Bang" reform of the late 1990s) has worked its way through the industry, and carriers have adapted to their new environment, albeit reluctantly. Large staff reductions and network rationalization have enabled carriers to survive, if not to thrive. Moreover several major additional factors are creating new changes in the telecommunications sector today, including high budgetary pressures and slower growth due to the global recession, a shift in residential and personal use from wired services to wireless, an intense competition between cable and wired services providers and rapid advances in Internet and wireless technologies, including more advanced cellular handsets and wider availability of 3G services.

This unavoidable disturbance in the telecommunications industry has forced the oper-

[1] From the website of the International Telecommunications Union ITU: http://www.itu.int
[2] From the website of the Internet World Stats: http://www.internetworldstats.com

ating companies to change the way they approach their business. Ingenuity, innovation, cost control are three major sources of competitive advantages. Client satisfaction, also, has become an essential key criteria for success but not at any price. The main concerns of telecommunications companies are now reliability, quality of service and price, even if these three objectives may be contradictory. Indeed to improve the quality of service as the reliability, service providers need, for instance, to increase the number of links as the capacities and hence its construction and maintenance cost which has a direct effect on the price of the service.

Survivability becomes then a major preoccupation leading to potential advantages. If the rupture of a single telephone line, for instance, is not a catastrophic event for an operator, the inability to transfer data, even for a limited period of time, could be unacceptable to the user. Even a small risk of a failure in the operator's network could discourage a client from making use of the service. Failures however cannot be entirely eliminated, whether it be a question of equipment (switching, router, transmission ...) at a network node, or a physical break in a connection link. However operating companies, due to the growth of competition and consequently due to the need to limit costs, cannot invest in excess capacities, allowing them to ensure a maximum level of client satisfaction. Operating companies try therefore to ensure that their networks will be survivable enough, which means that enough capacity is available, so that all traffic interrupted by a failure may be rerouted. Nevertheless this implies huge capital investment for their network operators.

Furthermore, strategic considerations like mergers, acquisitions and other industry changes redefined also the telecommunications industry. For instance, AT&T and SBC merged and MCI merged into Verizon. Sprint and Nextel have combined to create wireless giant Sprint Nextel. The competitive landscape is shifting dramatically due to these mergers. In addition, government regulations are evolving quickly, which will bring even bigger changes to business strategies. Overall, the telecommunications industry is in a state of continuous technological and economic flux driven by intense competition and new technologies.

As a consequence, a key issue in telecommunications is to find the right trade-off between capacity allocation and service level , or as Wu [104] said :"*...Reducing network protection cost, while maintaining an acceptable level of survivability, has become an important challenge for network planners and engineers...*".

1.2 Network design and survivability

1.2.1 Network design problems

Network design is present everywhere, from chips (micro integrated circuits) to trips (global transport networks) and network design is also important, as hundreds of billions of euros are devoted to annual investments in infrastructure networks. Network design problems which, among other issues, include topological optimization and optimum network dimensioning, arise in many important areas of applications, such as telecommunications (such as the design and expansion of backbone, metropolitan, local access networks, restoration planning,...) see for instance Geoffrion in [47] and Minoux

in [83], transportation systems (package delivery, less-than-truckload consolidation, hub location, train blocking, rail freight scheduling, airline route planning, fleet assignment, flight scheduling,...), supply chain designs (warehouse and plant location, distribution systems design, supplier selection,...) see Barnhart in [12], public utilities planning (water, electricity, gas, waste,...), flexible and automated manufacturing systems, energy systems... In this thesis, we have decided to focus mainly on the telecommunications sector.

But even at this level, network design is interesting and challenging since many classical optimization models exist and research opportunities are varied (theory, algorithms, applications,...) see for instance Altinkemer in [5], Balakrishnan *et al.* in [11], Ben-Ameur and Gouveia in [13], Gabrel *et al.* in [45] and Minoux in [85]. An important class of problems that have been studied is the class of capacited problems, in which the network edges have capacity constraints and in which the aim is to find optimal flows taking into account these capacity constraints. In [3], Agarwal *et al.* deal with the design of capacited multicommodity networks with multiple facilities, as Labbé *et al.* in [71], Minoux in [84], Rajan and Atamtürk in [91], Randazzo *et al.* in [92] and Wu in [104]. Depending on the types of facilities[3] available to design the network, this problem may have several versions. On the other hand, some others researches as Bienstock *et al.* in [16], Feremans *et al.* in [39], De Giovanni *et al.* in [49], Maurras and Vanier in [82] and Randazzo *et al.* in [93] focus on uncapacited problems and try to define the right capacity which leads to the minimum total cost.

Another field of research concerns the way the cost function will be modeled. The study of Maurras and Vanier in [82] uses a linear cost function which means that the transfer cost of an edge increases linearly with the quantity of data using this edge. Others use two cost components, one fixed cost for the link installation or activation and one variable cost depending on the quantity of data using this edge. Minoux in [45, 84, 85] prefers to use a discrete, non-convex, discontinuous and step increasing cost function on the links which is, according to him, appropriate to capture two major aspects of actual cost structure in telecommunications network engineering:

- ◇ the discreteness of capacity and cost increase on the links as well as at the nodes.
- ◇ the so-called "economy of scale" phenomenon, which is basically that the cost per unit of capacity installed decreases with the maximum capacity provided by an equipment.

But since few years a major domain of interest in network design concerns the survivable character of networks.

1.2.2 Survivability

As already stated, the Internet has seen tremendous growth in the past decade and has now become the critical information infrastructure for both personal and business appli-

[3]Facilities with a single fixed capacity, i.e. T1 facility which provides a transmission rate of 1.5 megabits/second. In addition to a high capacity facility such as T1, a facility of unit capacity is also available (Two-facility loading problem). Finally, several facilities of different capacities and costs are available.

cations. The Internet is based on a large-scale network infrastructure which is expected to work all the time without interruption or loss of quality. Service disruption even for a short time can be catastrophic in the world of e-commerce, causing economic damage as well as tarnishing the reputation of a network service provider or a company. In addition, many emerging services such as Voice over IP and virtual private networks require stringent service availability and reliability. Because the probability that edges (i.e. optic fiber links, copper lines...) and/or vertices (i.e. routers, switches...) break down is not null, and because a failure of those lines or of the switches can result in traffic loss or could even disconnect some parts of the network, network providers have to take into account such failure situations when they design their networks. Building networks that work in cases of edges or vertices breakdowns became a key issue for network provider companies during the last years. Moreover, it has been observed that most failures are transient: 50% last less than a minute. Hence there is a growing demand for failure resilient methods that ensure high service availability and reliability despite transient link failure. As expressed by Liu and Trivedi in [75], the survivability analysis was first discussed in the context of military command, control and communication (C^3) systems in 1970s.

A frequent practise in dimensioning survivable networks is to design two networks: a "base network" used in normal operation with no failure and a so-called "reserve network" dedicated to rerouting traffic in case of failure; see for instance the studies of Labbé *et al.* in [71], Lisser *et al.* in [74], Maurras and Vanier in [82] and Minoux in [83]. In other words, capacities on the reserve network are not used to route traffic in normal situations and capacities on the base network are not used to reroute traffic in case of failure. Research also focuses on the rerouting techniques. Two different rerouting methods may be used: local or global rerouting. With local rerouting, each failure induces a new local demand between the end points of the failed link with a value equal to the total flow initially routed on this failed link. However with global rerouting (see the study of Lisser *et al.* in [74]), each demand affected by the failure is rerouted from its proper origin to its destination (see also the work by Lisser *et al.* in [73]). Others issues in survivability are also interesting. Altinkemer in [5] cites some studies linked with the survivability concept like the one of Rosenberg in [96] who investigates capacity requirements for node and arc survivability in a rectilinear grid communication network when a failure occurs. The study of Soni and Pirkul in [99], tries to re-establish connectivity during a link or node failure by adding new constraints where each case is simple enough to solve.[4] In [6], Aneja *et al.* tackle the problem using another approach. They say that usually it is not easy to reorient flows quickly enough to take advantage of alternate routes. They define a so-called "residual flow" as the flow which does not use any of the destroyed nodes and/or arcs, in other words, this residual flow is the global flow not affected by the failure. The objective of their research is therefore to determine a flow such that the residual flow value is maximized.

As the scope of applications is large, the scope of methods used for the resolution of this kind of problems is also diversified [13]. Some studies choose branch-and-cut techniques like the studies of Bienstock *et al.* in [16], De Giovanni *et al.* in [49] and Minoux in [83], column generation like Shaio in [97], column-and-cut-generation like Rajan and

[4] As we can see later, this concept is quite similar to our decomposition principle.

Atamtürk in [91], lagrangian and linear relaxation like Wynants in [105], tabu search like Girard *et al.* in [50] and Wynants in [105], simulated annealing like Holmberg and Yuan in [60] and, as our work, the Benders decomposition like Benchakroun *et al.* in [14], Geoffrion and Graves in [47], Magnanti *et al.* in [80], Maurras and Vanier in[82] and Randazzo *et al.* in [93].

1.3 Graph theory key concepts

In this section, we present some basic definitions and notations of graph theory, which will be frequently used in the subsequent chapters.

As stated previously, in this thesis we focus on the survivable network design problem for telecommunications applications. More generally in network design problems, networks can be represented by an **undirected graph** $G = (V, E)$, where the set V represents the **vertex set** of cardinality n, and where the set E is the **set of potential edges** of cardinality m. If two vertices i and $j \in V$ we define by ij a fixed edge between i and j. These nodes i and j will be called the **end nodes** of ij. The edge ij is then **incident** to i and j, while these two nodes are **adjacent** to each other. A loop in a graph G is an edge whose two end nodes coincide. Two edges are **parallel** if they have the same end nodes. A graph $G = (V, E)$ is **simple** if E does not contain parallel edges or loops. Moreover, the graph G is **complete** if there exists an edge ij between each pair of nodes $i, j \in V$.

We also consider **directed graphs** (or digraphs) $D = (V, A)$, where V is the node set and A is the arc set. An **arc** between two nodes i and $j \in V$ is a couple (i, j) where i is the **tail** or the origin of the arc and j its **head** or destination. The set A is obtained from each edge ij of E by creating two directed arcs (i, j) and (j, i). These two arcs (i, j) and (j, i) are called **antiparallel**. We call the graph $G = (V, E)$ the **underlying graph** of the digraph $D = (V, A)$ if there exists a bijection between the arcs of A and the edges of E, such that for each arc $(i, j) \in A$ there exists an edge $ij \in E$ and for each edge $ij \in E$ the arc (i, j) or the arc (j, i) is in A. The **associated digraph** $D(G)$ of a graph G is the digraph obtained from G by replacing each edge by two antiparallel arcs with the same ends. In the following we call both graph G or $D(G)$ by G and we change only the set E or A.

A **path** P of G is an alternate sequence of nodes and edges $(i_1, e_1, i_2, e_2, ..., i_{q-1}, e_{q-1}, i_q)$ where $e_k = (i_k, i_{k+1})$ for $k = 1, ...q - 1$. We denote a path P by either its sequence $(i_1, ..., i_q)$ or its edge sequence $(e_1, ..., e_{q-1})$. The nodes i_1 and i_q are called the end nodes of P, while its other nodes are said **internal**. A path is **simple** if it does not contain the same node twice. By opposition, a non-simple path is called a **walk**. A path whose end nodes are o and d is called an **od-path**. A **cycle** in G is a path whose end nodes coincide, that is, $i_1 = i_q$. A cycle is simple if, with the exception of i_1, it does not contain the same node twice. A **chord** is an edge between any two non-adjacent nodes of a path (cycle). Two *od*-path are **edge-disjoint** (respectively **arc-disjoint**) if they have no edges (respectively arcs) in common. They are **node-disjoint** if they have no internal nodes in common. A graph $G = (V, E)$ is said to be **connected** if there is a path between any two vertices.

We consider also a set Q of different **commodities** that must be routed through the network. A commodity $q \in Q$ can be represented by a pair of node $o(q)$, the origin node and $d(q)$ the destination node which have to be connected. Let $L \geq 2$ be a fixed integer, for a demand $\{o(q), d(q)\} \in Q$, an **L-od-path** in G is a path between $o(q)$ and $d(q)$ of length at most L, where the length of a path is defined as the number of its edges (also called hops). We speak of **L-hop-constrained paths**.

Let $G = (V, E)$ be an undirected graph. If $W \subset V$ is a node subset of G, then the set of edges that have only one node in W is called a **cut** and denoted by $\delta(W)$. The number of edges incident to some node i is equal to $|\delta(i)|$ and is called **the degree** of the node i. Given two nodes o and d, a cut $\delta(W)$ such that $o \in W$ and $d \in \overline{W} = V \setminus W$ is called an **od-cut**. A **partition** $\Pi = (V_0, V_1, ..., V_p)$ of V is a collection of disjoint node subsets whose union is V.

A graph $\tilde{G} = (\tilde{V}, \tilde{E})$ is an **induced subgraph** of $G = (V, E)$ if $\tilde{V} \subseteq V$ and $\tilde{E} \subseteq E$. Given a graph $G = (V, E)$, a **matching** M in G, or set of **independent edges**, is a set of pairwise non-adjacent edges, that is, no two edges share a common vertex. A **stable set** S in G, or set of **independent nodes**, is a set of pairwise non-connected nodes. That is, it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two nodes. A **bipartite graph** (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V, that is, U and V are independent sets.

The traditional **network design problem** consists in looking for, among all the potential edges, the ones on which install capacity in order to allow connectivity for all the different commodities $q \in Q$. The result of this optimization problem is a subgraph $G' = (V', E')$, where $E' \subseteq E$ represents the subset of selected edges and where $V' \subseteq V$ is the subset of vertices which are extremities of at least one selected edge.

1.4 Outline of the thesis

In Chapter 2 we study the uncapacitated case of the survivable network design problem. We consider the failure of one arc at the same time and we build a specific failure scenario for each arc of the network. The objective is to define the optimal capacity which allows a feasible routing for all commodity even if a single arc is broken. We build a large-scale arc-flow formulation and we develop a Benders decomposition approach to solve these huge problems. We show how fast the Benders decomposition approach is for large networks compared with a basic implementation of the large-scale model with solver CPLEX.

In Chapter 3 we study a satellite problem which is the stable set of size 2 problem. The theoretical results of this chapter will be useful to prove a nice property in Chapter 4. Given a graph $G = (V, E)$ and a weight w_i associated to each vertice $i \in V$, the stable set of size 2 problem consists in finding a set of 2 non-connected vertices of maximum weight. In this chapter we give a formulation and we prove that it gives a complete description of the associated polyhedron. We define conditions for some classes of in-

equalities to be facet defining.

In Chapter 4 we study a version of the capacitated survivable network design problem. Given a graph $G = (V, E)$ the K edge (arc)-disjoint L-paths problem consists in finding between only one pair of terminals, one origin node o and one destination node d exactly K edge-disjoint paths composed by a maximum of L hops. K paths are, by definition, edge-disjoint if they do not share any common edge. For some new services it is not sufficient for customers to guarantee that the K paths are edge-disjoints. Indeed, new applications like video, audio or IP telephony... require small transmissions delays (i.e. short paths must be used for each commodity) because the transmission time is one of the most important key criteria for customer satisfaction. Consequently an upper limit on the length of the paths is fixed to ensure that short paths will be used to route the different commodities on the network. In this chapter we propose an extended formulation to model this problem and we demonstrate some nice properties for some values of the parameters K and L.

In Chapter 5 we extend the problem developed in Chapter 4 to the multicommodity case. We propose a formulation which is valid for any K and any L and we propose some cutting plane algorithms based on the Benders decomposition. We develop also four branch-and-cut algorithms and we discuss about the best place in a branch-and-bound tree to add the Benders cuts in order to accelerate the solving process. From a theoretical point of view we prove that the Benders cuts generalise the constraints used by Huygens *et al.* in [30, 61, 62, 63] to formulate the problem for $L = 2, 3, 4$, as well as new inequalities when $L \geq 5$.

Finally, in Chapter 6 we study a variation of the previous problem in which a subset of the edge set is assumed to be reliable. By reliable we mean that the edges belonging to this subset of reliable edges are not prone to failure, are very reliable edges. We propose again an extended formulation for this new problem as a branch-and-cut algorithm based on the Benders decomposition approach. We show how fast this algorithm is compared to a basic but huge formulation simply used with CPLEX solver. We conclude the chapter by an economical analysis concerning the real interest of this kind of reliable edges for network providers.

Chapter 2

Survivability & decomposition methods: first contacts

2.1 Introduction

In this chapter, as in the remainder of this thesis, the main objective is to design survivable networks which can work, even in case of potential failures. On the one hand the network designed has to be survivable but on the other hand the installation cost has to be as low as possible for obvious management reasons. Most of the time, this kind of problems can be solved using linear models which are often intuitive and easy to build but which are composed also of a huge number of variables and constraints. These large-scale problems are, of course, very complicated to solve. This chapter is a first contact with a usual technique in operational research often used to solve large-scale problems. This technique is called the decomposition method and more especially here the Benders decomposition. The key idea is to split a huge problem into many smaller ones normally easier to solve.

This chapter is organized as follows: In Section 2.2 we detail the objectives of this research, how, in this chapter, we consider the concept of survivability. Section 2.3 is used to explain the main objectives and principles of the decomposition techniques and how these techniques are frequently used in the literature to solve different varieties of problems, for telecommunications applications or not. In Section 2.4 we formulate the problem under study using linear models and we develop our different approaches to design survivable network. Finally in Section 2.5 a numerical study is realized which shows how fast the model is using the Benders decomposition compared to a basic implementation of the traditional global model with solver CPLEX.

2.2 Research objectives

As Wynants in [105], this chapter focuses on a particular kind of network flow problems: the network synthesis (*NS* for short) or dimensioning problem which consists in determining the amount of capacity to install on each link of a given network in order to satisfy some known traffic requirements at minimum cost. In this chapter as in the remainder

of this thesis, we study more precisely the subclass of the survivable problems (*SNDP*) which is a particular class of the *NS* problems. As for the *NS*, a classical *SNDP* instance is characterized by a graph $G = (V, E)$, a cost structure and a set of requirements to send on the network. Adding capacity to links induces some costs which depend on the nature of the application. In this study, we decide to approximate these installation costs by a linear function dependent on the amount of capacity installed. As mentioned in the introduction of this thesis, by survivability we want to design networks which work even in case of failure and more especially here in case of link (edge) failure and not in case of node failure. For simplicity reasons, we consider in this chapter only cases of single link failure, it means that at most one link is considered as damaged.

We formally represent the network as a graph $G = (V, E)$, where V is the set of vertices or nodes and E is the set of non-directed potential edges linking the different elements of the node set[1]. Because each edge between node i and j can be used in both directions (from i to j and also from j to i), we define a second set A, which represent the set of directed potential arcs with the following property: $|A| = 2|E|$. By definition we decide to use the following notation when we talk about an edge $ij \in E$ and about the two linked arcs (i, j) and $(j, i) \in A$. The main idea of this chapter is to find the minimum network capacity which will be sufficient in each case (failure and non-failure) to send information for each pairs of "origin-destination". We illustrate this idea in the following simple example.

Figure 2.1: Example of a multicommodity network with 2 demands

Consider the simple network represented in Figure 2.1. This network is composed of 8 nodes (A,B,C,D,E,F,G,H), 10 potential links and 2 different demands (10 units from H to A and 5 units from B to G)[2]. Assume also that the optimal topology of this network leads to the situation represented in Figure 2.2. We can see in this figure that among the 10 potential edges, only 6 are used in the optimal topology without failures, leading to two different paths ($(B - C - E - G)$ for the demand of 5 units from B to G, and $(H - F - D - A)$ for the demand of 10 units from H to A)[3]. If we consider an edge cost of 1 € by unit of capacity, this solution leads to a total installation cost of 45 €[4] which minimizes the total cost without failure occurrence.

Now, assume that only the two central edges (CE) and (DF) can break down one at a time. Using the topology of Figure 2.2 in case of failure, leads certainly to the optimal

[1] In our application to the sector of telecommunications, and more precisely to the Internet world, the different nodes are the routers and the computers whereas the edges represent the link between these routers.

[2] In this simple example, we ignore the cost structure associated with the links.

[3] Note that, in the picture, the capacity of each edge is indicated beside each edge used.

[4] 45 is just equal to the sum on the edge of the capacity used.

solution in terms of cost but on the other hand this topology can not be totally effective, because it is impossible to route all demands without delay and good service quality.

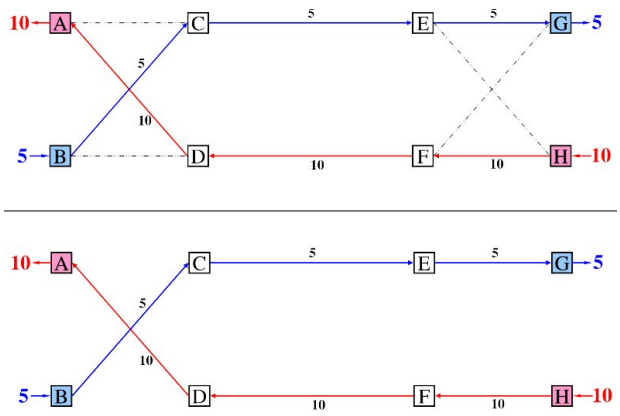


Figure 2.2: Solution without failure

As an alternative approach, an intuitive solution would be to fix, for each potential edge of the network, a capacity equal to the sum of the different demands to send on the network. In this case, this intuitive solution leads to a total cost of 150 € because we have 10 potential edges and the sum of the demands is equal to 15 units. This solution is very efficient in terms of service level because this topology works in any failure situation. But, of course in terms of cost, it is very expensive.

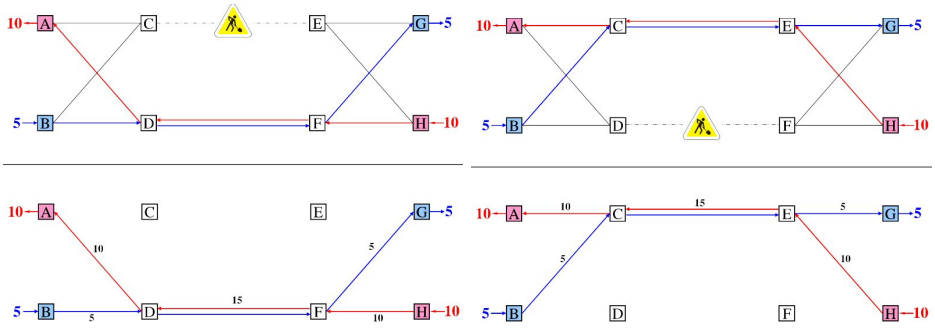


Figure 2.3: Solution with one edge failure at a time

Another approach, (which is the one we shall adopt here) is to compute for each failure case a new optimal network topology.[5] We see clearly in Figure 2.3 that the two configurations are quite different. Because the network configuration has to work for the two different cases of failure, we have to use a topology like the one represented in Figure 2.4, which is an aggregate solution. From an economical point of view, this solution leads to

[5]In the Figure 2.3, the left part represents the failure of edge (CE) whereas the right part represents the failure of edge (DF).

a total cost of 90 €. This cost is greater than the one illustrated in the Figure 2.2 but it is the minimum cost which leads to a 100% service level.

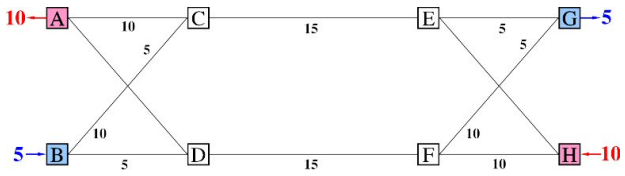


Figure 2.4: Optimal solution with failures

In this simple example it is relatively easy to imagine the aim of this chapter. The problem is that, in practice, we do not use small networks with only 8 nodes and 10 edges. For instance, and as expressed in [74], a realistic France Telecom's main interconnection network counts 60 nodes and 120 links. With this kind of network, it is more complicated to find the optimal solution for all scenarios of failure because the number of variables and constraints of the model explode, and the problems are often large-scale. That is the reason why we introduce in this work the concept of decomposition.

2.3 Literature review about decomposition methods

Since the original work in 1960 of Dantzig and Wolfe, the fathers of the well known simplex algorithm, the idea of decomposition, by which a large problem is decomposed into smaller problems, has existed as an attractive approach to large-scale linear programming. These two authors proposed the first version of the decomposition techniques known in the literature. Without going into detail, (the reader can find more explanations of the theory in [77]), suppose that the basic model is a linear programming problem with two sets of constraints as follows:

$$\begin{array}{lrcl} \max & cx & & \\ s.t. & A_1 x & \leq & b_1 \\ & A_2 x & \leq & b_2 \\ & x & \geq & 0 \end{array} \tag{2.1}$$

where $c \in \mathbb{R}^n, A_1 \in \mathbb{R}^{m \times n}, b_1 \in \mathbb{R}^m, A_2 \in \mathbb{R}^{q \times n}, b_2 \in \mathbb{R}^q$ are given constants and $x \in \mathbb{R}^n$ is a vector of variables. This kind of linear program with the block-angular structure arises from the optimization of systems consisting of coupled subsystems. Applied to these block-angular structures, the elegant decomposition principle introduced by Dantzig and Wolfe, implies that the entire problem can be solved by using a coordinated sequence of independent subproblems corresponding to the subsystems. In others words, the fundamental idea is to solve (2.1) by interaction between two optimization problems, one of which is subject to the first set of constraints and the other subject to the second set of constraints.

Two years later, in 1962, J.F. Benders devised a clever approach for exploiting the

structure of mathematical programming problems with so-called complicating variables (variables which, when temporarily fixed, render the remaining optimization problem considerably more tractable). For the class of problems specifically considered by Benders, fixing the values of the complicating variables reduces the given problem to an ordinary linear program, parameterized, of course, by the value of the complicating variables vector. The algorithm he proposed for finding the optimal value of this vector employs a cutting-plane approach. Linear programming duality theory such as Farkas lemma was also employed to derive the natural family of cuts characterizing the representations. The Benders decomposition thus leads consequently to certain improvement in terms of algorithm convergence and also in terms of rapidity, which justifies the fact that it has been used in a significant number of applications [7, 12, 14, 47, 67, 80, 82, 93, 44].

More formally, the original problem can be written as follows:

$$\begin{array}{lrcl} \max & c_1x_1 + c_2x_2 & & \\ s.t. & A_1x_1 + A_2x_2 & \leq & b \\ & x_1, x_2 & \geq & 0 \end{array} \tag{2.2}$$

The Benders decomposition idea is to rewrite this original problem parameterizing one group of variables, for example the x_2 variables. This leads to the following relaxed problem:

$$\begin{array}{lrcl} \max & c_1x_1 + w_p(x_1) & & \\ s.t. & x_1 & \geq & 0 \end{array} \tag{2.3}$$

Let x_1^* be the optimal solution of (2.3) and $w_p(x_1^*)$ be the optimal solution of the following subproblem in which :

$$\begin{array}{lrcl} w_p(x_1^*) = \max & c_2x_2 & & \\ s.t. & A_2x_2 & = b - A_1x_1^* & \\ & x_2 & \geq 0 & \end{array} \tag{2.4}$$

The dual version of the subproblem can be formulated as:

$$\begin{array}{lrcl} w_d(x_1^*) = \min & y_2^T(b - A_1x_1^*) & & \\ s.t. & y_2^TA_2 & \geq c_2 & \\ & y_2 & \geq 0 & \end{array} \tag{2.5}$$

The idea is thus to solve the dual subproblem and owing to duality properties conclusions which say that the optimal dual solution is always an upper bound of the primal one, we can write the following new constraint:

$$w_d(x_1^*) = y_2^T(b - A_1x_1^*) \geq w_p(x_1^*) \tag{2.6}$$

for any $y_2 \geq 0$ such that $y_2^T A_2 \geq c_2$.

Consequently, the optimal solution of the dual subproblem leads to the addition at each iteration, of one or more new constraints (also called Benders Cuts) to the relaxed problem (2.3), also called "Master Program":

$$\begin{array}{rrcl} \max & c_1 x_1 + w_p(x_1) & & \\ s.t. & y_2^{T*} A_1 x_1 + w_p(x_1) & \leq & y_2^{T*} b \\ & x_1 & \geq & 0 \end{array} \tag{2.7}$$

The algorithm can be seen as a two stage optimization model in which we fix firstly the value of one variables group and then solve the Master Program. The optimal solution of this MP become the parameters of the subproblems. We solve then the subproblems and find the optimal solution which becomes, in turn, the new parameters of the MP... The Dantzig-Wolfe decomposition algorithm is thus called "**primal decomposition algorithm**", and is characterized by a division of the problem between two groups of constraints, whereas the Benders decomposition algorithm takes the name of "**dual decomposition algorithm**", and is based, on the opposite, on a variables division.

2.4 Formulations & models

In this section we propose models which are largely based on the well-known multi-commodity network flow formulations (*MCNF*). As said before, our contribution is to introduce the occurrence of failure. We can deal with this latter problem in two different ways. Firstly, and as used in the **Model 1**, we choose, what we call, an independent approach to manage the possibility of failures. In others words, each edge can break down separately, and for each edge we solve independently a different problem in which one particular edge is deleted from the original graph G. This approach is consequently similar to solve m different problems, (where m here is equal to the number of edges), and these problems have different independent solutions. Then to determine the global optimal solution we decide to define for each edge of the network a capacity equal to the worst case seen in the different independent problems. In other words, for each edge ij we look what are the m different capacity solutions given by the m independent models and we decide to take the maximum value.

Although we can hope good results in terms of CPU time because the size of each independent model stays reasonable, we are aware that, of course, this approach is not the best in terms of cost. But it is an easy and intuitive way to obtain an upper bound on the real problem. Nevertheless, the best way to deal with failure cases is to apply now a global optimization approach. Indeed, we illustrate in the results section, that the independent approach leads often to a network topology too expensive, because this method does not link the different failure scenarios together. Note that the global optimization is formulated through the **Model 2**.

This model integrates the m different scenarios by creating lots of variables and constraints leading to large-scale problems. As said in Section 2.3, one contribution of this

chapter is to use the Benders decomposition method. We thus apply this method to the Model 2 and find a new formulation (called **Benders approach**) in which the number of variables and constraints decreases considerably compared to the Model 2.

Finally, we can both improve the previous Models 2 and the associated Benders formulation by reducing again the size of the models. Indeed we could decrease slightly the number of variables used if we group together all the commodities with a common destination. This remark is integrated in a last formulation called **Model 3** and a Benders formulation is again derived (**Benders aggregated approach**).

2.4.1 Assumptions

For the following models, we assume that the cost structure is linear. Indeed, we decide to not consider fixed costs on the edges in order to use a linear program and not a mixed integer one. When a given edge ij is considered as broken, we assume that the two associated directed arcs (i, j) and (j, i) are consequently broken also. In our simulations we use complete and sparse graphs, nevertheless for the second class of graphs it is mandatory that there exist at least two different paths between two given nodes. This condition is crucial if we want to reroute correctly some flows when a particular edge is broken, without loss of service level. All the following models use a node-arc formulation. Last but not least, we consider only one edge failure at a time. A graph $G = (V, E)$ is used to represent the network where V is the set of network nodes, E is the set of potential undirected network edges, $A = \{(i, j), (j, i)|\ ij \in E\}$ is the set of directed network arcs and $Q \subset N \times N \times \mathbb{R}$ is the set of network demands or commodities. A demand $q \in Q$ is determined by a triple $o(q)$ the origin node of the demand q, $d(q)$ the destination node of the given demand q and $f(q)$ the size of the demand or the quantity of information which has to be routed from $o(q)$ to $d(q)$. For each candidate edge $ij \in E$ we also use the parameter c_{ij} which is the cost associated with edge ij. This cost is the cost we have to pay for 1 unit of capacity to install on the edge ij. In the remainder of this chapter we develop and implement the three models and the two Benders decomposition approaches.

2.4.2 Model 1: Independent approach

The independent approach uses almost the same formulation as the *MCNF* (e.g. [4]). A new set $S = E$ is defined and represents the different failure scenarios. Because to each edge is associated a failure scenario it is clear that the two sets are equivalent. The decision variables are firstly the x_{ij}^{q-mn},$(i, j) \in A$, $q \in Q$ and $mn \in S$ which represent the flow using arc (i, j) for the demand q of size $f(q)$ from the source node $o(q)$ to the destination node $d(q)$ in the case of the failure of edge mn. The second decision variable are the y_{ij}^{mn} which represent the capacity of edge ij in case of edge mn failure. Finally y_{ij} is the final capacity of edge ij taking into account all the different failure scenarios. The model can be written as:

Objective Function

$$\min \sum_{mn \in S} \sum_{ij \in E} c_{ij} y_{ij}^{mn} \tag{2.8}$$

Capacity Definition Constraints

$$\sum_{q\in Q} x_{ij}^{q-mn} + \sum_{q\in Q} x_{ji}^{q-mn} \leq y_{ij}^{mn}, ij \in E, mn \in S \tag{2.9}$$

Flow Conservation Constraints

$$\sum_{j\in V} x_{ji}^{q-mn} + \sum_{j\in V} x_{ij}^{q-mn} = \begin{cases} f(q) & \text{if} \quad i = d(q) \\ -f(q) & \text{if} \quad i = o(q) \\ 0 & \text{otherwise} \end{cases} , i \in V, q \in Q, mn \in S \tag{2.10}$$

Failure Definition Constraints

$$y_{mn}^{mn} = 0, mn \in S \tag{2.11}$$

Final Capacity Constraints

$$y_{ij} \geq y_{ij}^{mn}, ij \in E, mn \in S \tag{2.12}$$

Non-negativity Constraints

$$x_{ij}^{q-mn} \geq 0, (i,j) \in A, q \in Q, mn \in S \tag{2.13}$$

$$y_{ij}^{mn} \geq 0, ij \in E, mn \in S \tag{2.14}$$

$$y_{ij} \geq 0, ij \in E \tag{2.15}$$

where the objective function (2.8) minimizes the total cost of reserving capacity on edge in each scenario. The Capacity Definition Constraints (2.9) and the Flow Conservation Constraints (2.10) are constraints commonly used in *MCNF*. Constraints (2.11) introduce the failure in the network. Constraints (2.12) define the final capacity of the edges which is equal to the maximum capacity along the different scenarios of failure. Finally, constraints (2.13), (2.14), (2.15) are the non-negativity constraints. This approach is really independent because in the objective function, we optimize the sum on the scenario set of the independent capacities. The optimal solution is given by the value of $y_{ij} = \text{argmax}[y_{ij}^{mn},\ mn \in S]$. From the particular structure of the objective function, this model can be decomposed by failure scenario in order to solve m independent small problems where m is equal to the number of failure scenarios, in others words $m = |S|$.

2.4.3 Model 2: Global approach

Another alternative is to model the problem as a global optimization problem. This new model is almost similar to the previous one:

Objective Function

$$\min \sum_{ij\in E} c_{ij} y_{ij} \tag{2.16}$$

Capacity Definition Constraints

$$\sum_{q\in Q} x_{ij}^{q-mn} + \sum_{q\in Q} x_{ji}^{q-mn} \leq y_{ij}, ij \in E, mn \in S \tag{2.17}$$

Flow Conservation Constraints

$$\sum_{j\in V} x_{ji}^{q-mn} + \sum_{j\in V} x_{ij}^{q-mn} = \begin{cases} f(q) & \text{if} \quad i = d(q) \\ -f(q) & \text{if} \quad i = o(q) \\ 0 & \text{otherwise} \end{cases} , i \in V, q \in Q, mn \in S \tag{2.18}$$

Failure Definition Constraints

$$x_{mn}^{q-mn} = x_{nm}^{q-mn} = 0, q \in Q, mn \in S \tag{2.19}$$

Non-negativity Constraints

$$x_{ij}^{q-mn} \geq 0, (i,j) \in A, q \in Q, mn \in S \tag{2.20}$$

$$y_{ij} \geq 0, ij \in E \tag{2.21}$$

One main difference between Model 1 and Model 2 is the formulation of the objective function. In (2.16) the summation on the set of scenarios has disappeared. Now we determine directly in (2.17) the global capacity because the variable y_{ij} is computed taking into account the link between all the different scenarios of failure. In other words, constraints (2.17) link all together the different failure scenarios and ensure the fact that the approach is global. And consequently the constraints called "*Final Capacity Constraints*" (2.12) as the variable y_{ij}^{mn} used in Model 1 are no more necessary in this global model.

2.4.4 Benders approach

Model 2 is a large size model due to the important number of variables (mainly the variables x_{ij}^{q-mn}) and also due to the important number of constraints. Moreover, Model 2 is composed of two groups of distinct variables, the capacity variables y_{ij} linking all the failure scenarios and the flow variables x_{ij}^{q-mn}. These two conditions (large size model and two groups of variables) motivate our choice in favour of using the Benders decomposition approach for solving this particular Model 2. That is the reason why we have to split the global problem into a Master Problem first and some SubProblems secondly. The Master Problem which is a relaxed problem can be modeled as follows:

Master Problem:

Objective Function

$$\min \sum_{ij\in E} c_{ij} y_{ij} \tag{2.22}$$

Non-negativity Constraints

$$y_{ij} \geq 0, ij \in E \tag{2.23}$$

This Master Problem (here called MP) is function only of the y_{ij} variables, the remaining of the Model 2 (which is only function of the variables x_{ij}^{q-mn}) represents the SubProblems for a given scenario of failure $\overline{mn}$:

SubProblems for $\overline{mn}$ fixed:

Objective Function

$$\min \sum_{ij \in E} \sum_{q \in Q} 0 \cdot x_{ij}^{q-\overline{mn}} \tag{2.24}$$

Capacity Definition Constraints

$$\sum_{q \in Q} x_{ij}^{q-\overline{mn}} + \sum_{q \in Q} x_{ji}^{q-\overline{mn}} \leq y_{ij}, ij \in E \tag{2.25}$$

Flow Conservation Constraints

$$\sum_{j \in V} x_{ji}^{q-\overline{mn}} + \sum_{j \in V} x_{ij}^{q-\overline{mn}} = \begin{cases} f(q) & \text{if} \quad i = d(q) \\ -f(q) & \text{if} \quad i = o(q) \\ 0 & \text{otherwise} \end{cases} , i \in V, q \in Q \tag{2.26}$$

Failure Definition Constraints

$$x_{mn}^{q-\overline{mn}} = x_{nm}^{q-\overline{mn}} = 0, q \in Q \tag{2.27}$$

Non-negativity Constraints

$$x_{ij}^{q-\overline{mn}} \geq 0, (i,j) \in A, q \in Q \tag{2.28}$$

We compute now the dual version of the SubProblems, in which the variables y_{ij} of the Master Problem become parameters $\overline{y}_{ij}$. For the dual formulation, we need to introduce two new decision variables linked to the two main kinds of constraints (the Capacity Definition Constraints and the Flow Conservation Constraints) in the primal version of the SubProblems. Let $\sigma_{ij}^{\overline{mn}}$ be the marginal profit[6] given by one more unit of capacity available on edge ij in case of the failure of edge $\overline{mn}$, $ij \in E$. Let $\pi_i^{q-\overline{mn}}$ be the value of the shortest path from node $o(q)$ to node i, for demand q when edge $\overline{mn}$ is down, $i \in V$ and $q \in Q$. The new SubProblems are:

SubProblems for $\overline{mn}$ fixed: Dual Version

Objective Function

$$\max \sum_{q \in Q} ((\pi_{d(q)}^{q-\overline{mn}} - \pi_{o(q)}^{q-\overline{mn}}) f(q)) - \sum_{ij \in E} \overline{y}_{ij} \sigma_{ij}^{\overline{mn}} \tag{2.29}$$

Shortest Path 1 Constraints

$$\pi_j^{q-\overline{mn}} - \pi_i^{q-\overline{mn}} \leq \sigma_{ij}^{\overline{mn}}, ij \in E, q \in Q, \ \ (i \neq \overline{m}) \text{ or } (j \neq \overline{n}) \tag{2.30}$$

Shortest Path 2 Constraints

$$\pi_i^{q-\overline{mn}} - \pi_j^{q-\overline{mn}} \leq \sigma_{ij}^{\overline{mn}}, ij \in E, q \in Q, \ \ (i \neq \overline{m}) \text{ or } (j \neq \overline{n}) \tag{2.31}$$

Normalization Sigma Constraints

$$\sigma_{ij}^{\overline{mn}} \leq 1, ij \in E, \ \ (i \neq \overline{m}) \text{ or } (j \neq \overline{n}) \tag{2.32}$$

[6]Note that this new group of variables is directly linked with the constraints of Capacity Definition of the primal SubProblems whereas the other new variables $\pi_i^{q-\overline{mn}}$ are directly linked with the constraints of Flow Conservation of the primal SubProblems

Normalization Pi Constraints

$$\pi_{o(q)}^{q-\overline{mn}} = 0, q \in Q \tag{2.33}$$

Correction Constraints

$$\sigma_{ij}^{\overline{mn}} = 0, ij \in E, \ \ (i \neq \overline{m}) \text{ and } (j \neq \overline{n}) \tag{2.34}$$

Non Negativity Constraints

$$\sigma_{ij}^{\overline{mn}} \geq 0, ij \in E \tag{2.35}$$

where inequalities (2.31) are the same kind of inequalities than (2.30), but in place of going from i to j, we follow the edge ij in the opposite way. The normalization constraints (2.32), (2.33) and (2.34) exist only to prevent the algorithm from going to infinity. As seen in Section 2.3, due to duality properties, if optimality is not yet reached, we introduce new Benders cuts in the Master Problem which becomes:

Master Problem: Modified

Objective Function

$$\min \sum_{ij \in E} c_{ij} y_{ij} \tag{2.36}$$

Benders Cuts

$$\sum_{q \in Q} ((\overline{\pi}_{d(q)}^{q-\overline{mn}} - \overline{\pi}_{o(q)}^{q-\overline{mn}}) f(q)) - \sum_{ij \in E} y_{ij} \overline{\sigma}_{ij}^{\overline{mn}} \leq 0, mn \in S \tag{2.37}$$

Non-negativity Constraints

$$y_{ij} \geq 0, ij \in E \tag{2.38}$$

2.4.5 Model 3: Aggregated approach

Finally we adapt the Model 2 taking into account aggregation by destination. Some new sets or parameters have been created. Let $DEST$ be the subset of all different demand destinations. Let $ORIG[i]$ where $i \in DEST$ be the subset of the different demand origins for a special destination. Let f^d be the aggregate quantity we have to send to the destination d and let f_i^d where $d \in DEST$, $i \in ORIG[d]$ be the particular demand quantity we have to send from a specific origin and a specific destination with: $\sum_{i \in ORIG[d]} f_i^d = f^d, d \in DEST$. Then Model 2 becomes Model 3 described as:

Objective Function

$$\min \sum_{ij \in E} c_{ij} y_{ij} \tag{2.39}$$

Capacity Definition Constraints

$$\sum_{d \in DEST} x_{ij}^{d-mn} + \sum_{d \in DEST} x_{ji}^{d-mn} \leq y_{ij}, ij \in E, mn \in S \tag{2.40}$$

Flow Conservation Constraints

$$\sum_{j\in V} x_{ji}^{d-mn} + \sum_{j\in V} x_{ij}^{d-mn} = \begin{cases} f^d & \text{if} \quad i = d \\ -f_i^d & \text{if} \quad i \in ORIG[d] \\ 0 & \text{otherwise} \end{cases}, i \in V, d \in DEST, mn \in S \quad (2.41)$$

Failure Definition Constraints

$$x_{mn}^{d-mn} = x_{nm}^{d-mn} = 0, d \in DEST, mn \in S \quad (2.42)$$

Non-negativity Constraints

$$x_{ij}^{d-mn} \geq 0, (i,j) \in A, d \in DEST, mn \in S \quad (2.43)$$

$$y_{ij} \geq 0, ij \in E \quad (2.44)$$

It is obvious that if all the demands have different destinations, this model leads to the same performance that the previous Model 2, because it is not possible to aggregate anything. Otherwise, as with Model 2, we can also apply here the Benders decomposition method to this new Model 3 which leads to the **Benders aggregated approach** tested in the next section.

2.5 Numerical experiments

The different models presented before have all been implemented in AMPL language and solve with the version 8.0 of the CPLEX solver on a DELL Latitude C640 Pentium IV laptop with a 1.8 Ghz processor and 256 MB of RAM (the code can be found in [19]). We used 8 different instances of networks which are all euclidean graphs and described in the following Table 2.1.

Instances	$\|V\|$	$\|A\|$	$\|Q\|$	Complete\Sparse
3/6/2	3	6	2	C
6/26/5	6	26	5	S
7/18/9	7	18	9	S
10/32/16	10	32	16	S
30/150/2	30	150	2	S
30/150/5	30	150	5	S
30/150/7	30	150	7	S
30/150/10	30	150	10	S

Table 2.1: Details of the instances used.

In this section, we want to evaluate the performance of the different models developed during this study. As illustrated on Figure 2.8 we want to evaluate three different things. Our first test evaluates how bad the solution proposed by the independent approach is compared with the solution of the global optimization. On the opposite we could test how difficult is the global optimization compared with the independent approach by looking at the CPU time needed to obtain the optimal solution in both cases. As an illustration, Figure 2.5 shows the problem of the instance 6/26/5. It is a quasi real-based network composed of 6 nodes (6 cities in Belgium (OOS for Ostende, GEN for Gand, ATH for

Ath, KOR for Courtrai, STR for Rhode-Saint-Genèse and BKC for Berchem)). The 5 commodities are drawn with 5 different colours, for instance, the 15 units of demand from KOR to GEN with the orange arrows.

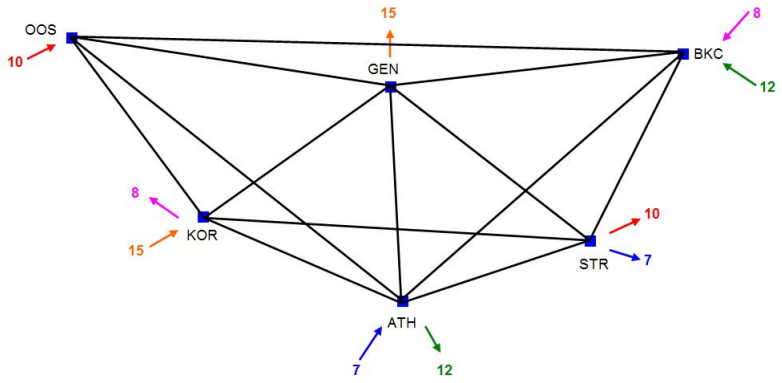


Figure 2.5: Example of network to design.

Figure 2.6 shows the optimal solution of the independent approach. The associated cost can be easily computed by summing the capacity allocated to each edge times the euclidean distance of the edge leading to a total cost of 15106 € . In this optimal topology two edges (the two dashed lines) are not used (OOS-BKC and OOS-ATH). Globally we show that the capacity is not well balanced along the network because the values are inside a large interval [0-23] with an average of capacity equal to 11.62 and a standard deviation of 6.63 which leads to a big coefficient of variation (0.57).

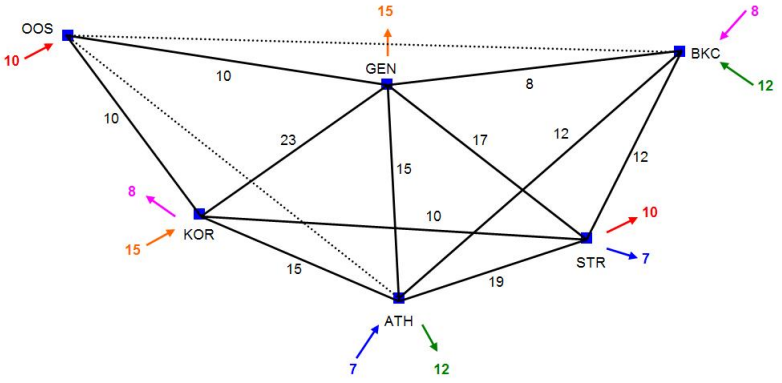


Figure 2.6: Optimal solution of the independent approach problem.

Figure 2.7 shows the optimal solution of the associated global approach. The total cost is here equal to 11095 € . The saving (here the saving is equal to (optimal solution of

the independent approach - optimal solution of the global approach divided by optimal solution of the global approach and expressed in percentage) in terms of cost is large, here around 36%, which motivates the utilization of the global approach. We see also that here the capacity is better balanced. All the edges of the network are used and the capacity is included in a smaller interval [1-10] with an average capacity of 7.77 and a standard deviation of 2.92 leading to a reasonable coefficient of variation (0.37).

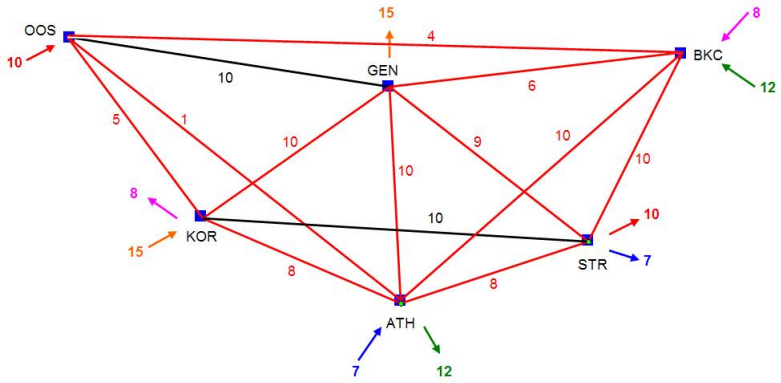


Figure 2.7: Optimal solution of the global approach problem.

In a second test phase we evaluate how fast the Benders decomposition model is compared to a basic implementation of the global model (Model 2) with solver CPLEX 8.0. And finally we wish to know what is the effect linked to the aggregation step in terms of CPU time.

Instances	IA Cost	GA Cost	Saving (%)
3/6/2	160	160	0
6/26/5	15106	11095	36
7/18/9	780	685	13
10/32/16	44446	33316	25
30/150/2	21327	14982	33
30/150/5	52221	32501	60
30/150/7	62126	39971	55
30/150/10	76768	51695	48

Table 2.2: Cost of the optimal solution.

Table 2.2 summarizes the test phase and gives us an idea about the performance of the independent approach. The first column represents the instance of the network tested. The second column gives the cost of the optimal solution found by using the independent approach. The third column represents the optimal solution found by using the Model 2, the global approach. Column four gives an idea of the potential savings linked to the use of the global approach. This table shows that the global model leads to substantial

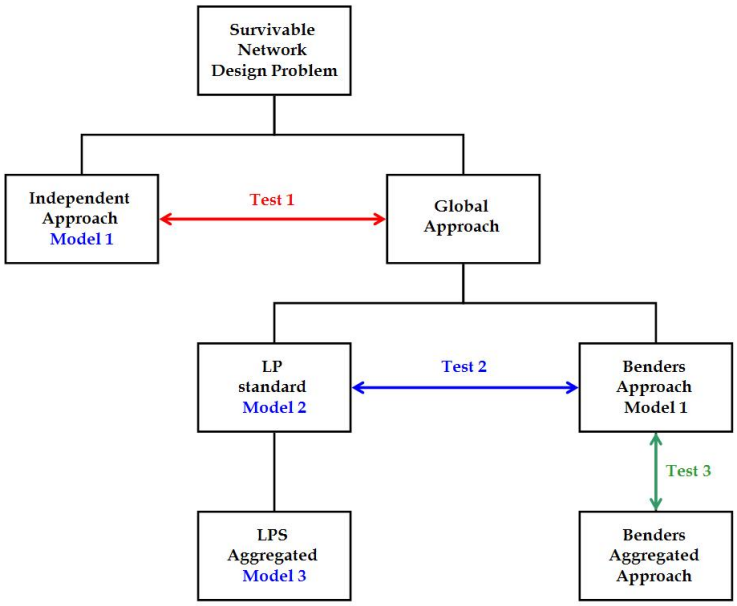


Figure 2.8: Illustration of the testing phase.

cost savings that increase with the size of the network. This leads to the conclusion that Models 2 and 3 and the associated Benders approach and Benders aggregated approach which all the four use global optimization, will be chosen if we want to find the most economical solution.

Next, we have to evaluate if the decomposed models (the Benders approach and the Benders aggregated approach) are better in terms of CPU time than the non-decomposed ones (Model 2 and Model 3). Table 2.3 shows again in the first column the different instance used. In column two, three, four and five, we find respectively the CPU time needed to find the optimal solution[7] for Model 1 (the independent approach), Model 2 (a basic implementation of the global model with solver CPLEX), the Benders approach and finally for the Benders aggregated approach. The last column gives the gap between the Benders aggregated approach and the Model 2.

Here, the conclusions are not so obvious as in the case of the previous evaluation. Indeed, for the small-scale problems (the first five instances), we see that the decomposed approaches take more time to find the optimal solution. This is mainly due to the fact that the decomposed programs have to jump from the master problem to the subproblems and need a big number of iterations to reach the optimal solution which is highly time consuming compared to the easy structure of these small networks. For easy and small problems like the first five, it is clear that they are easier to solve using the basic implementation with solver CPLEX than using the decomposed one. However, in case of large-scale problems, (notice that real problems are most likely to be large-scale prob-

[7]Notice that the solution found by the Model 1 is not in reality the optimal solution. Only Models 2, 3 and Benders approach gives us the optimal network topology with failure.

Instances	IA (Model 1)	GA (Model 2)	Benders	BendersAgg	Ratio $\frac{Model2}{BendersAgg}$
3/6/2	0.03	0.01	0.11	0 .11	0.09
6/26/5	0.11	0.07	1.52	1.22	0.05
7/18/9	0.09	0.06	1.21	0.57	0.10
10/32/16	0.23	1.69	5.07	2.55	0.66
30/150/2	0.79	5.58	26.04	21	0.26
30/150/5	1.09	322	165.83	82.61	3.89
30/150/7	1.28	452	258.77	76.75	5.88
30/150/10	1.58	889.13	440.39	66.66	13.33

Table 2.3: CPU time in sec needed to find the optimal solution.

lems than small-scale), we find some improvements in terms of CPU time for the two decomposed programs. As an example, the Benders aggregated approach is respectively around 4, 6 and 14 times faster than the basic implementation of the global Model 2 with CPLEX for the last three instances. These three biggest instances of our test set stay however small compared to instances of real life, that is the reason why we could expect better results if we test our Benders models on bigger real instances.

Furthermore if we compare the second and the fifth column, we see that in each case, it takes less time using the independent approach than using the decomposed approaches. Thus, in terms of times it seems that the independent approach gives the best solution whereas in terms of cost, the decomposed ones are better. People who works in the management world have to make the judicious trade-off between economy of cost and economy of time.

Finally, we want to evaluate if an aggregation by destination is profitable, in terms of CPU time, for the Benders decomposition model. In Table 2.3, as said before, we have the performance in terms of CPU time, and in Table 2.4 we find the performance in terms of number of variables included in each model.

Topology	GA	Benders	BendersAgg
3/6/2	39	30	30
6/26/5	1703	572	338
7/18/9	1467	657	216
10/32/16	8208	2832	592
30/150/2	22575	10200	10200
30/150/5	56325	16950	10200
30/150/7	78825	21450	10200
30/150/10	112500	28200	10200

Table 2.4: Comparison of number of variables used in the principal models and approaches.

The two tables are providing similar conclusions for this test phase. Doing an aggregation by destination decreases the number of variables and consequently the complexity of the

problems to solve and also, of course, the CPU time. In others words, we can say that the Benders aggregated approach is the best approach in terms of CPU time and also in terms of cost, for large-scale problems.

2.6 Summary

In this chapter we modeled the problem of designing survivable telecommunications networks. We propose to design a single network for both base traffic and restoration. That means that the capacity of our network is better used both in the case of failure or without failure than with methods which optimize a reserve network because they create a parallel network with capacity which is not used in the usual cases of non-failure.

Another specificity of our work is to take advantage of decomposition methods in order to improve the solving process. As seen in Section 2.5, the improvements are encouraging.

Moreover, aggregation allows us to decrease substantially the size of the model and leads to additional improvements and better convergence.

In addition to the decomposition method which leads to the optimal solution in a very acceptable time, we propose an alternative heuristic (the independent one) which converges faster than the decomposed one, but often with an excess of cost.

The experiments show that our final method is able to solve a large-scale problem up to around 10 times as fast as the basic implementation in CPLEX of the global model.

Chapter 3

On the stable set of size 2 polyhedron

3.1 Introduction

THis chapter focuses on the description of the polyhedron of the stable set of size 2 problem. A stable set of size 2 in a graph is a pair of nodes which are not connected by an edge. In graph theory, the stable set problem is well know to be one of the basic and most fundamental problems. As a consequence and as it is to be expected, lots of studies search to reduce their problem to a stable set structure to prove the NP-hardness because it is well know that the stable set problem is NP-hard. The problem of finding a maximum weight stable set of size 2 is obviously simple because by enumeration this problem can be solved in polynomial time. Nevertheless, finding the complete description of the associated polyhedron is not a trivial work.

This chapter is organized as follows, in Section 3.2 we develop a short literature review of the stable set problem. In Section 3.3 we propose a formulation for the problem and we prove that it gives a complete description of the associated polyhedron. Then, in Section 3.4, based on a theoretical result from Janssen and Kilakos in [66], we prove that our inequalities are facet defining. This short study has to be seen as a preliminary study to prepare readers for Chapter 4.

3.2 Literature review

Given a graph $G = (V, E)$ a stable set (also called independent set) S is defined as a subset of vertices $S \subseteq V$ which are not connected, i.e. $ij \notin E$ for all $i, j \in S$.

Figure 3.1 shows on the left a graph G composed of 6 vertices and 8 edges. On the centre of the Figure 3.1 we have a feasible stable set of size 2 composed of the two red nodes a and e. Finally the maximum stable set in G is the set composed of nodes a, e and c, (see Figure 3.1 on the right). Indeed it is not possible to find a subset of not connected nodes which contains more than 3 nodes in this graph G.

Given a graph $G = (V, E)$ and a weight w_i associated to each vertice $i \in V$, the weighted

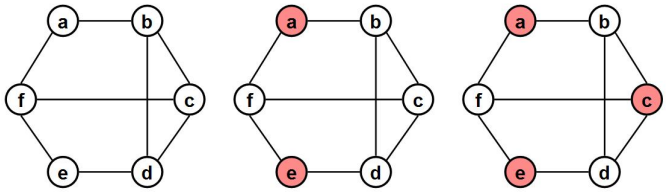


Figure 3.1: Graph G (left), feasible stable set (centre), maximum stable set (right).

stable set problem consists in finding a stable set of maximum weight. For every subset $S \subseteq V$, let $\mathcal{X}^S \in \mathbb{R}^V$ denote its incidence vector, i.e., the vector defined by:

$$\mathcal{X}_i^S = \begin{cases} 1 & if \quad i \in S \\ 0 & \text{otherwise} \end{cases}$$

The stable set polyhedron of G, denoted $STAB(G)$, is the convex hull of the incidence vectors of its stable sets, i.e. $STAB(G) = conv(\{\mathcal{X}^S : S \text{ is a stable set of } G\})$. A linear system $Ax \leq b$, is defining for $STAB(G)$ if $STAB(G) = \{x : Ax \leq b\}$ and hence the original optimization problem is equivalent to the linear program $max\{\sum_{i \in V} w_i x_i : Ax \leq b\}$.

As introduced before, the stable set problem is NP-hard, see for instance Grotschel *et al.* in [59], or Lovász in [78]. Therefore one would not expect to find a good defining linear system for the stable set polyhedron of general graphs. We define $\alpha(G)$ as the stability number of G which represents the maximum size of a stable set in G. It is well known that the computation of $\alpha(G)$ is also NP-hard. Although there is no hope to get a complete description, many interesting classes of facets, and more generally, of valid inequalities for $STAB(G)$ are known. We can first identify two classes of trivial valid inequalities:

$$x_i \geq 0 \quad i \in V, \tag{3.1}$$

$$x_i + x_j \leq 1 \quad ij \in E \tag{3.2}$$

Inequalities (3.1) are the non-negativity constraints and inequalities (3.2) are called the edge constraints which mean that if the edge ij belongs to G, then node i and j are adjacent and consequently both i and j can not be in the same feasible solution. Note that if G is bipartite, (3.1) and (3.2) suffice to completely characterize $STAB(G)$. Moreover the class of the edge constraints can be easily extended to the class of clique inequalities (3.3).

$$\sum_{i \in P} x_i \leq 1 \quad \text{where } P \text{ is a clique,} \tag{3.3}$$

By definition, in a graph G, a clique P is a set of vertices $P \subseteq V$ which are pairwise adjacent vertices, $ij \in E$ for all $i, j \in P$. As an example, in Figure 3.1, nodes b, c and d

form a clique because each pair of nodes of the clique are linked by an edge. Inequalities (3.1) and (3.3) suffice to describe $STAB(G)$ if and only if G is a perfect graph. Two others classes of valid inequalities are the odd hole inequalities (3.4) and the odd anti-hole inequalities (3.5). In graph theory, a cycle graph is a graph that consists of a single cycle. A cycle graph with n vertices and n edges is called C_n and every vertex has degree 2; that is, every vertex has exactly two edges incident with it. An induced cycle graph of odd length at least 5 is called an odd hole (Note that a odd cycle with length less than 5 is a clique). An induced subgraph that is the complement of an odd hole is called an odd anti-hole.

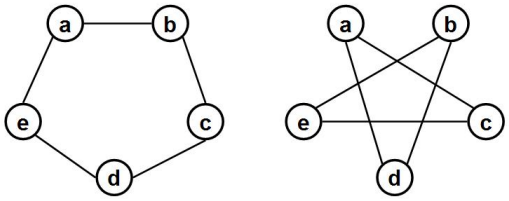


Figure 3.2: Odd hole graph with 5 nodes on the left and the associated odd anti-hole graph on the right.

$$\sum_{i \in C} x_i \leq \frac{|C|-1}{2} \quad \text{where } C \text{ is a cordless odd cycle,} \tag{3.4}$$

$$\sum_{i \in B} x_i \leq 2 \quad \text{where } B \text{ is the complement of a cordless odd cycle,} \tag{3.5}$$

Inequality (3.4) is facet defining if $(C = V)$. If $(B = V)$, then inequality (3.5) is facet defining. If there are nodes not occurring in the inequality then they may sometimes be added to constraints with non-zero coefficients by lifting process.

Nevertheless for particular graphs the defining linear systems $Ax \leq b$ are well know, for instance, bipartite graphs, line graphs, see Edmonds in [36], and graphs with no stable set of size three, see Shepherd in [98]. Remember that by definition a graph is bipartite (or bigraph) if it is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; that is, U and V are independent sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles. The weighted stable set problem for each of these classes is polynomially solvable. However, for a special class of graphs, which is the class of claw-free graphs, it is much more complicated to define the linear system $Ax \leq b$. By definition, a graph is claw-free if no vertice has three pairwise non-adjacent neighbours or if the graph does not contain any claw as an induced subgraph. In graph theory, a star S_k is the complete bipartite graph composed of $(k + 1)$ vertices and k edges, a star is a tree with one internal node and k leaves. A star with three edges is called a claw (see for instance Figure 3.3).

Minty [86] gives a polynomial time algorithm to solve the weighted stable set problem for the class of claw-free graphs based on the algorithm of Edmonds in [36, 37] to find a maximum weight matching in a graph. A matching (also called independent edge set)

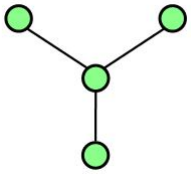


Figure 3.3: Representation of a claw.

is defined as a set of edges without common vertices. Given a graph $G = (V, E)$, a matching M in G is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex. As the matching problem is looking for independent edges set and as the stable set problem is looking for independent vertices set, both problems are closely linked. This link is very narrow between a graph G and its associated line graph $L(G)$. Indeed the line graph associated to G, denoted by $L(G)$ is the graph whose nodes are the edges of G. Two nodes of $L(G)$ are adjacent if the corresponding edges of G are incident. As a consequence, it is obvious that looking for a stable set in $L(G)$ is similar to looking for a matching in G. Figure 3.4 represents a graph G and the associated line graph $L(G)$. Note that given a graph G, the associated line graph $L(G)$ is claw-free, but the converse is not true. On the small example of Figure 3.4 it is easy to verify this property. Even if Minty gives a polynomial time algorithm, the stable set problem for claw-free graphs is not yet under control from the polyhedral point of view as there can occur arbitrarily complicated facets.

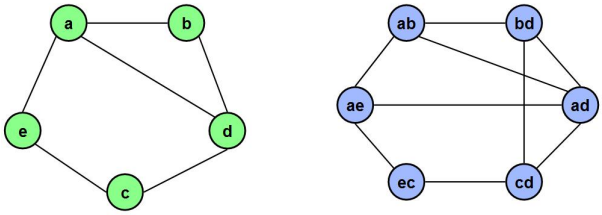


Figure 3.4: Representation of a graph G and its line representation $L(G)$.

Until now, there is not yet any complete description of the weighted stable set problem for claw-free graphs. Nevertheless big progresses were achieved by Ben-Rebea [94], in his thesis. There, he suggested a linear description for quasi-line graphs. By definition, a graph G is quasi-line if the neighbourhood of each node can be partitioned into two cliques. Note that all graphs G which are quasi-line are also claw-free, but the converse is not true. In [88] Oriolo and later on in [38] Eisenbrand *et al.* study the stable set polytope of quasi-line graphs and prove the Ben Rebea conjecture, which says that the stable set polytope of a quasi-line graph is completely described by clique-family inequalities, by showing that it also holds for fuzzy circular interval graphs. Indeed in [22] Chudnovsky *et al.* state that each fuzzy circular interval graph are also quasi-line graph and conse-

quently also claw-free graph. By definition, given a set of intervals $\{I_1, I_2, ..., I_n\}$ defined in a real line, the corresponding interval graph is $G = (V, E)$, where $V = \{I_1, I_2, ..., I_n\}$, and $I_i, I_j \in E$ if and only if $I_i \cap I_j \neq \emptyset$. A circular interval graph is an interval graph where the intervals set is plotted on a circle and not on a real line.

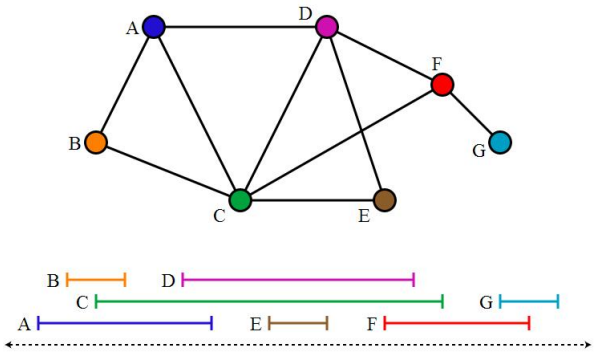


Figure 3.5: Seven intervals of the real line and the corresponding seven vertices interval graph G.

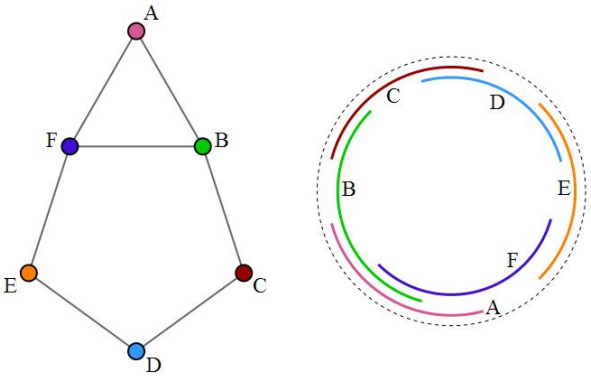


Figure 3.6: Example of Circular interval graph.

Giles and Trotter [48] showed that any complete defining system for stable set polyhedron of claw-free graphs must include several classes of more complicated inequalities. In [72] Liebling *et al.* extend the work of Giles and Trotter and deal with non-rank facets of the stable set polytope of claw-free graphs. They present, amongst other things, a complete description of the stable set polytope for graphs with stability number limited to 2.

In this chapter we focus on the stable set of size 2 problem for any graph G. The simple purpose of this problem's variation is given a weight w_i associated to each vertice i to find exactly two vertices which are not connected leading to the maximum weight. Figure 3.7 shows a graph G, each feasible solution of the stable set of size 2 problem is exactly composed of two vertices which are not connected in G. If we draw the comple-

mentary graph of G, called $G^{'}$, each pair of vertices linked by an edge in $G^{'}$ represents a feasible solution of our problem.

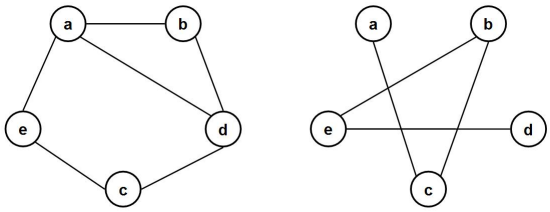


Figure 3.7: Original graph G (on the left) and its complementary graph $G^{'}$ (on the right).

3.3 Formulation

In this section we propose a formulation and we prove that it gives a complete description of the associated polyhedron for the stable set of size 2 problem. These results are direct consequences of results by Janssen and Kilakos in [66] but we provide here an alternative proof. Given a graph $G = (V, E)$ and a weight w_i associated to each vertice $i \in V$, the stable set of size 2 problem consists in finding two non-connected vertices leading to the maximum total weight. The following formulation $STAB-2$ models this problem.

$$
\begin{aligned}
& & \max \quad & \sum_{i \in V} w_i \, x_i & & (3.6)\\
(STAB-2) & & \text{s.t.} \quad & \sum_{i \in V} x_i = 2 & & (3.7)\\
& & & 2\sum_{i \in P} x_i + \sum_{i \in Q(P)} x_i \leq 2 & P \text{ a clique} & (3.8)\\
& & & x_i \geq 0 & i \in V & (3.9)
\end{aligned}
$$

Variables x_i is a continuous variable indicating the percentage of node i included in the solution. (3.6) is the objective function which maximizes the total weight of selected vertices. Inequality (3.7) ensures that at least two nodes are selected to form a feasible solution and inequalities (3.8) embrace edge inequalities as clique inequalities and also more complex kind of inequalities (which will be detailed later). In (3.8), P is considered as a clique and by definition $Q(P)$ is the neighbourhood of this clique P, thus $Q(P) = \{i \in V \setminus P : \forall j \in P, ij \in E\}$. $Q(P)$ is the set of nodes which are all connected to all the nodes of a given clique P. As a consequence, if P is a maximal clique $Q(P) = \emptyset$ and (3.8) is equivalent to (3.3) . Finally inequalities (3.9) are non-negativity constraints. Inequalities (3.8) combined with (3.9) ensure that $0 \leq x_i \leq 1$.

For more general problems, other things being equal, such that $X \subset \mathbb{Z}^n$ is an integer set and P is a formulation for X, so $P = \{x \in \mathbb{R}^n_+ : Ax \geq b\}$, $X = P \cap \mathbb{Z}^n$ and $conv(X) \subseteq P$, we can prove that $P = conv(X)$, or equivalently that P is integral by using the following methodology introduced by Lovász (see section 9.2.2. in [102]). For

all $c \in \mathbb{R}^n$ for which $M(c) \neq X$ and the optimum value is finite, $M(c) \subseteq \{x : a^q x = b^q\}$ for some inequality $a^q x \geq b^q$ defining P, where $P \cap \{x : a^q x = b^q\}$ is a proper face of P. Here $M(c) = \arg\min\{cx : x \in X\}$ is the set of optimal solutions for a given cost vector $c \in \mathbb{R}^n$. This condition is sufficient because if $\pi x \geq \pi_0$ defines a facet of $conv(X)$, and we take $c = \pi$, then $M(\pi)$ can only lie in that facet. It follows that the faces of P contain all the facets of $conv(X)$.

Theorem 1. *(STAB-2) gives a complete description of the stable set of size 2 polyhedron for any graph G.*

Proof.

a.) First, assume that there exists $j \in V$ such that $x_j = 0$ for all optimal solution, then the constraint $x_j = 0$ is saturated for all optimal solution and in this case it is the inequality (3.9) which is satisfied at equality for $i = j$.

b.) Otherwise for all $i \in V$, there exists an optimal solution such that $x_i = 1$. Then there exists $j(i) \in V$ such that $(w_i + w_{j(i)}) = z^*$ and $ij(i) \notin E$ with z^* be the value of the optimal solution, i.e. the maximum total weight. If we define $P = \{i \in V : w_j \leq w_i, j \in V\}$, as the set of vertices which have the same maximum weight $w_i = w_{max}$ and let $\overline{w} = (z^* - w_{max})$. Let $\overline{w} < w_{max}$. We know that P is a clique. If it is not the case an optimal solution can be built by taking two vertices from P leading to a total weight $z^{'} = (w_{max} + w_{max}) > z^*$ which is not possible as z^* is considered as the value of the optimal solution.

   i) Consider a solution composed of the two vertices $\{i, j\}$ with $i \in P$. Then $j \notin Q(P)$ as $Q(P)$ is composed of vertices which are all connected to all vertices of P, if an edge exists between i and j the pair $\{i, j\}$ can not be an optimal solution. Since $i \in P$ and $j \notin P \cup Q(P)$ then inequality (3.8) is satisfied at equality.

   ii) The last case to consider is a solution composed of two vertices $\{k, l\}$ such that $k, l \notin P$. We know that $\overline{w} < w_k < w_{max}$ and $\overline{w} < w_l < w_{max}$. If it is not the case any solution $\{i, k\}$ or $\{i, l\}$ with $i \in P$ could lead to a total weight higher than z^* which is not possible taking into account the definition of z^*. Assume $k \notin Q(P)$ then there exists $i \in P$ such that $ik \notin E$ and $\{i, k\}$ is a feasible solution. The total weight of this solution is $(w_k + w_{max}) > z^*$ which is a contradiction with the definition of z^*. Consequently k and $l \in Q(P)$ with $x_k = 1$ and $x_l = 1$ and inequality (3.8) is satisfied at equality.

All optimal solutions are included in one facet which ends the proof. □

3.4 Facets

In this section we derive conditions for inequalities (3.7), (3.8) and (3.9) to be facet defining. In [66] Janssen and Kilakos study the bounded stable sets problem. A k-stable set in a graph is a stable set of size at most k. They study the convex hull of the k-stable sets when $k = 2$ and $k = 3$. Theorem 3.1 from Janssen and Kilakos in [66] states that for any graph G with $\alpha(G) \leq 2$, the following system describes the stable set polytope of G and is totally dual integral and each inequality is facet defining:

$$\sum_{i \in P} x_i \leq 1 \qquad P \text{ a maximal clique} \qquad (3.10)$$

$$(STAB - JK) \qquad 2\sum_{i \in P} x_i + \sum_{i \in Q(P)} x_i \leq 2 \qquad P \text{ a clique} \qquad (3.11)$$

$$x_i \geq 0 \qquad i \in V \qquad (3.12)$$

$$\sum_{i \in V} x_i \leq 2 \qquad (3.13)$$

Note that inequalities (3.11) are defined for each clique P such that none of the connected components of the complement of $Q(P)$ is a bipartite graph. This formulation (STAB-JK) admits stable sets of sizes one or two. (STAB-2) is the face of (STAB-JK) defined by (3.13) and so all results for (STAB-JK) are valid for (STAB-2).

3.5 Summary

As a summary, in this chapter we give a study of the stable set of size 2 problem. After a short literature review of the general stable set problem we propose a formulation which gives the complete description of the polyhedron. We prove in this chapter that all optimal solutions of our problem are included in one facet of the polyhedron. Finally we prove that all our inequalities are facet defining. This chapter has to be seen as a preliminary chapter for a proof which will be demonstrated in the next chapter concerning the K edge (arc) disjoint L-paths problem.

Chapter 4

On the K edge (arc)-disjoint L-paths problem: Properties of a layered extended formulation

4.1 Introduction

Nowadays our worldwide society depends largely on the performance of huge information systems more than often organized in large-scale, complex and costly networks. With time, the material (routers, fiber optic cables, ...) worsens and the risk of failure is something that networks managers must control as well as possible in order to guarantee the best service to users. As a consequence, the development of survivable networks becomes a crucial field of research and investigation. Here, a survivable network is considered as a network which can route the various demands without loss of service quality even in case of failure (link or node failure). A telecommunications network is often represented by an undirected graph $G = (V, E)$, where V represents the node (vertice) set, and E the set of edges or potential links. Given two distinct nodes o (the origin node of demand) and d (the destination node of demand) of V, an *od*-path is defined as a sequence $P = (v_0, (v_0, v_1), v_1, ..., (v_{l-1}, v_l), v_l)$, where $l \geq 1$, $v_0, v_1, ..., v_l$ are distinct nodes, $v_0 = o$, $v_l = d$, and (v_{i-1}, v_i) is an arc connecting v_{i-1} and v_i (for $i = 1, ..., l$).

In the literature, survivability problems are generally tackled by two major rerouting techniques. The first one, called local rerouting, consists in rerouting the traffic between the extremities of the failed link. In this study we focus on the second technique called end-to-end rerouting. This technique considers all the demands affected by the failure and reroutes these demands from their respective origins to their respective destinations. In Chapter 2, we manage survivability by working on what we call "total rerouting" techniques because when an arc is broken we reroute systematically all the demands from their origins to their destinations. In this chapter to ensure the possibility of rerouting techniques (local, global,...) we suggest to find for each demand, at least K different paths from the origin node o to the destination node d. In order to construct these paths we have to add one more constraint: These K paths have to be, for instance, "edge-disjoint", i.e. if a particular edge belongs to one of the K paths, this particular edge

cannot be used by another path. It guarantees that, if a maximum of $(K-1)$ edges break down simultaneously, it is always possible to reroute all the demands since one feasible path remains functional. More formally, a collection $P_1, P_2, ..., P_k$ of od-paths is called edge-disjoint (respectively arc-disjoint or node-disjoint) if any edge (respectively, any arc or any node except for o and d) appears in at most one path. A subgraph H of G is called K edge-survivable (respectively, K arc-survivable or K node-survivable) if for any $o, d \in V$, H contains at least a predetermined number K of edge-disjoint (respectively, arc-disjoint or node-disjoint) od-paths. We have to point out that using "total rerouting" as it was done in Chapter 2 gives always a network in which there exist at least 2 edge-disjoint paths between each commodity.

However, new applications like video, audio or IP telephony... require small transmission delays (i.e. short paths must be used for each commodity) because the transmission time is one of the most important key criteria for customer satisfaction. As a consequence, the survivability constraints are not sufficient to guarantee a routing with a good quality of service since the primary path where the traffic flows or the secondary paths, if a path fails, may lead to unacceptable delays. Since in most of the routing technologies, delay is caused at the switching nodes because node processing times dominate over queueing delays, it is usual to measure the delay in a path in terms of its number of intermediate nodes, or equivalently, its number of arcs (here called number of hops). Thus, to guarantee the required quality of service, we impose a limit on the number of arcs of the routing paths, so that the traffic may be routed, or rerouted if a path fails, on a different path with a quality of service guaranteed in terms of delay. These, so called hop constraints also guarantee a certain level of transmission reliability in the sense that the probability that all the transmission lines in the path are working decreases with the number of arcs [103]. We say that a L-path is a path with at most L arcs (or hops). In this chapter, we are referring to problems where we want to minimize the cost of the design guaranteeing that between a single source and a single destination there exists K edge (or arc, or node)-disjoint L-paths.

As a consequence, we focus on end-to-end rerouting with hop constraints and in the literature, three major problems of this kind have been studied before. The hop-constrained spanning tree problem consists in finding the minimum spanning tree such that the path between the root and any other node has no more than L hops. This problem has been largely studied by Gouveia [51, 52] in which the author proposes a multicommodity network flow formulation and discusses the use of Lagrangian relaxation to improve the lower bound. In [58], Gouveia and Requejo proposed the hop-indexed formulation using a layered representation of the graph. In [26] Dahl gives a polyhedral study of the problem when the number of hops is limited to 2.

The hop-constrained shortest path problem consists in finding between two distinct nodes o and d a minimum cost path with no more than L hops with L fixed. In [27], Dahl and in [29], Dahl and Gouveia give a complete description of the polytope when $L \leq 3$ using the **jump inequalities** for which the separation problem can be solved in polynomial time. In [28], Dahl *et al.* extend the previous work when $L = 4$.

The third problem is the hop-constrained network design problem. This problem fo-

cuses on finding a minimum cost subgraph such that there exist at least K edge-disjoint paths between each pair of terminals. Moreover, each path must use at most L hops. In [10], Balakrishnan *et al.* give a mixed integer formulation for the problem when $K = 1$, they derive a Lagrangian relaxation to improve the lower bound. In [90], Pirkul *et al.* propose multicommodity network flow models and some heuristics based on linear relaxation. In [31], Dahl *et al.* study the problem where there are for each pair of terminals K edge-disjoint paths of maximum length L. They prove that this problem is NP-hard even when $K = 1$ and $L = 2$. In [63], Huygens *et al.* study the same problem but when there is only one commodity, or one pair of terminals. They formulate an IP model when $K = 2$ and $L = 3$ in the space of the design variables. Bley [18] shows that the single demand node-disjoint length-restricted paths is NP-complete.

This chapter focuses on a mix of the second and the third category of problems, something between the hop-constrained shortest path problem and the hop-constrained network design problem because the objective is to find between only one pair of terminals, one origin node o and one destination node d exactly K edge-disjoint paths composed by a maximum of L hops. It is not really similar to the hop-constrained shortest path problem because in this problem the objective is only to find one path between a pair of terminals and here we want to find K paths. And it is not really a design problem because we assume that the number of commodity is strictly equal to one in this chapter. We decide to analyze first this "simple" single commodity problem because in the literature most of the studies propose models which formulate the problem only in terms of the design variables, we choose here to use a formulation based on an extended space of variables. We show how the extended hop-indexed model can be suitable for the K edge (arc or node)-disjoint hop-constrained network design problem in the single commodity case which is similar to the K edge (arc or node)-disjoint L-paths problem. Note that in Chapter 5 we extend naturally this work to the multicommodity case. Finally we prove here that our formulation gives a complete description of the associated polyhedron for:

- $\diamond$ $K = 1$ and any value of L for the arc and the edge-disjoint cases,
- $\diamond$ $L = 3$ and any value of K for the arc-disjoint case,
- $\diamond$ $L = 3$ and $K = 2$ for the edge-disjoint case.

We illustrate also the quality of the lower bound proposed by the extended formulation when $L > 3$ through some numerical results.

This chapter is organized as follows: In Section 4.2 we first give a well known formulation expressed in terms of design variables z. Then we introduce an alternative extended formulation using flow variables x. In Section 4.3 we explain the main differences between the arc-disjoint and the edge-disjoint cases and we show that the link between both disjointness cases are strongly dependent of the value of the parameter L. In Section 4.4 we demonstrate nice properties concerning the hop-indexed formulation introduced before for given value of parameters K and L. In Section 4.5 we dissociate the routing and the design problems. Finally, in Section 4.6, we evaluate how the hop-indexed model is suitable for the problem under study by evaluating the average gap between the optimal integer solution and the optimal solution of the linear relaxation for large values of parameters L and K.

4.2 Towards an hop-indexed formulation

4.2.1 Formulation in the space of the design variables z

Given a graph $G = (V, E)$ where V is the vertices set and E is the edges set, and given a pair of terminal o the origin node and d the destination node and parameters L and K, most of the publications in this area (see for instance Huygens *et al.* in [63, 62, 61, 30] and Diarrassouba in [34]) propose a formulation of the problem expressed with design z variables. One weakness of this kind of formulation is that it is difficult to find generalized cuts for different values of parameters K and L. A second weakness is that the number of cuts explodes when L and K take large values. Given a function $c : E \to \mathbb{R}$ which associates a cost c_{ij} to each edge $ij \in E$, the problem can be formulate as follows for $L = 3$, for instance:

$$\begin{aligned}
\min \quad & \sum_{ij \in E} c_{ij} z_{ij} & & \\
\text{s.t.} \quad & z(\delta(W)) \geq K, & W \subset V, & \quad (4.1)\\
& z(T) \geq K, & \text{L-path-cuts } T, & \quad (4.2)\\
& z_{ij} \in \{0, 1\}, & ij \in E. & \quad (4.3)
\end{aligned}$$

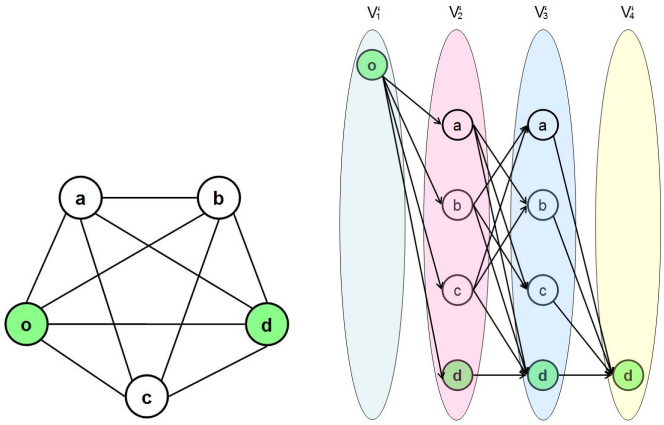
Inequalities (4.1) are *od*-**cut inequalities** and ensure that there exist exactly K paths between source node o and destination node d. By definition if $W \subset V$ is a node subset of G, then the set of edges that have only one node in W is called a cut and denoted by $\delta(W)$. A cut $\delta(W)$ such that $o \in W$ and $d \in \overline{W} = V \backslash W$ will be called an **od-cut** . Inequalities (4.2) are called **L-od-path-cut inequalities** and ensure that the length of the paths is at most L hops. Let $V_0, V_1, ..., V_{L+1}$ be a partition of nodes set V such that $o \in V_0$, $d \in V_{L+1}$ and $V_i \neq \emptyset$ for all $i \in \{1, ..., L\}$. Let T be the set of edges ij where $i \in V_u$, $j \in V_v$ and $|u - v| > 1$. Then inequality $z(T) \geq 1$ is valid for the hop-constrained shortest path problem. For K paths the right hand side is replaced by K.

This formulation lacks inequalities to describe the problem for $L \geq 4$ for instance. Introducing the **two-layered 4-path-cut** Huygens *et al.* ([62]) describe the problem for $L = 4$ and $K = 2$. However more inequalities are still needed to propose a valid formulation for more general cases. That is the reason why we decide to use a more intuitive model which can be easily adapted to be valid for all K and all L.

4.2.2 Alternative layered graph

The work of Gouveia [52] on hop-constrained tree problems suggests that the layered graph approach (leading to the so-called hop-indexed model) is a useful tool for modeling network design problems with hop constraints. This approach has been used with significant success, later on, on related telecommunications problems such as [57],[56]. That is the reason why, in order to propose a good model for the K edge-disjoint hop-constrained network design problem in the single commodity case, we use this alternative layered representation. This representation consists in transforming the problem defined

in the original graph into an equivalent problem defined in a different graph with L layers as it is illustrated in Figure 4.1.



(a) (b)

Figure 4.1: Basic network (a) and its Layered representation (b) when $L = 3$

From the original non-directed graph $G = (V, E)$, we create a directed layered graph $G^{'} = (V^{'}, A^{'})$, where $V^{'} = V_1^{'} \cup \ldots \cup V_{L+1}^{'}$ with $V_1^{'} = \{o\}$, $V_{L+1}^{'} = \{d\}$ and $V_l^{'} = V \backslash \{o\}$, $l = 2, \ldots, L$. Let v_l be the copy of $v \in V$ in the $l - th$ layer of graph $G^{'}$. Then, the arcs sets are defined by $A^{'} = \{(i_l, j_{l+1}) \mid i_l \in V_l^{'}, j_{l+1} \in V_{l+1}^{'}, l \in \{1, \ldots, L\}\}$, see Figure 4.1 (b). In the sequel, an (undirected) edge in E with endpoints i and j is denoted ij while a (directed) arc between $i_l \in V_l^{'}$ and $j_{l+1} \in V_{l+1}^{'}$ is denoted by (i, j, l). As a consequence we build $(L - 1)$ copies of nodes creating exactly L copies (or layers) of arcs in between. Note that finding a path composed of at most L hops in the original graph G is similar to finding a path in the layered graph $G^{'}$. Because the number of layer of arcs is exactly equal to L in $G^{'}$, it means that all feasible paths in $G^{'}$ is exactly composed of L hops. We need nevertheless to add something to allow paths composed by a number of hops which is less than L. The solution is to add in the layered graph a special kind of arc (d, d) with an associated cost equal to 0. Indeed if the length of the path is less than L, it means that the destination node d is reached before the last copy of node subset. To keep the flow conservation condition satisfied we need to create this special kind of arc. Note that each path between o and d in the layered graph is composed of exactly L arcs (hops) (which corresponds to a maximum of L arcs (hops) in the original one). In fact this is the main idea of this transformation since in the layered graph, any path is feasible with respect to the hop constraints. This alternative representation of the original graph G gives us additional information the about paths because now we know exactly in which position an edge ij is used.

Figures 4.2 and 4.3 show an illustration of the problem. Here we plot a simple network composed of 5 nodes and 10 potential edges. If the objective is to find 2 edge-disjoint paths with maximum 3 hops between origin node o and destination node d, a feasible solution could be the one proposed by the red and the blue paths of Figure 4.2. This

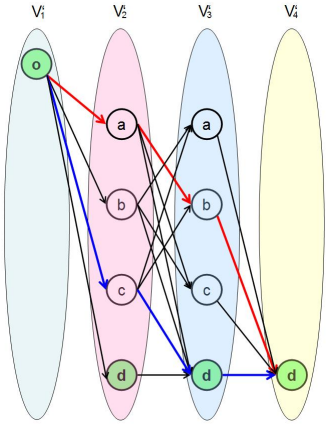


Figure 4.2: Feasible solution for $K = 2$ and $L = 3$ in the layered graph G'.

solution is feasible for G' but also for the original graph G because each selected edge is either blue or red. The right part of Figure 4.3 shows the selected edges needed to have a feasible solution. As it is here a design problem, network providers will install capacity only on these 5 selected edges in order to ensure survivability but also quality of service.

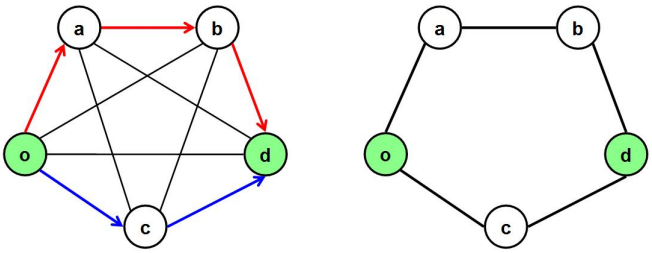


Figure 4.3: Feasible solution for $K = 2$ and $L = 3$ in the original graph G.

The objective of this chapter is to formulate the problem here with the hop-indexed model but also to prove that for some values of K and L our model gives a complete description of the associated polyhedron.

4.2.3 Formulation in the space of the flow variables x

In this section we show how it is easy to play with the hop-indexed formulation to obtain valid models for any L and any K for the edge-disjoint, the arc-disjoint and the node disjoint cases. Then in the following sections we focus only on the edge and the arc disjoint cases. The node-disjoint case was largely studied in [57], [56] and [18].

In order to model these problems we need to guarantee first the existence of K L-paths between the source node o and the destination node d. To be sure of that, we use a usual network flow formulation with simple flow conservation constraints and we send from

the source node a flow of K units. Because we formulate our model using the layered representation we are sure that the length of the K paths is composed of at most L links. We obtain the following model:

$$\min \quad \sum_{ij \in E} \sum_{l \in \{1,\ldots,L\}} c_{ij} \ (x^l_{ij} + x^l_{ji})$$

$$(Hop) \quad \text{s.t.} \quad \sum_{j:(j,i,l-1) \in A'} x^{l-1}_{ji} - \sum_{j:(i,j,l) \in A'} x^l_{ij} = \begin{cases} -K & \text{if} \quad (i = o) \\ K & \text{if} \quad (i = d) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases},$$

$$i \in V', l \in \{1, \ldots, L+1\}, \quad (4.4)$$

$$x^l_{ij} \quad \in \{0,1\}, \qquad (i,j,l) \in \{A' \setminus (d,d,l)\}, \quad (4.5)$$

$$x^l_{dd} \quad \geq 0 \text{ and integer}, \qquad l \in \{2, ..., L\}. \quad (4.6)$$

This formulation models the so-called "routing optimization problem". The goal is to determine the K edge or arc disjoint paths leading to the less expensive routing cost[1]. Flow or routing variable x^l_{ij} models the quantity of information sent on arc (i,j) when this arc is used in position l in one of the K paths. Compared with the design variables z, we use now x variables with one more index indicating at which position an arc (i,j) is used. Constraints (4.4) are traditional flow conservation constraints ensuring that the totality of the flow going from node o reaches node d and constraints (4.5) are trivial binary constraints. Constraints (4.6) are needed to ensure paths using less than L hops. For instance, for $L = 3$ and $K = 2$, the set of the two following paths $(o - i - d)$ and $(o - j - d)$ composed of 2 hops is a feasible solution (for the arc and the edge-disjoint versions). In the layered representation $(o - i - d)$ (respectively $(o - j - d)$) becomes $(o - i - d - d)$ (respectively $(o - j - d - d)$) so that $x^L_{dd} = 2$.

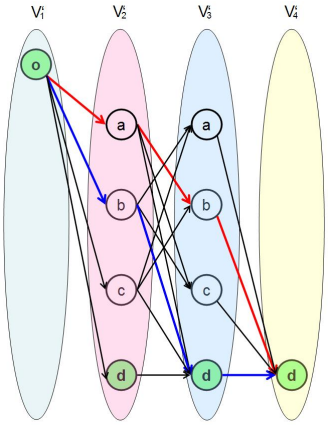


Figure 4.4: Feasible flow in the layered graph G'.

A nice property of this network flow model is that all optimal solutions are integer because the matrix of constraints composed of (4.4), (4.5) and (4.6) is totally unimodular.

[1] At the end of this chapter we develop the difference between routing and design problems

Unfortunately, this formulation is not sufficient in our case to model exactly the K edge (arc) disjoint L-paths problem. Indeed, the feasible flow in the layered graph $G^{'}$ (depicted on Figure 4.4) is not feasible in the original graph G because both red and blue paths use the same arc (b, d), but in different positions. If network providers decide to use this topology, users can be penalized if a failure touches this arc (b, d). In this case, no solution exists to send information between node o and d.

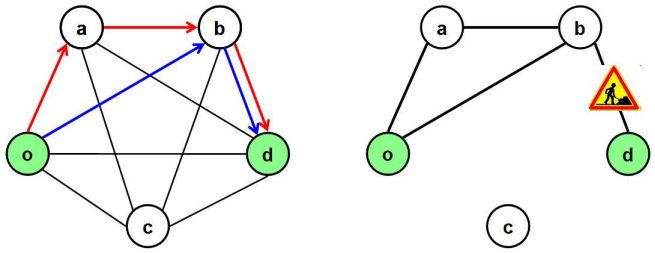


Figure 4.5: Feasible flow in the layered graph is not systematically a feasible flow in the original graph.

As a consequence the natural model Hop need to add new kind of cuts (called here survivability cuts) to avoid this overlapping situation. In the case of arc-disjointness the following valid inequality can be added:

$$\sum_{l\in\{1,\dots,L\}} x^l_{ij} \leq 1, \quad (i,j) \in A, \tag{4.7}$$

The interpretation of this cut says that if an arc (i, j) is used in one of the K paths in position l then this arc can not be used again in another path. We can easily adapt this cut for the case of edge-disjointness:

$$\sum_{l\in\{1,\dots,L\}} (x^l_{ij} + x^l_{ji}) \leq 1, \quad ij \in E, \tag{4.8}$$

or for the case of node-disjointness:

$$\sum_{l\in\{1,\dots,L\}} \sum_{j:ij\in E} x^l_{ij} \leq 1, \quad i \in V \setminus \{o, d\}, \tag{4.9}$$

(4.7) are called arc survivability inequalities as (4.8) are called edge survivability inequalities and (4.9) are called node survivability inequalities. We build three models, HopA for the arc-disjoint case, HopE for the edge-disjoint case and HopN for the node-disjoint case.

Model HopA

$$
\begin{aligned}
\min \quad & \sum_{ij \in E} \sum_{l \in \{1,\ldots,L\}} c_{ij}\ (x^l_{ij} + x^l_{ji}) & & \\
\text{s.t.} \quad & \sum_{j:(j,i,l-1) \in A'} x^{l-1}_{ji} - \sum_{j:(i,j,l) \in A'} x^l_{ij} = \begin{cases} -K & \text{if } (i = o) \\ K & \text{if } (i = d) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, & & \\
(HopA) \quad & & & i \in V', l \in \{1,\ldots,L+1\}, \\
& \sum_{l \in \{1,\ldots,L\}} x^l_{ij} \leq 1, & & (i,j) \in A, \\
& x^l_{ij} \in \{0,1\}, & & (i,j,l) \in \{A' \setminus (d,d,l)\}, \\
& x^l_{dd} \geq 0 \text{ and integer}, & & l \in \{2,...,L\}.
\end{aligned}
$$

Model HopE

$$
\begin{aligned}
\min \quad & \sum_{ij \in E} \sum_{l \in \{1,\ldots,L\}} c_{ij}\ (x^l_{ij} + x^l_{ji}) & & \\
\text{s.t.} \quad & \sum_{j:(j,i,l-1) \in A'} x^{l-1}_{ji} - \sum_{j:(i,j,l) \in A'} x^l_{ij} = \begin{cases} -K & \text{if } (i = o) \\ K & \text{if } (i = d) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, & & \\
(HopE) \quad & & & i \in V', l \in \{1,\ldots,L+1\}, \\
& \sum_{l \in \{1,\ldots,L\}} (x^l_{ij} + x^l_{ji}) \leq 1, & & ij \in E, \\
& x^l_{ij} \in \{0,1\}, & & (i,j,l) \in \{A' \setminus (d,d,l)\}, \\
& x^l_{dd} \geq 0 \text{ and integer}, & & l \in \{2,...,L\}.
\end{aligned}
$$

Model HopN

$$
\begin{aligned}
\min \quad & \sum_{ij \in E} \sum_{l \in \{1,\ldots,L\}} c_{ij}\ (x^l_{ij} + x^l_{ji}) & & \\
\text{s.t.} \quad & \sum_{j:(j,i,l-1) \in A'} x^{l-1}_{ji} - \sum_{j:(i,j,l) \in A'} x^l_{ij} = \begin{cases} -K & \text{if } (i = o) \\ K & \text{if } (i = d) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, & & \\
(HopN) \quad & & & i \in V', l \in \{1,\ldots,L+1\}, \\
& \sum_{l \in \{1,\ldots,L\}} \sum_{j \in N} x^l_{ij} \leq 1, & & i \in V, \\
& x^l_{ij} \in \{0,1\}, & & (i,j,l) \in \{A' \setminus (d,d,l)\}, \\
& x^l_{dd} \geq 0 \text{ and integer}, & & l \in \{2,...,L\}.
\end{aligned}
$$

It is important to bring to the fore that all these three formulations are valid for any L and any K which was not the case for models developed in the space of the design variables z. Now we have to determine if the linear relaxation of models HopA, HopE and HopN gives a complete description of the associated polyhedron.

4.3 Arc-disjoint versus Edge-disjoint

In this section we study how the three different survivability approaches are linked. First, given L and K, let $\mathcal{X}_A$, the set of all feasible arc-disjoint solutions of model HopA, $\mathcal{X}_E$, the set of all feasible edge-disjoint solutions of model HopE and $\mathcal{X}_N$, the set of all feasible node-disjoint solutions of model HopN. It is obvious by definition that $\mathcal{X}_N \subset \mathcal{X}_E \subset \mathcal{X}_A$. Indeed all edge-disjoint solution is of course arc disjoint, and each node disjoint solution is also edge-disjoint.

Now, if we consider the $L = 3$ case and if we assume that the routing costs are non-negative we can affirm that $\mathcal{X}_A^*$ (the set of all optimal arc-disjoint solutions of model HopA) $= \mathcal{X}_E^*$. This is illustrated on Figure 4.6. On the left part of the Figure 4.6, we have a feasible integer routing for the arc-disjoint case. The blue and the red paths are arc-disjoint because each arc is used only one time. This feasible routing is not optimal with non-negative costs because a non optimal loop exists between node i and j. This loop leads to additional costs compared with situation of the right part of the Figure 4.6. If we remove the loop we obtain again a feasible integer routing which is obviously less expensive. As a consequence using a loop is never optimal and thus using arcs (i, j) and (j, i) together in different paths is never optimal. This is similar to say that $\mathcal{X}_A^* = \mathcal{X}_E^*$. The same thing appear when $L = 4$ (see Figure 4.7). Indeed when we remove the loop from the left part of the Figure 4.7, two new feasible paths appear, one composed with 2 hops and the others with 4 hops. Thus $\mathcal{X}_A^* = \mathcal{X}_E^*$ is also true with non-negative costs.

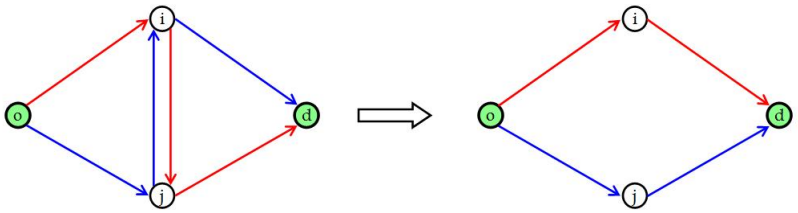


Figure 4.6: Feasible arc-disjoint routing (a.) vs feasible edge-disjoint routing (b.) for $L = 3$

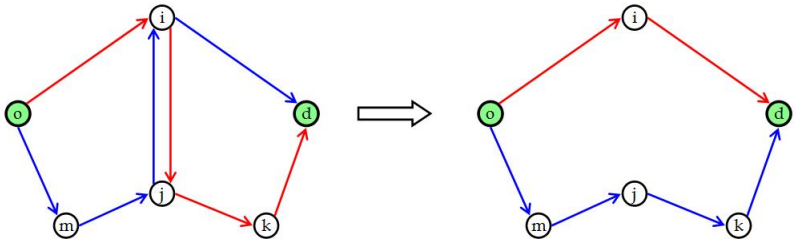


Figure 4.7: Feasible arc-disjoint routing (a.) vs feasible edge-disjoint routing (b.) for $L = 4$

Property 1. *The arc-disjoint case is similar to the edge-disjoint case in terms of optimal solutions when $L = 3, 4$ and when the costs are non-negative leading to $\mathcal{X}_A^* = \mathcal{X}_E^*$.*

However, for $L = 5$, this is not true. Indeed as shown on Figure 4.8 if we remove the loop from the left part of the Figure we obtain a solution which is not feasible because one path uses 2 hops and the other uses 6 hops. Consequently when $L = 5$, $\mathcal{X}_A^* \neq \mathcal{X}_E^*$ because there exist arc-disjoint optimal solutions which are not edge-disjoint.

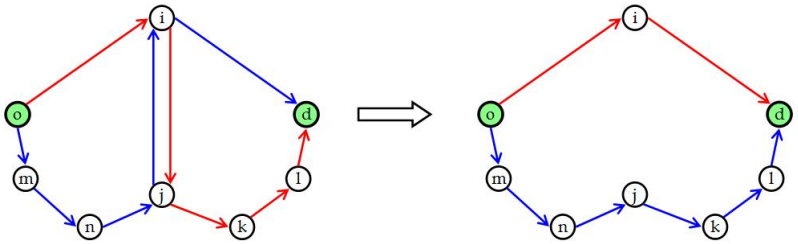


Figure 4.8: Feasible arc-disjoint routing (a.) vs unfeasible edge-disjoint routing (b.) for $L = 5$

Property 2. *The arc-disjoint case is different to the edge-disjoint case in terms of optimal solution when $L > 5$ leading to $\mathcal{X}_A^* \neq \mathcal{X}_E^*$.*

4.4 Properties of the Hop-indexed formulation

We want to prove now that these formulations (HopA and HopE) give the complete description of the associated polyhedron for special values of K and L.

We know that if we ignore constraints (4.7) or (4.8), then the extreme points of the linear programming relaxation are integer. This simply follows from the fact that the reduced models are similar to a natural network flow problem in the expanded layered graph (and thus, the corresponding matrix is totally unimodular). In general, this property is lost when constraints (4.7) or (4.8) are added. For a few cases we can still prove integrality.

In fact, and as far as we know, it is interesting to point out that this question has not even been addressed for the case with one single path, that is when $K = 1$.

Proposition 1. *For each value of L and $K = 1$, the optimal solution of the linear programming relaxation of models HopA and HopE are integer when $c_{ij} > 0, (i, j) \in A$.*

Proof. Let us assume that the optimal solution is fractional. Let $P_1, \ldots, P_T$ be the set of paths used in that solution, ordered by non-decreasing cost. Let α_i be the flow carried on path P_i (clearly $\sum_{i=1}^{T} \alpha_i = 1$). It is easy to see that:

(i) P_1 must carry the maximum allowed flow since it is the shortest path,

(ii) P_1 must use at least one arc (i, j) (respectively one edge ij) in different positions. Otherwise, this path would define an optimal integer solution for our problem.

However, if an arc (respectively an edge) is used twice in that path, the path contains a cycle. Since costs are positive, sending the same amount of flow on the cheaper path obtained by removing this cycle leads to a cheaper solution. Hence a contradiction and the solution cannot be fractional. □

Note that we assume here that costs are position independent which means that using an arc (i, j) in position l has the same cost than using the same arc but in another position. Indeed, in more complex problems (for instance if the congestion level of an arc is needed) we could assume that using a given arc (i, j) in position l is more (or less) expensive than using the same arc but in position $(l + q)$.

Proposition 2. *For $L = 3$ and $K \geq 1$, the polyhedron defined by the constraints of the linear relaxation of model HopA is integral.*

Proof. To prove the integrality, we demonstrate that the matrix H defined by constraints (4.4), (4.7) and the non-negativity constraints on x variables of our model HopA is totally unimodular (here after referred as TUM) using the following theorem from [102].

Theorem 2. *The following statements are equivalent.*

i. H is TUM.

ii. For every $J \subseteq N = (1, ..., n)$, there exists a partition J_1, J_2 of J such that $|\sum_{j \in J_1} h_{ij} - \sum_{j \in J_2} h_{ij}| \leq 1$, for $i = 1, ..., m$

As said by Wolsey in [102]: *Note that because H is TUM if and only if its transpose is TUM, statement ii. can equivalently be phrased in terms of partitions of subsets of rows of H; that is, for every $Q \subseteq M = (1, ..., m)$, there exists a partition Q_1, Q_2 of Q such that $|\sum_{i \in Q_1} h_{ij} - \sum_{i \in Q_2} h_{ij}| \leq 1$, for $j = 1, ..., n$.* This extension of the theorem states that for any subset of constraints Q (for any subset of rows), and for each variable (for each column of the matrix), the sum of the coefficients included in Q_1 (the sum of the elements of the column in Q_1) minus the sum of the coefficients included in Q_2 (the sum of the elements of the column in Q_2) has to be at most equal to 1.

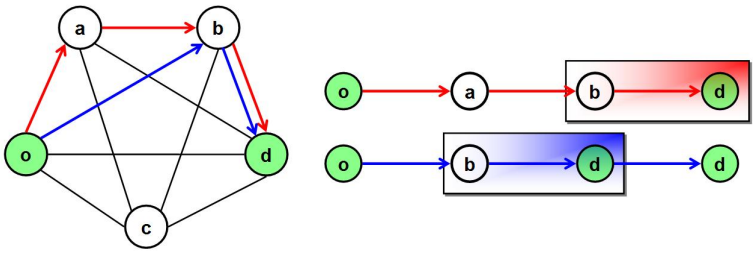


Figure 4.9: Characteristics of paths when $L = 3$

When $L = 3$ and $K \geq 1$, as shown on Figure 4.9, it exists only one way to use the same arc (i, j) in different positions: one path using arc (i, d) in position 2 (respectively 3) and another path using the same arc (i, d) in position 3 (respectively 2). Note that only arcs of the type (i, d) are arcs that can be used more than once (in the case at hand more than once means exactly twice because it is not possible to use a particular arc (i, d) in three different positions when $L = 3$ as in position 1 we find always an arc like (o, i)). Based on this observation, inequalities (4.7) are equivalent to:

$$x^2_{id} + x^3_{id} \leq 1, i \in V \setminus \{o, d\} \tag{4.10}$$

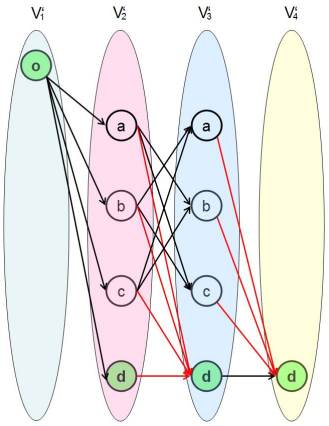


Figure 4.10: Layered representation when $L = 3$

We have to prove that by replacing inequalities (4.7) in model HopA, by inequalities (4.10), the optimal solution of the linear relaxation is always integer. Here, our matrix H is made up of two kind of constraints: the flow conservation constraints (4.4) and the new version of the arc survivability constraints (4.10). We know from Wolsey in [102], that a matrix B only made up of flow conservation constraints is totally unimodular. In order to prove that our matrix H is totally unimodular, we first, based on Figure 4.10, eliminate the variable x^3_{dd} using flow conservation constraints at destination node d, leading to:

$$\sum_{i:(i,d,2)\in A'} x^2_{id} + \sum_{i:(i,d,3)\in A',(i\neq d)} x^3_{id} = K \tag{4.11}$$

This reformulation modifies the structure of our matrix H, assuming H' the new matrix. By proving that H' is totally unimodular we prove also that the linear relaxation of our model HopA gives a complete description of the associated polyhedron. For the simple example illustrated on Figure 4.1, we can look at the structure of this new H' matrix. In this matrix we find $(2 + (L - 1)(|V| - 1))$ flow conservation constraints and $(|V| - 2)$ constraints (4.10).

Looking at the structure of the modified constraints matrix, we can easily see that only

x^1_{oa}	x^1_{ob}	x^1_{oc}	x^1_{od}	x^2_{ab}	x^2_{ac}	x^2_{ad}	x^2_{ba}	x^2_{bc}	x^2_{bd}	x^2_{ca}	x^2_{cb}	x^2_{cd}	x^2_{dd}	x^3_{ad}	x^3_{bd}	x^3_{cd}	RHS
-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-K
1	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	-1	-1	-1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	1	0	0	0	-1	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	-1	0	0
0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	-1	0
0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	1	1	K
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1

Table 4.1: H' matrix for the example with 5 nodes

variables x^2_{id} and x^3_{id} $(i \neq d)$ lead to columns of the matrix with more than two non-zero elements. Using this observation, we can construct the partition Q_1 and Q_2 as follows.

a.) If inequality (4.11) is in Q, we build Q_1 with all the flow conservation constraints of Q and with (4.11). Q_2 is built with all the inequalities (4.10) that are included in Q. By construction, for each column x^l_{id}, $l = 2, 3$, the sum of the elements in Q_1 is 0 or 1. The same holds for Q_2, so the absolute value of the difference is always less or equal to 1.

b.) If inequality (4.11) is not in Q, we build Q_1 with all the flow conservation constraints of Q and with all the inequalities (4.10). Q_2 is thus empty. The sum of the elements in Q_1 has an absolute value less than 1, and so does the absolute value of the difference since Q_2 is empty.

We can conclude that the transformed constraint matrix is totally unimodular, and thus the polyhedron defined by the constraints of the linear relaxation of HopA is integral. Then, as a consequence, for $L = 3$ and $K \geq 1$, the optimal solution of the linear relaxation of model HopA is always an integer routing. □

Proposition 3. *For $L = 3$ and $K \geq 1$, the optimal solution of the linear relaxation of model HopE is integer if $c_{ij} > 0, ij \in E$.*

Proof. As shown before both arc and edge-disjoint versions are naturally linked. Nevertheless, the edge case is a more restricted and complicated case. Some feasible solutions of the arc-version are not feasible for the edge-version (As illustrated on Figure 4.11(a)). Indeed arc (i, j) and arc (j, i) are both used once, but edge ij is used twice.

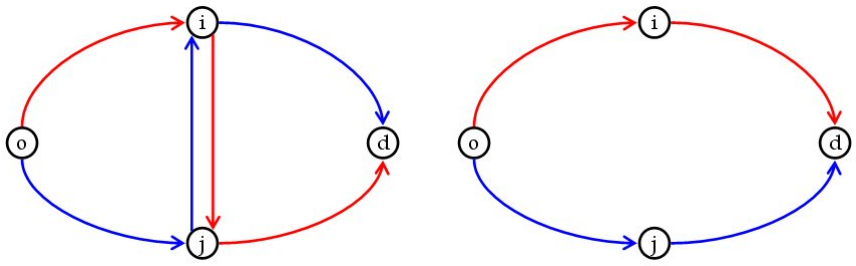


Figure 4.11: Feasible arc-disjoint solution (a) and feasible arc or edge disjoint solution (b).

Note that if $c_{ij} > 0$, this solution, which uses a loop between nodes i and j, is dominated by the one of Figure 4.11(b). Then when $c_{ij} > 0$, using a loop is never optimal leading to the fact that the edge-version is strictly identical to the arc-version for which we have the complete description of the polyhedron. □

When costs can be negative, $c_{ij} \leq 0$, then the previous proposition is no more true. Indeed we can obtain fractional optimal routing like the one presenting on Figure 4.12. This fractional point exists because the edge ij with a negative cost is used twice. Here, it is profitable to use the two arcs (i, j) and (j, i) in both direction which build a negative loop between two nodes and because the two arcs are linked together by constraints (4.8)

for edge ij, the optimal solution is obviously fractional. The objective now is to find new valid inequalities which can cut this first structure of fractional point.

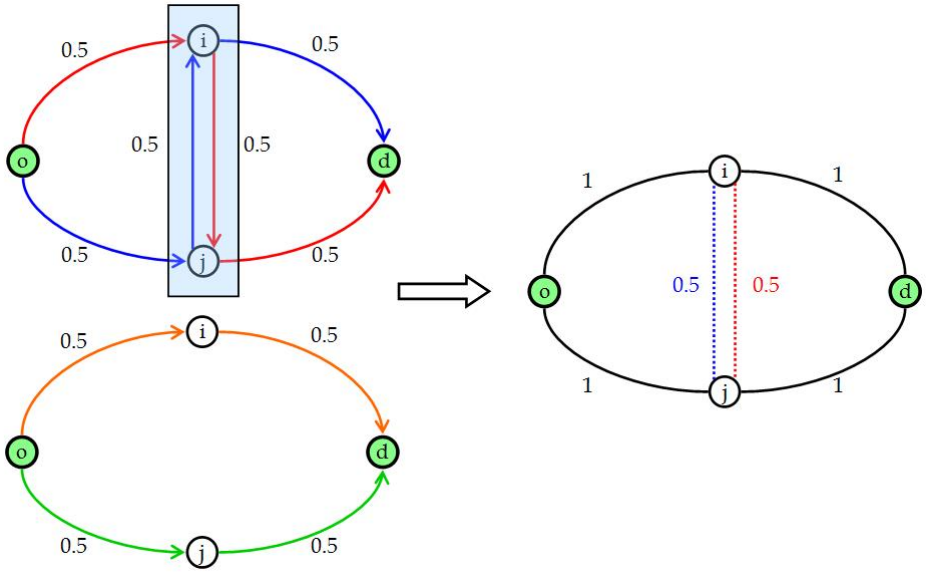


Figure 4.12: Illustration of the first fractional routing case: loop between 2 nodes.

We identify a first category of new cuts:

$$x^2_{ij} + x^2_{ji} + x^2_{id} \leq 1, (i,j) \in A | (i \neq j) \text{ and } (i, j \neq o) \text{ and } (i, j \neq d), \tag{4.12}$$

This cut, for each arc (i,j) *s.t.* $(i \neq j)$ and $(i, j \neq o)$ and $(i, j \neq d)$, is based on the previous cut (4.8) lifted by the term x^2_{id}. The main idea is the following: If x^2_{ij} or x^2_{id} is used in the solution it means that x^1_{oi} is also used, then $x^2_{id} = 0$ or $x^2_{ij} = 0$ otherwise x^1_{oi} will be used twice which is not valid. In the previous example (see Figure 4.12) we see that $x^2_{ij} = x^2_{ji} = x^2_{id} = 0.5$ which is cut by the new constraint (4.12).

We identify also a second class of valid inequalities. Indeed, if we consider the polyhedron defined by inequalities (4.4),(4.10),(4.12), this polyhedron is not completely described because some extreme points can be fractional as the one proposed in Figure 4.13.

This figure represents a fractional point because there exist 2 loops linked together. A first loop between three nodes i, j and k and a second loop inside the first, between the two nodes i and j. On the left part of the Figure 4.13 we see the 4 coloured paths on which a quantity of flow equal to 0.5 is sent. This solution satisfies all inequalities (4.4),(4.10) and (4.12). We need to find thus a new class of inequalities to cut these kind of fractional extreme points. We find the following new class of valid inequalities:

$$x^1_{oi} + x^3_{jd} + x^2_{ji} + x^2_{jd} \leq 2, (i,j) \in A | (i \neq j) \text{ and } (i, j \neq o) \text{ and } (i, j \neq d), \tag{4.13}$$

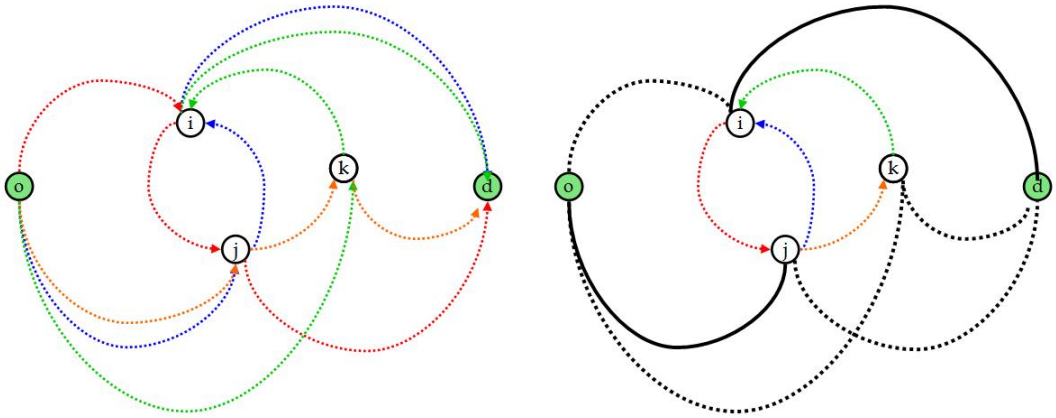


Figure 4.13: Illustration of the second fractional routing case: loop between 3 nodes.

We show now that the two new class of inequalities are valid for the edge-disjoint problem even with negative costs, and when $L = 3$ and $K = 2$.

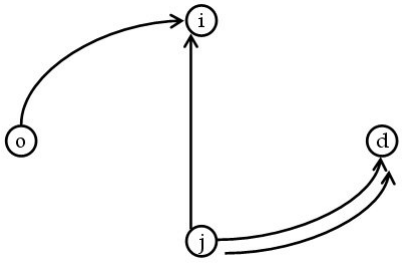


Figure 4.14: Arcs included in the inequality (4.13)

Inequalities (4.12) and (4.13) are both valid when $L = 3$ and $K = 2$. Indeed concerning inequalities (4.12) the use of arc (i, j), (j, i) or (i, d), all the three in position 2 is mutually exclusive and consequently only one of x^2_{ij}, x^2_{ji} or x^2_{id} can be equal to 1. If $x^2_{ij} = 1$, as we are in the edge-disjoint case, $x^2_{ji} = 0$ and $x^1_{oi} = 1$. Consequently, $x^2_{id} = 0$ because it is not possible to have again node i in position 2. If $x^2_{ji} = 1$, then $x^2_{ij} = 0$ and $x^3_{id} = 1$. If $x^3_{id} = 1$, $x^2_{id} = 0$ from inequality (4.10).

Case	x^1_{oi}	x^2_{ji}	x^2_{jd}	x^3_{jd}
(a)	1	1	0	0
(b)	1	0	1	0
(c)	1	0	0	1
(d)	0	1	0	1

Table 4.2: Feasible configurations for inequality (4.13).

Regarding inequality (4.13), from (4.10), it is valid to say that x^2_{jd} and x^3_{jd} can not be both equal to 1 and from (4.12), it is also valid that x^2_{ji} and x^2_{jd} can not be both equal to 1. Then the following cases illustrated on Figure 4.15 are the only feasible configurations.

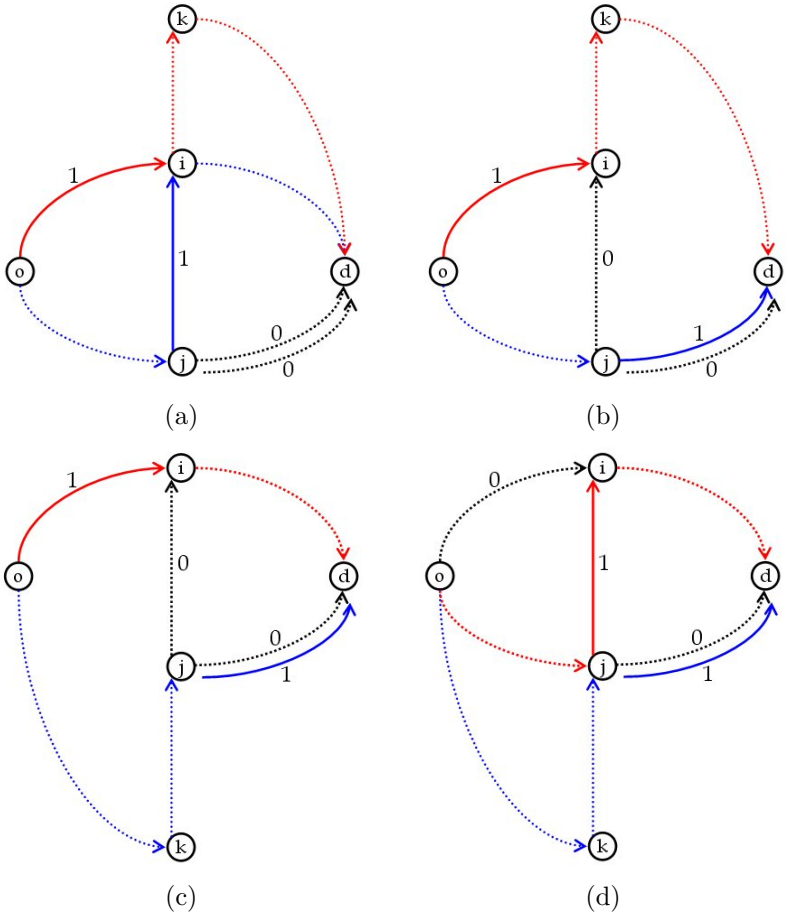


Figure 4.15: Illustration of the feasible configurations for inequality (4.13).

Obviously, when $K = 2$, having more than two of these four variables equal to 1 leads to an infeasible solution. Note that inequality (4.12) is valid for $L = 3$ and any value of K but not valid when $L > 3$. As an example if $L = 4$ we can have one path $(o-j-i-k-d)$ and a second $(o-i-d-d-d)$. This solution is feasible and leads to $x^2_{ji} = x^2_{id} = 1$ which violates new cut (4.12). The same idea can be applied for new cut (4.13). This new class of inequalities is valid for $K = 2$ and any value of L, but becomes invalid when $K > 2$. As an example assume $K = 3$ and $L = 3$, we can obtain a feasible solution using three paths $(o - i - k - d)$, $(o - j - i - d)$ and $(o - m - j - d)$ leading to $x^1_{oi} = x^2_{ji} = x^3_{jd} = 1$ which violates new cut (4.13).

Taking into account the three different new cuts, the linear relaxation of model HopE is equivalent to the following model HopE-LR:

Model HopE-LR

$$
\begin{aligned}
\min \quad & \sum_{ij \in E} \sum_{l \in \{1,\ldots,L\}} c_{ij}\ (x^l_{ij} + x^l_{ji}) & & \\
\text{s.t.} \quad & \sum_{j:(j,i,l-1)\in A'} x^{l-1}_{ji} - \sum_{j:(i,j,l)\in A'} x^l_{ij} = \begin{cases} -K & \text{if } (i=o) \\ K & \text{if } (i=d) \text{ and } (l=L+1), \\ 0 & \text{otherwise} \end{cases} & & \\
(HopE-LR) \quad & & & i \in V', l \in \{1,\ldots,L+1\}, \\
& x^2_{id} + x^3_{id} \le 1, & & i \in V \setminus \{o,d\}, \\
& x^2_{ij} + x^2_{ji} + x^2_{id} \le 1, & & (i,j) \in A | (i \neq j) \text{ and } (i, j \neq o) \text{ and } (i, j \neq d), \\
& x^1_{oi} + x^3_{jd} + x^2_{ji} + x^2_{jd} \le 2, & & (i,j) \in A | (i \neq j) \text{ and } (i, j \neq o) \text{ and } (i, j \neq d), \\
& x^1_{oi} \le 1, & & i \in V \setminus \{o\}, \\
& x^l_{ij} \ge 0, & & (i,j,l) \in A'.
\end{aligned}
$$

Proposition 4. *For $L = 3$ and $K = 2$, and for any cost c_{ij}, model HopE-LR gives a complete description of the associated polyhedron for the edge version problem HopE.*

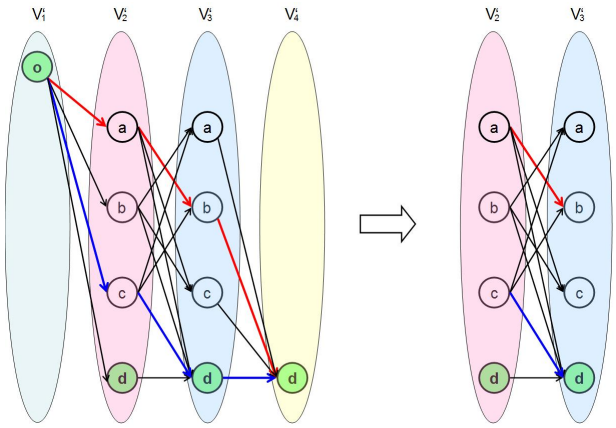


Figure 4.16: Layered graph G' and the reduced layered graph G''

Proof. The proof starts with a first observation: when $L = 3$, looking for two edge-disjoint paths in the full layered graph G' is similar to looking for only two different arcs in a reduced layered graph G''. From a directed full layered graph $G' = (V', A')$, we derive a directed reduced layered graph $G'' = (V'', A'')$ where $V'' = V'_2 \cup V'_3$ with $V'_l = V \setminus \{o\}$, $l = 2, \ldots, L$. Then, the arcs set is defined by $A'' = \{(i_2, j_3) \mid i_2 \in V'_2, j_3 \in V'_3$. This reduced layered graph is composed only of the second arc layer between V'_2 and V'_3 (see for instance Figure 4.16). Indeed if we know that the variable $x^2_{ij} = 1$ because $L = 3$ then automatically we know also which are the first and the last arcs used to form a feasible path between o and d. Here $x^1_{oi} = x^3_{jd} = 1$. As a consequence we can reduce considerably the number of variables of model HopE-LR. Indeed, now variables x^1_{oi} and x^3_{jd} are not yet necessary. We can rewrite model HopE-LR only depending of the variable x in the second arc layer which leads to the alternative model HopE-LR$'$.

Model HopE-LR$'$

$$\min \quad \sum_{ij \in E} c_{ij}^2 \ (x_{ij}^2 + x_{ji}^2) \tag{4.14}$$

$$\sum_{i \in V} \sum_{j \in V \setminus \{o\}} x_{ij}^2 = 2, \tag{4.15}$$

$$x_{id}^2 + \sum_{k \in V \setminus \{d\}} x_{ki}^2 \leq 1, \qquad i \in V \setminus \{o, d\}, \tag{4.16}$$

$$x_{ij}^2 + x_{ji}^2 + x_{jd}^2 \leq 1, \qquad (i,j) \in A, \tag{4.17}$$

$$\sum_{k \in V \setminus \{o\}} x_{ik}^2 + x_{ji}^2 + x_{jd}^2 + \sum_{k \in V \setminus \{o,d\}} x_{kj}^2 \leq 2, \qquad (i,j) \in A \setminus \{(d,d)\}, \tag{4.18}$$

$$\sum_{k \in V \setminus \{o\}} x_{ik}^2 \leq 1, \qquad i \in V \setminus \{o\}, \tag{4.19}$$

$$x_{ij}^2 \geq 0, \qquad (i,j) \in A. \tag{4.20}$$

Model HopE-LR$'$ is equivalent to model HopE-LR as $c_{ij}^2 = c_{oi} + c_{ij} + c_{jd}$. Constraint (4.15) is equivalent to the flow conservation constraints (4.4). Constraints (4.16) are equivalent to the arc disjointness constraints (4.10) as constraints (4.17) are equivalent to constraints (4.12) and finally constraints (4.18) are equivalent to constraints (4.13). Constraints (4.20) are simple non-negativity constraints. If we prove that all the extreme points of model HopE-LR$'$ are integer we prove also that model HopE-LR gives a complete description of the polyhedron of our edge-disjoint problem.

From the reduced layered graph G'' we create a conflict graph $G^c = (V^s, E^s)$ (see for instance Figure 4.17) where all arcs (i,j) of G'' are now considered as nodes ij of G^c. A link between two nodes m and n exists in G^c if and only if the nodes m and n cannot be used together or are incompatible. As the nodes of G^c are the arcs of G'' if a link exists between two nodes of G^c it means that the two corresponding arcs from G'' can not be both used in a solution or are incompatible (for instance arcs (i,j) and (j,i)).

Now, given a graph G, solving the K edge-disjoint L paths problem when $L = 3$ and $K = 2$ is equivalent to finding in G^c two nodes (two arcs in G'') which are compatible or as a consequence, which are not connected. This problem is typically a stable set problem and more precisely a stable set problem of size 2 which has been largely studied in Chapter 3. We know already all the inequalities needed to have the complete description of the polyhedron for the stable set of size 2 problem. We need to prove now that only inequalities of model HopE-LR$'$ are facets of the stable set of size 2 problem.

Figure 4.17 shows the conflict graph G^c of G''. For a given node ba (which represents the arc (b,a) in our original problem), the five extremities of the red links are the five categories of arcs which are incompatible with the arc (b,a). For example, arcs (b,a) and (b,c) are incompatible because the two induced paths are $(o-b-a-d)$ and $(o-b-c-d)$ which use both the arc (o,b) in first position, leading to a solution which is not edge-disjoint.

By isolating the term linked to a given arc (i,j), Constraints (4.18) can be rewritten as:

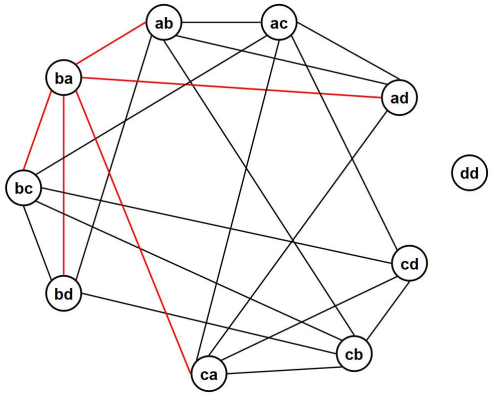


Figure 4.17: Conflict graph G^c.

$$2\ x_{ij}^2 + x_{ji}^2 + x_{jd}^2 + x_{id}^2 + \sum_{k \in V \setminus \{o,j,d\}} x_{ik}^2 + \sum_{k \in V \setminus \{o,i,d\}} x_{kj}^2 \leq 2, \tag{4.21}$$

For a given arc (i, j), this constraint means that among the set of arcs defined as $S_{ij} = \{(i, j), (j, i), (j, d), (i, d), (i, k), (k, j), k \in V \setminus \{o, i, j, d\}\}$, either only the arc (i, j) or at most two other arcs of S_{ij} are in the solution.

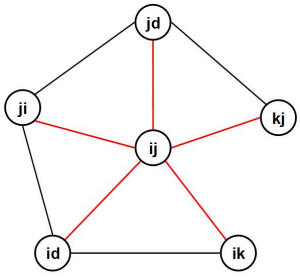


Figure 4.18: Neighbourhood of node ij in the alternative graph G^c.

In Chapter 3 we prove that we need three categories of constraints to completely define the stable set of size 2 problem: there are inequalities (3.7), (3.8) and (3.9). It is obvious that equality (4.15) is equivalent to (3.7) and that inequalities (4.20) are equivalent to inequalities (3.9). We have to prove now that inequalities (4.16), (4.17), (4.18) and (4.19) are the only possible inequalities of the structure (3.8) which appear in our special kind of graph G^c.

a.) Firstly inequalities (4.16), (4.17) and (4.19) are equivalent to (3.8) where P is a maximal clique. As already said, when P is a maximal clique it is obvious that $Q(P) = \emptyset$. We have to prove that for all these three inequalities the clique P is indeed a maximal clique.

i) Concerning inequalities (4.16), for a given node $i \in G$, the clique $P = \{id\} \cup \{ki : k \in V \setminus \{o, i, d\}, i \in V \setminus \{o\}\}$ is a maximal clique. To prove that this clique P is maximal we prove that there does not exist any node mn which could be added to P to form a larger clique. Thus, for all nodes mn the set $\{mn\} \cup P$ is no more a clique. If we assume that $m \neq i$, then we know that $n \neq i$ (because all the nodes of the form $ki \in P$) and all nodes mn are compatible with the node id which belongs to P, then nodes mn and id are not connected and $\{mn\} \cup P$ is not a clique. If $m = i$, then $n \neq d$ and all nodes in are compatible with nodes $ki \in P$ if and only if $k \neq n$. As a consequence the link between nodes mn and ki does not exist and $\{mn\} \cup P$ is not a clique. As a conclusion the clique $P = \{\{id\} \cup \{ki :, k \in V \setminus \{o, i, d\}\}$ is a maximal clique.

ii) Concerning inequalities (4.19), for a given node $i \in G$, the clique $P = \{ik, k \in V \setminus \{o, i\}, i \in V \setminus \{o\}\}$ is a maximal clique. Indeed, if we assume a node mn such that $mn \notin P$ we see that $\{mn\} \cup P$ is not a clique. If $m \neq i$, either $n = i$ and this node mi is compatible with nodes $ik : k \in V \setminus \{o, i, m\}$ or $n \neq i$ and this node mn is compatible with nodes $ik : k \in V \setminus \{o, i, n\}$.

iii) Concerning inequalities (4.17), for a given arc $(i, j) \in G$, the clique $P = \{ij, ji, id\}$ is a maximal clique. Indeed there does not exist any node $mn \notin P$ such that $\{mn\} \cup P$ is a clique. If $m \neq i \neq j$ then either $n = i$ and node mi is compatible with node ij, either $n = j$ and node mj is compatible with node ji or $n \neq i \neq j$ and node mn is compatible with nodes ij and ji. If $m = i$, then $n \neq j \neq d$ and node in is compatible with node ji. Finally if $m = j$, then $n \neq i$ and node jn is compatible with node ij.

b.) Now we consider a clique P which is not a maximal clique and we assume that the node ij belongs to P. We prove that, for all arc $(i, j) \in A$, the clique P composed only of the single vertice $P = \{ij\}$ is the only non-maximal clique which ensures that inequalities (3.8) define a facet. Figure 4.18 shows a node ij and its neighbourhood. To form a clique which contains already the node ij we need to include at least one more node belonging to the neighbourhood of ij.

i) If we consider that $ij \in P$ and $ji \in P$ then we know that $id \notin P$ since $\{ij, ji, id\}$ is a maximal clique and we assume that P is not a maximal clique. For the same reason $jd \notin P$. We know also that $ik \notin P, k \in V \setminus \{o, i, j, d\}$ because ik and ji are compatible and we know also that $kj \notin P, k \in V \setminus \{o, i, j, d\}$ because kj and ji are also compatible. Our clique P is thus restricted to $P = \{ij, ji\}$ and in this case the neighbourhood of P is equal to $Q(P) = \{id, jd\}$. Inequalities (3.8) for P becomes:

$$2x_{ij}^2 + 2x_{ji}^2 + x_{id}^2 + x_{jd}^2 \leq 2 \tag{4.22}$$

This inequality is not a facet as it can be expressed as the linear combination of two inequalities (3.8) where P is considered as a maximal clique. Indeed

(3.8) for the maximal clique $\{ij, ji, id\}$ (respectively for the maximal clique $\{ij, ji, jd\}$) is $x_{ij}^2 + x_{ji}^2 + x_{id}^2 \leq 1$ (respectively $x_{ij}^2 + x_{ji}^2 + x_{jd}^2 \leq 1$ for the maximal clique $\{ij, ji, jd\}$). The sum of these two inequalities leads to inequality (4.22). As a consequence, if $ij \in P$ then $ji \notin P$.

ii) If we consider a clique P which contains the node ij and a subset of the nodes $\{ik, k \in V \setminus \{o, i, j\}\}$, then we know that $ji \notin P$, $jd \notin P$ and $lj \notin P, \forall\ l \in V \setminus \{o, i, j, d\}$ as ji, jd and all lj are compatible with all nodes ik. $P \subset \{ik : k \in V \setminus \{o, i, j\}\}$ and $Q(P) = \{il : l \in V \setminus \{o, i, j\}, il \notin P\}$. $P \cup Q(P)$ forms a maximal clique and the inequality induced by P is dominated by the inequality defined by $P^{'} = P \cup Q(P)$.

iii) If we consider a clique P which contains the node ij and a subset of all the nodes $\{kj, k \in V \setminus \{o, i, j, d\}\}$ then we know that all the nodes $\{il, l \in V \setminus \{o, i, j\}\}$ are not in P since all nodes kj are compatible with all nodes il. We know also that $ji \notin P$ since all nodes kj and the node ji are also compatible. As a consequence, $P \subset \{ij\} \cup \{kj, k \in V \setminus \{o, i, j, d\}\}$ and $Q(P) = \{jd\} \cup \{lj, l \in V \setminus \{o, i, j, d\}, lj \notin P\}$. This clique P is included in a maximal clique $P^{'} = \{jd\} \cup \{kj, k \in V \setminus \{o, j, d\}\}$ thus the inequality (3.8) associated to our clique P is redundant with inequality (4.16) associated to node $j \in G$ and if $ij \in P$ then $kj \notin P, k \in V \setminus \{o, i, j, d\}$.

iv) If we consider a clique P which contains the node ij and the node jd, we know that the node $ji \notin P$ since $\{ij, ji, jd\}$ is a maximal clique and it is assumed that P is not a maximal clique. We know also that all nodes $ik : k \in V \setminus \{o, i, j\}$ are not in P since all these nodes are compatible with the node jd. Only a subset of the nodes $\{kj, k \in V \setminus \{o, i, j, d\}\}$ can be in P because if all the nodes kj are in P then P is a maximal clique. As a consequence $P = \{ij\} \cup \{jd\} \cup \{kj : k \in V \setminus \{o, i, j, d\}\}$ and $Q(P) = \{lj, l \in V \setminus \{o, i, j, d\}, lj \notin P\}$. $P \subset P^{'}$ where $P^{'}$ is the maximal clique $P^{'} = \{jd\} \cup \{kj : k \in V \setminus \{o, j, d\}\}$. The inequality (3.8) associated to our clique P is redundant with inequality (4.16) associated to node $j \in G$ and if $ij \in P$ then $jd \notin P$.

As a consequence, we know that P is a clique which is not maximal and we know also that $ij \in P$. i), ii), iii) and iv) induce that $ji \notin P$, $jd \notin P$, $ik \notin P, k \in V \setminus \{o, i\}$, $lj \notin P, l \in V \setminus \{o, i, j, d\}$. Then for non-maximal clique, inequalities (3.8) are limited to $P = \{ij\}, (i, j) \in A$, ij is the only element in P and (3.8) is equivalent to (4.21).

Our model HopE-LR$^{'}$ gives a complete description of the stable set of size 2 problem for the graph G^c as HopE-LR gives then also a complete description of the 2 edge-disjoint 3-paths problem for the original graph G. □

4.5 Routing problem versus Design problem

In this section we point out the distinction between the routing problem and a closely related problem: the design problem. The difference between these problems is roughly located at the level of the objective function. For the routing problem, as seen before,

the objective is to find really the different routes used to send the K units of flow from the source node o to the destination node d. The objective function is thus equal to the sum on all the edges set of the flow variables x of an edge ij times the routing cost of an edge ij leading to:

$$\sum_{ij \in E} c_{ij} \sum_{l=1}^{L} (x_{ij}^l + x_{ji}^l) \tag{4.23}$$

The associated design problem is the problem for which we want to determine among the potential arcs (or edges) the ones to install in order to guarantee K arc (or edge)-disjoint L paths. Then, new variables, called here design variables, have to be introduced and, as a consequence, the total cost depends on these design variables and no more on the flow variables. For this design problem we can decide to associate new design variables either with each arc or with each edge. For the arc design we add the new variable $y_{ij}, (i,j) \in A$ which is a binary variable taking the value 1 if the arc (i,j) is included in the optimal topology and 0 otherwise. Respectively for the edge design we add the new variable $z_{ij}, ij \in E$ which is also a binary variable taking the value 1 if the edge ij is included in the optimal topology and 0 otherwise. As a consequence we can determine two version of the objective function, for the arc design version:

$$\sum_{ij \in E} c_{ij}(y_{ij} + y_{ji}) \tag{4.24}$$

as for the edge design version:

$$\sum_{ij \in E} c_{ij} z_{ij} \tag{4.25}$$

In this design problem the objective function is equal to the sum of the installation costs of the different edges or arcs which composed the topology. Note that in this study we make the assumption that we have a function $c : E \to \mathbb{R}$ which associates a cost c_{ij} to each edge $ij \in E$. We define $c_{\dots}^R$ as the routing cost and $c_{\dots}^D$ as the design cost. For the two directed arc (i,j) and (j,i) associated to the same edge ij, we assume the following property: $(c_{(i,j)}^R = c_{(j,i)}^R = c_{ij}^R)$ and $(c_{(i,j)}^D = c_{(j,i)}^D = c_{ij}^D)$. We assume also for the design problem and the routing problem that the cost to install an edge (or an arc) is the same that the cost to use this edge (or arc). In other words, for an edge (or an arc), the routing cost of this edge is assumed to be similar to the design cost of this edge, $(c_{(i,j)}^R = c_{(j,i)}^R = c_{ij}^R) = (c_{(i,j)}^D = c_{(j,i)}^D = c_{ij}^D) = c_{ij}$.

For the design problem we see that we have the choice between two alternatives (arc design vs edge design) but we should not forget the second choice concerning the arc or the edge-disjointness. As a summary, for the design problem we can consider four different models, as illustrated on Table 4.3:

	Arc Design	Edge Design
Arc-disjoint case	HopA_y	HopA_z
Edge-disjoint case	HopE_y	HopE_z

Table 4.3: Summary of the different design models.

As said before, in both cases (arc design and edge design) we need to add extra (arc or edge) variables but also constraints relating the new variables. For the arc design case, a new binary variables y_{ij} is needed, $(i,j) \in A$. The new constraints for this case are the following:

$$x_{ij}^l \leq y_{ij}, \quad (i,j) \in A, l \in 1,...,L, \tag{4.26}$$
$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{4.27}$$

This constraint (4.26) can be strengthened leading to:

$$\sum_{l \in 1,...,L} x_{ij}^l \leq y_{ij}, \quad (i,j) \in A, \tag{4.28}$$

For the edge design case, a new binary variable z_{ij} is needed, $ij \in E$. The new constraints for this case are the following:

$$x_{ij}^l \leq z_{ij}, \quad ij \in E, l \in 1,...,L, \tag{4.29}$$
$$x_{ji}^l \leq z_{ij}, \quad ij \in E, l \in 1,...,L, \tag{4.30}$$
$$z_{ij} \in \{0,1\}, \qquad ij \in E. \tag{4.31}$$

These constraints can be strengthened leading to:

$$\sum_{l \in 1,...,L} x_{ij}^l \leq z_{ij}, \quad ij \in E, \tag{4.32}$$
$$\sum_{l \in 1,...,L} x_{ji}^l \leq z_{ij}, \quad ij \in E. \tag{4.33}$$

The final model can be obtained by considering together the different and adequate components. Note however, that in some situations we can combine together the disjoint enforcing constraints with the new constraints defining the extra variables. In fact, such a combination not only removes one set of constraints but even leads to a stronger linear programming relaxation. The four complete design models are the following:

Model HopA$_y$

$$
\begin{aligned}
& \min \quad \sum_{ij \in E} c_{ij}\ (y_{ij} + y_{ji}) \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1)\in A'} x_{ji}^{l-1} - \sum_{j:(i,j,l)\in A'} x_{ij}^{l} \quad = \begin{cases} -K & \text{if} \quad (i = o) \\ K & \text{if} \quad (i = d) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, \\
(HopA_y) \qquad & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i \in V^{'}, l \in \{1, \ldots, L+1\}, \\
& \sum_{l\in\{1,\ldots,L\}} x_{ij}^{l} \leq y_{ij}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i,j) \in A, \\
& x_{ij}^{l} \quad \geq 0 \text{ and integer}, , \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i,j,l) \in A^{'}, \\
& y_{ij} \quad \in \{0,1\}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i,j) \in A.
\end{aligned}
$$

Model HopA$_z$

$$
\begin{aligned}
& \min \quad \sum_{ij \in E} c_{ij}\ z_{ij} \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1)\in A'} x_{ji}^{l-1} - \sum_{j:(i,j,l)\in A'} x_{ij}^{l} \quad = \begin{cases} -K & \text{if} \quad (i = o) \\ K & \text{if} \quad (i = d) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, \\
(HopA_z) \qquad & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i \in V^{'}, l \in \{1, \ldots, L+1\}, \\
& \sum_{l\in\{1,\ldots,L\}} x_{ij}^{l} \leq z_{ij}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ij \in E, \\
& \sum_{l\in\{1,\ldots,L\}} x_{ji}^{l} \leq z_{ij}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ij \in E, \\
& x_{ij}^{l} \quad \geq 0 \text{ and integer}, , \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i,j,l) \in A^{'}, \\
& z_{ij} \quad \in \{0,1\}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ij \in E.
\end{aligned}
$$

Model HopE$_y$

$$
\begin{aligned}
& \min \quad \sum_{ij \in E} c_{ij}\ (y_{ij} + y_{ji}) \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1)\in A'} x_{ji}^{l-1} - \sum_{j:(i,j,l)\in A'} x_{ij}^{l} \quad = \begin{cases} -K & \text{if} \quad (i = o) \\ K & \text{if} \quad (i = d) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, \\
(HopE_y) \qquad & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i \in V^{'}, l \in \{1, \ldots, L+1\}, \\
& \sum_{l\in\{1,\ldots,L\}} (x_{ij}^{l} + x_{ji}^{l}) \leq 1, \qquad\qquad\qquad\qquad\qquad\qquad\qquad ij \in E, \\
& \sum_{l\in\{1,\ldots,L\}} x_{ij}^{l} \leq y_{ij}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i,j) \in A, \\
& x_{ij}^{l} \quad \geq 0 \text{ and integer}, , \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i,j,l) \in A^{'}, \\
& y_{ij} \quad \in \{0,1\}, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (i,j) \in A.
\end{aligned}
$$

Model HopE$_z$

$$
\begin{aligned}
& \min \quad \sum_{ij \in E} c_{ij}\ z_{ij} \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1)\in A'} x_{ji}^{l-1} - \sum_{j:(i,j,l)\in A'} x_{ij}^{l} = \begin{cases} -K & \text{if } (i=o) \\ K & \text{if } (i=d) \text{ and } (l=L+1) \\ 0 & \text{otherwise} \end{cases}, \\
& (HopE_z) \qquad\qquad i \in V^{'}, l \in \{1,\ldots,L+1\}, \\
& \sum_{l\in\{1,\ldots,L\}} (x_{ij}^{l} + x_{ji}^{l}) \le z_{ij}, \qquad ij \in E, \\
& x_{ij}^{l} \quad \ge 0 \text{ and integer}, , \qquad (i,j,l) \in A^{'}, \\
& z_{ij} \quad \in \{0,1\}, \qquad ij \in E.
\end{aligned}
$$

We can now state some properties:

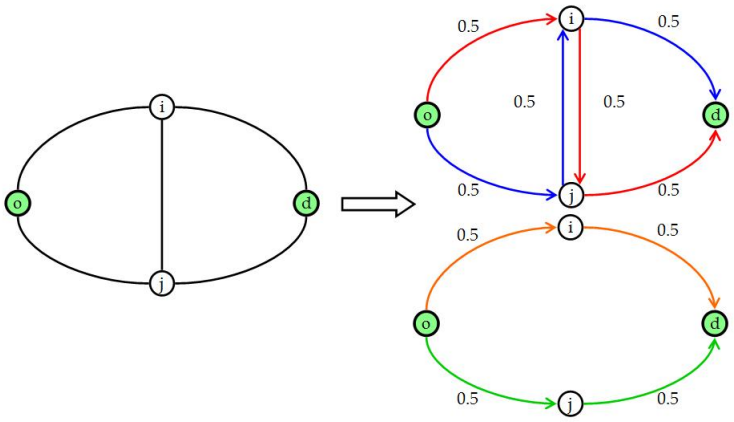


Figure 4.19: Integer design with fractional routing.

Property 3. *Let $\overline{x}$ be a feasible solution of the K edge-disjoint L-paths routing problem and $\overline{z}$ (respectively $\overline{y}$) be a feasible solution of the associated K edge-disjoint (respectively arc-disjoint) L-hops design problem, if $\overline{x}$ is integer then $\overline{z}$ (respectively $\overline{y}$) is also integer. The converse is not always true.*

Indeed, in the previous models routing variables x and design variables z or y are linked by constrains with the common following structure:

$$\sum_{l=1}^{L} x_{ij}^{l} \le z_{ij}(or\ y_{ij}) \tag{4.34}$$

It is clear that if x are integer then obviously z or y are also integer. But if z or y are integer we could have feasible fractional x which satisfy the constraint like it is illustrated on Figure 4.19. We have an integer solution in terms of design (on the left) but with a fractional solution in terms of routing (on the right).

Property 4. *Let $G^{'}(x) = (V, E^{'}(x))$ be a subgraph where $E^{'}(x) \subseteq E$ depends on x and contains only the edges for which the routing variables x are equal to 1, namely $E^{'}(x) = \{ij \in E \text{ s.t. } \exists\ l \in \{1, ..., L\} : x^l_{ij} = 1\}$. Let $z(x)$ be the z edge design variables associated to the routing variables, and let $G^{''}(z) = (V, E^{''}(z))$ be a subgraph where $E^{''}(z) \subseteq E$ depends on z and contains only the edges for which the design variables z are equal to 1, namely $E^{''}(z) = \{ij \in E \text{ s.t. } z_{ij} = 1\}$. If the cost c_{ij} are non-negative, for all optimal routing flow x, we have that $G^{'}(x) = G^{''}(z)$.*

Property 5. *Let $G^{'}(x) = (V, A^{'}(x))$ be a subgraph where $A^{'}(x) \subseteq A$ depends on x and contains only the arcs for which the routing variables x are equal to 1, namely $A^{'}(x) = \{(i, j) \in A \text{ s.t. } \exists\ l \in \{1, ..., L\} : x^l_{ij} = 1\}$. Let $y(x)$ be the y arc design variables associated to the routing variables, and let $G^{''}(y) = (V, A^{''}(y))$ be a subgraph where $A^{''}(y) \subseteq A$ depends on y and contains only the arcs for which the arc design variables y are equal to 1, namely $A^{''}(y) = \{(i, j) \in A \text{ s.t. } y_{ij} = 1\}$. If the cost c_{ij} are non-negative, for all optimal routing flow x, we have that $G^{'}(x) = G^{''}(y)$.*

If the costs are non-negative, then, due to inequalities like (4.34), the design is only composed of the set of edges (or arcs) used for the routing problem. On the opposite and as it is illustrated on Figure 4.20, with negative costs, it is possible to add an edge (or an arc) (like the blue edge ij) which has a negative cost in the optimal design even if this edge (or arc) is not used in the routing problem simply because the negative cost of edge ij (or arc (i, j)) decrease the value of the objective function of the design problem but has no real impact on the objective function of the routing problem.

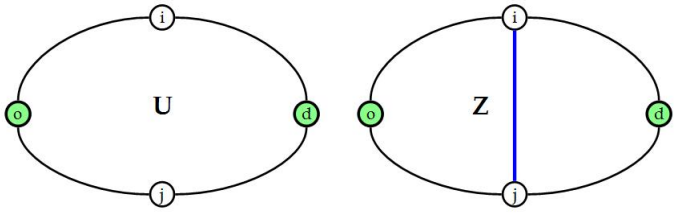


Figure 4.20: Integer edge-disjoint routing $\neq$ integer edge-disjoint design with $c_{ij} \leq 0$.

We can now extend the two propositions (2) and (3) developed in the previous section to our four design models:

Property 6. *For $L = 3$ and $K \geq 1$, the polyhedron defined by the constraints of the linear relaxation of model $HopA_y$ is integral.*

Proof. The proof is trivial. Indeed due to Proposition 2 we know that the routing variables x are always integer. In model HopA_y, x and y variables are linked by inequalities (4.28). If we look at the matrix of constraints, we have the same matrix than for model HopA with an identity submatrix for variables y. Then if x are integer, automatically y are also integer and all the extreme points are integer which is a sufficient condition for a model to describe completely the polyhedron. □

Property 7. *For $L = 3$ and $K \geq 1$, the polyhedron defined by the constraints of the linear relaxation of model $HopA_z$ is integral.*

Proof. The proof of this property is similar to the previous as both matrix HopA and HopA_z are similar. Indeed the flow conservation constraints are strictly the same for both matrix and concerning the new cuts (4.10), we have one new cut for each arc (i, d), but not for the reverse arc (d, i) which does not exists in the layered graph. As a consequence, the number and the nature of additional new cuts (4.10) is the same that for model HopA and by the way the matrix of HopA_z is also totally unimodular which ends the proof. □

We have to point out that if for each edge ij the design cost is equal to the routing cost then the value of an optimal solution of the routing problem HopA is equivalent to the value of the associated optimal solution of the arc design problem HopA_y but different compare to the value of the associated optimal solution of the edge design problem HopA_z. Indeed, in HopA_y if an edge ij is used in both directions we count two times the cost of this edge as in HopA_z we count only one time the cost of this edge.

Property 8. *For $L = 3$ and $K = 2$, the polyhedron defined by the constraints of the linear relaxation of model HopE_y (respectively HopE_z) in which the edge-disjoint constraints are replaced by the inequalities (4.10),(4.12) and (4.13) is integral.*

Here again the proof is trivial. Indeed Proposition 4 ensures that the linear relaxation of the edge-disjoint model is always integer in terms of x routing variables. It follows directly that y variables (respectively z variables) are also integer taking into account the structure of the inequalities linking the routing and the design variables. Note that here the value of all the optimal solutions for the three models HopE, HopE_y and HopE_z are always equal.

4.6 Extension to $L \geq 4$ cases

In the previous section we proved that our formulations gives a complete description of the associated polyhedron when $L = 3$ for different versions of the main problem. A natural extension is to look at the cases where $L \geq 4$. First it is important to note that inequalities (4.7) or (4.8) stay valid for all L and for all K. Remember that Proposition 1 ensures, for the case where $L = 4$ and $K = 1$, that the linear relaxation of model HopA and HopE is always integer. Then, when we analyze the extension where $L = 4$ and $K = 2$, for the simplest case (the arc-disjoint case) with positive cost, we find counter-examples in which the optimal solution of the linear relaxation is fractional. One of these counter-examples is illustrated on Figures 4.21 and 4.22.

The optimal solution of the linear relaxation model consists in sending a flow equal to 0.5 on each path (P_1, P_2, P_3, P_4), leading to an optimal LP cost of 37. Note that this solution is not feasible for the arc-disjoint case because some arcs (like for example the arc (b, c) in red) are used in different paths (arc (b, c) is used in position 3 in P_1 and in position 2 in P_2). The optimal solution of the IP consists in sending a flow of 1 on each path (P_3, P_4), leading to an optimal IP cost of 38. In this case, the optimal solution of the linear relaxation of our model is not always integer.

As a natural extension we could try to find new inequalities which cut this fractional structure when $L \geq 4$. But we first prefer to realize some numerical tests in order to see if these cuts are necessary or not. The main idea of this testing phase is to evaluate

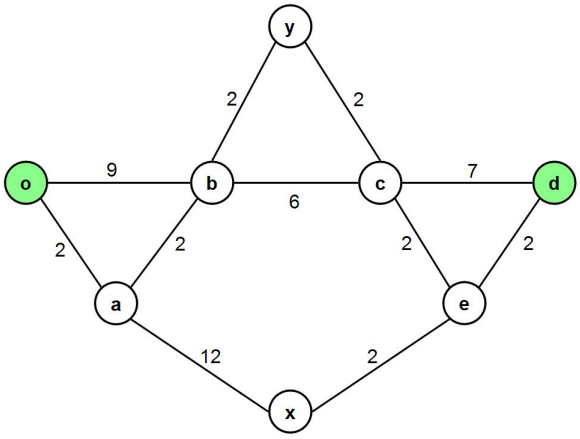


Figure 4.21: Fractional counter example when $L = 4$.

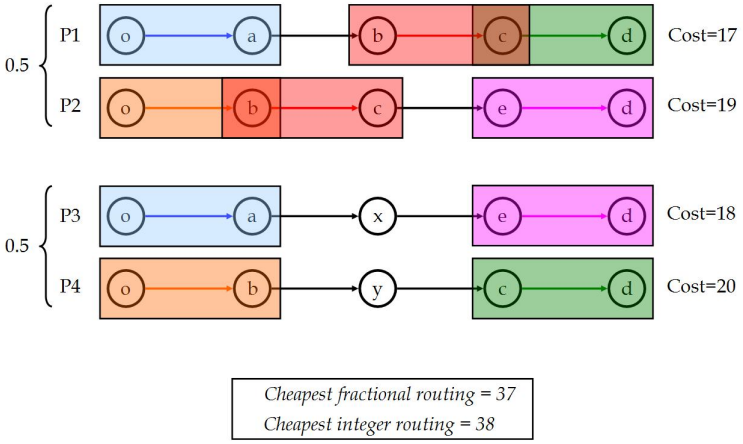


Figure 4.22: Different fractional paths when $L = 4$.

the gap between the optimal value of the linear relaxation and the integer optimal value of model Hop. We organize the tests as follows: we carried out series of tests (approximately 29.000 tests) on model HopA for different values of $L \in [4, 6]$, and different values of $K \in [2, 4]$, with randomly generated cost $c_{ij} \in$ different intervals ($[1 - 10]$ or $[1 - 100]$ or $[1 - 1000]$). These tests were done using complete network (by complete network we understand that there exists a potential edge between each pairs of nodes) and were carried out on a Pentium 3, 1.1 Ghz, 1 Gb Ram, using CPLEX 7.1. with language AMPL.

As shown on Table 4.4, out of roughly 29.000 tests, we find only 6 counter-examples (around 0.02 % of the set of tests) in which the optimal solution of the linear relaxation of model HopA is fractional and different from the optimal integer solution of model HopA. Moreover the quality of the lower bound is excellent (between [0%-0.8%]) as illustrated in the last column of Table 4.4. It means that the linear relaxation of model Hop

is very good and that, from an application point of view, the gap remaining to close is so small that the motivation to find new valid inequalities cutting the fractional points is low.

Instance	**L**	**K**	**Cost**	LP^*	IP^*	**Gap(%)**
1	4	3	[1-1000]	583	585	0.34
2	4	4	[1-100]	60.5	61	0.82
3	5	3	[1-1000]	697	700	0.45
4	5	4	[1-100]	127.5	128	0.39
5	6	4	[1-10]	29	29	0.00
6	6	3	[1-100]	132.5	133	0.38

Table 4.4: Fractional optimal solutions when $L > 3$.

4.7 Summary

In this chapter we formulate the problem of finding between two node o and d, K edge (or arc)-disjoint paths with a maximum of L hops in an extended space. We show how suitable is the extended formulation Hop to model different versions of the problem leading to valid formulations for any value of L and K, which is not the case of the existing formulations expressed in the space of the design variables. We prove that we have the complete description of the associated polyhedron for different versions of the problem when $L = 3$ and we formulate and prove some nice proposition and properties linked to this Hop model for $L = 3$. We conclude by a numerical study about the quality of the linear relaxation of model Hop for $L > 3$ which shows that the gap remaining to close between the optimal solution of the linear relaxation and the optimal solution of the integer problem is microscopic. Table 4.5 summarizes the theoretical and numerical contributions of this chapter.

		Arc-Disjoint		**Edge-Disjoint**	
		Costs Conditions	**Polyhedron**	**Costs Conditions**	**Polyhedron**
$L = 3$	$K = 1$	-	Complete Description	-	Complete Description
$L = 3$	$K = 2$	-	Complete Description	-	Complete Description
$L = 3$	$K \geq 3$	-	Complete Description	$c_{ij} \geq 0$	LP^* is Integer
$L = 4$	$K = 1$	$c_{ij} \geq 0$	LP^* is Integer	$c_{ij} \geq 0$	LP^* is Integer
$L = 4$	$K \geq 2$	-	Good Quality of the LB	-	Good Quality of the LB
$L = 5$	$K = 1$	$c_{ij} \geq 0$	LP^* is Integer	$c_{ij} \geq 0$	LP^* is Integer
$L = 5$	$K \geq 2$	-	Good Quality of the LB	-	Good Quality of the LB

Table 4.5: Different cases and associated contributions of this chapter.

Chapter 5

Benders decomposition for the hop-constrained survivable network design problem

5.1 Introduction

In this chapter we extend the work done in Chapter 4 to the multicommodity case. Given a graph with nonnegative edge weights and a set of pairs of nodes Q, we study the problem of constructing a minimum weight set of edges so that the induced subgraph contains at least K edge-disjoint paths containing at most L edges between each pair of terminals in Q. This problem merges two major streams of the network design literature, indeed it focuses on the survivability constraints as well as on the quality of service constraints.

On one hand, the K edge-survivable network design problem, denoted by $K-ESNDP$, consists in finding a K edge-survivable subgraph of G with minimum total cost, where the cost of a subgraph is the sum of the costs of its edges. A polyhedral study of the problem for $K = 2$ can be found in [100]. Similarly, the node-survivable network design problem, denoted by $NSNDP$, consists in finding a minimum cost node-survivable subgraph of G, see [79] among others. A variant of this problem only requires edge (or node) disjointness between pairs of node in a subset $S \subset V$. Note that a related network design problem with node-disjoint paths between some pairs of nodes which also incorporates hop-constraints in a disguised fashion is the bounded ring network design problem studied by Huygens *et al.* [41, 42, 43], among others. For more references about these problems, see the survey by Kerivin and Mahjoub [68].

On the other hand, the L-hop constrained network design problem consists in finding a least cost subgraph of G such that there exists a L-path between every pair of nodes, or the nodes in a subset S. This problem was studied by Balakrishnan and Altinkemer in [10] as a means of generating alternative base solutions for a network design problem. They presented a standard network flow formulation with an additional cardinality con-

straint for each commodity (to model the hop constraints). They have also derived a Lagrangean relaxation based method whose theoretical best bound improves the linear programming bound given by the previous formulation. Later on, in the context of a directed spanning tree problem (which can be seen as a special case of a single-source L-hop-constrained network design problem), Gouveia in [52] presented a layered network flow reformulation whose linear programming bound equals a lagrangean based bound similar to the one proposed by Balakrishnan and Altinkemer. Then this reformulation has been used in several network design problems with hop constraints (e.g, [90], [53] and [55]) and even some hop-constrained problems involving survivability considerations (more on this below). It is interesting to point out that the case with $L = 2$ already contains lots of structure. In fact Dahl *et al.* in [31] make a computational study of this variation of the problem.

Relevant for obtaining the good computational results for the *K-ESNDP*, is the fact that most of the best methods rely on so-called natural models, that is, models that use only one variable for each edge of the graph and an exponential sized set of constraints. However, for many of these inequalities the associated separation problem is well solved and thus, they can be efficiently separated leading to quite good cutting plane algorithms as for instance in the paper of Dalh and Stoer [33]. Unfortunately, finding a similar approach for the same type of problem with hop-constraints is much more complicated. The reason for this is that it is not straightforward to obtain a valid natural formulation for a variant of the single commodity case. This variant consists in finding a set of edges containing K edge-disjoint L-paths between the two given nodes. Itaí *et al.* in [65] and later Bley in [18] study the complexity of this problem for the node-disjoint and the edge-disjoint cases.

For $K = 1$, Dahl in [27] has provided such a formulation and shown that it describes the corresponding convex hull for $L \leq 3$. Later on, Dahl *et al.* in [28] have shown that finding such a description for $L \geq 4$ would be quite more complicated. For $K \geq 2$ the results are even worse. Huygens *et al.* in [63] have extended Dahl's result for $K = 2$ and $L \leq 3$. For $L \geq 4$, the only interesting result for the moment is the one given by Huygens *et al.* in [62] for $L = 4$ and $K = 2$ where a valid formulation has been given. However, nothing else is known which explains that the only cutting plane method for the more general problem with several sources and several destinations, (see Huygens *et al.* in [61]) only consider $L \leq 3$. Based on this, it makes sense to look for alternative ways of formulating the problem, and the layered approach described previously appears to be a good candidate for formulating the problem since it is easily generalized for the case with K disjoint paths. In fact, a similar approach has already been used for the version of the problem with node-disjoint paths (see [56] and [57] for a more complicated version) and the results in these papers (although for a slightly more complicated variant) give a sort of motivation for the method developed and tested in this chapter:

(i) the linear programming bound given by the model (if it can be solved) is often be reasonably good;

(ii) however, the model is difficult to use in a straightforward way with CPLEX because it has too many variables.

Thus, (i) and (ii) motivate the approach of using this kind of model but using a decomposition algorithm, such as the Benders decomposition. Furthermore, another outcome of this research is that the Benders cuts might give some relevant information for finding valid inequalities for the cases with $L \geq 5$.

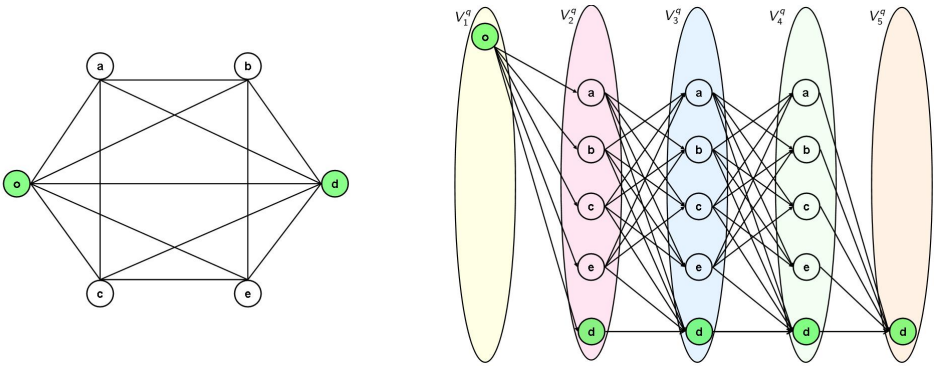
In this chapter, we formulate the network design problem for any $L, K \geq 1$ with multiple pairs of terminals $(o(q), d(q))$, for $q \in Q$, as an integer program based on the layered representation from Gouveia in [52]. Up to our knowledge, this is the first formulation for the problem valid for $L \geq 5$. As previously mentioned the model is too large to be used directly within CPLEX. Hence, we use a Benders decomposition method to efficiently handle the large number of variables and constraints. Although Benders decomposition has been widely used for hard mixed-integer problems — including fixed-charge network design problems [24] —, not much is said about the algorithmic aspect, most authors using "textbook implementations". Some recent works [40, 76] have highlighted the importance of the normalization constraint in the separation problem. Herein, we investigate another aspect of the algorithm, namely, when to generate cuts. We present a thorough computational study of various cutting plane and branch-and-cut algorithms on a large set of instances including the real based instances from *SNDlib* [89]. We confirm previous results obtained by Fortz and Poss in [44] showing that branch-and-cut algorithms outperform cutting plane algorithms. We also show that our Benders cuts contain the constraints used by Huygens *et al.* in [63] and [62] to formulate the problem for $L = 2, 3, 4$ in the space of natural design variables, as well as new valid inequalities when $L \geq 5$. Hence, for $L = 2, 3$ our branch-and-cut algorithms separate at the same time (and polynomially) "cuts inequalities" and "L-paths inequalities" while the branch-and-cut algorithm from Huygens *et al.* in [61] needs to separate both independently. Finally, we present a fast and efficient LP-based heuristic that provides the optimal solution for more than half of the instances.

In the next section we introduce the layered representation and describe our integer formulation. In Section 5.3, we reformulate the problem through Benders decomposition and discuss different algorithmical approaches. Section 5.4 compares the Benders cuts with previous known cuts for the problem, while computational results are presented in Section 5.5.

5.2 Problem description

As mentioned before, the main idea of [52] is to model the subproblem associated with each commodity with a directed graph composed of $(L+1)$ layers as illustrated in Figure 5.1.

Namely, from the original non-directed graph $G = (V, E)$, we create a directed layered graph $G^q = (V^q, A^q)$ for each commodity, where $V^q = V_1^q \cup \ldots \cup V_{L+1}^q$ with $V_1^q = \{o(q)\}$, $V_{L+1}^q = \{d(q)\}$ and $V_l^q = V \backslash \{o(q)\}$, $l = 2, \ldots, L$. Let v_l^q be the copy of $v \in V$ in the $l-th$ layer of graph G^q. Then, the arcs sets are defined by $A^q = \{(i_l^q, j_{l+1}^q) \mid i_l^q \in V_l^q, j_{l+1}^q \in V_{l+1}^q, l \in \{1, \ldots, L\}\}$, see Figure 5.1. In the sequel, an (undirected) edge in E with endpoints i and j is denoted ij while a (directed) arc between $i_l^q \in V_l^q$ and $j_{l+1}^q \in V_{l+1}^q$ is denoted by (i, j, l) (the commodity q is omitted in the notation as it is often clear from



(a) (b)

Figure 5.1: Basic network (a) and its layered representation (b) when $L = 4$

the context).

Note that each path between $o(q)$ and $d(q)$ in the layered graph G^q is composed of exactly L arcs (hops), which corresponds to a maximum of L edges (hops) in the original one. In fact this is the main idea of this transformation, since in the layered graph any path is feasible with respect to the hop constraints. The usual network flow equations defined in this layered graph yield the following model:

$$
\begin{aligned}
& \min \quad \sum_{ij \in E} c_{ij} z_{ij} \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1) \in A^q} x_{ji}^{l-1q} - \sum_{j:(i,j,l) \in A^q} x_{ij}^{lq} = \begin{cases} -K & \text{if} \quad (i = o(q)) \\ K & \text{if} \quad (i = d(q)) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, \\
(P) \qquad & \qquad\qquad i \in V^q, l \in \{1, \ldots, L+1\}, q \in Q, \quad (5.1) \\
& \sum_{l \in \{1,\ldots,L\}} \left(x_{ij}^{lq} + x_{ji}^{lq} \right) \leq z_{ij}, \qquad ij \in E, q \in Q, \quad (5.2) \\
& z_{ij} \in \{0,1\}, \qquad ij \in E, \quad (5.3) \\
& x_{ij}^{lq} \quad \text{integer}, \qquad (i,j,l) \in A^q, q \in Q. \quad (5.4)
\end{aligned}
$$

Each variable z_{ij} states whether edge $ij \in E$ is selected and each variable x_{ij}^{lq} describes the amount of flow through arc (i, j, l) for commodity q in layered graph G^q. Constraints (5.1) are the flow conservation constraints at every node of the layered graph which guarantee that K units of flow go from $o(q)$ to $d(q)$, while constraints (5.2) guarantee edge-disjointness of the paths. Note that (5.2) together with (5.3) imply that $x_{ij}^{lq} \leq 1$ for $i \neq j$, while (5.1) implies that $x_{dd}^{lq} \leq K$.

In the sequel, we assume the reader familiar with standard notions of polyhedral theory, see for instance [87].

5.3 Benders decomposition

5.3.1 Reformulation

When facing a complex mixed-integer optimization problem, the Benders decomposition method [15] can be used to project out complicating real variables. This projection results in the addition of many constraints to the problem. Benders decomposition has been widely studied for fixed charge network design problems, see Costa in [24] and Costa, Cordeau and Gendron in [25]. Indeed, these problems usually route multi-commodity flows on some network to be designed. Therefore, the associated formulations contain many constraints and variables bound together by the capacity constraints. Then, once we project out the flow variables, the subproblems become independent linear programs for each commodity (see, for instance, $SP(q, \overline{z})$ below), thus reducing significantly the size of the linear programs to solve. However, the classical framework does not apply to our model *(P)* because all of its variables are integer; classical duality theory does not allow us to project out variables with integer restrictions. It is well known indeed in the field of stochastic programming that integer recourse cannot be tackled through classical Benders decomposition, called L-shaped in stochastic programming [17]. Although Carøe and Tind in [20] generalize the L-shape to integer recourse using general duality theory, their framework stays mainly theoretical.

To avoid this difficulty, we introduce a new formulation for the problem, (P'), where we relax the integrality restrictions on x variables in (P), replacing (5.4) by

$$x_{ij}^{lq} \geq 0, \qquad (i,j,l) \in A^q, q \in Q. \tag{5.5}$$

We discuss below and in Section 5.4 whether (P') provides the same optimal design $\overline{z}$ as (P). Then, we can use the classical framework (see [24] among others) to project out x variables from (P'), resulting in the Benders reformulation

$$(BR) \qquad \begin{array}{lll} \min & \displaystyle\sum_{ij \in E} c_{ij} z_{ij} & \\ \text{s.t.} & \displaystyle K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E} z_{ij}\overline{\sigma}_{ij} \leq 0, & (\overline{\pi}, \overline{\sigma}) \in \displaystyle\bigcup_{q \in Q} \mathcal{R}^q, \\ & z_{ij} \in \{0,1\}, & ij \in E, \end{array}$$

where $\mathcal{R}^q$ contains the extreme points of the feasibility polyhedron for the dual subproblem $SP(q, \overline{z})$ described next. Given commodity $q \in Q$, let us introduce a dual variable π_i^l, associated with node $i \in V$ and layer l, for each constraint (5.1) and a dual variable σ_{ij} for each constraint (5.2). Defining $o := o(q)$ and $d := d(q)$, and adding the constraint $\pi_d^{L+1} \leq 1$ to normalize the dual cone (see [40, 76] for alternative choices of normalization constraints), we get our dual subproblem $SP(q, \overline{z})$

$$
\begin{array}{lll}
\max & K\pi_d^{L+1} - \sum_{ij\in E} \overline{z}_{ij}\sigma_{ij} & \\
\text{s.t.} & \pi_i^2 - \pi_o^1 - \sigma_{oi} \le 0, & i \in V\backslash\{o\}, \\
& \pi_i^{l+1} - \pi_j^l - \sigma_{ij} \le 0, & i,j \in V\backslash\{o,d\}, i<j, l \in \{2,\dots,L\}, \\
SP(q,\overline{z}) & \pi_j^{l+1} - \pi_i^l - \sigma_{ij} \le 0, & i,j \in V\backslash\{o,d\}, i<j, l \in \{2,\dots,L\}, \\
& \pi_d^{l+1} - \pi_i^l - \sigma_{id} \le 0, & i \in V\backslash\{o,d\}, l \in \{2,\dots,L\}, \\
& \pi_d^{l+1} - \pi_d^l \le 0, & l \in \{2,\dots,L\}, \\
& \pi_d^{L+1} \le 1, & \\
& \sigma_{ij} \ge 0, & ij \in E.
\end{array} \tag{5.6}
$$

Note that for each commodity $q \in Q$, one of the constraints in (5.1) is redundant, which can be represented by setting $\pi_o^1 = 0$.

Next, we discuss how to extend this procedure to (P). We need first to introduce some notations. Given $\overline{z} \in \{0,1\}^{|E|}$, let $\mathcal{X}_i(\overline{z})$ be the set of binary vectors defined by (5.1),(5.2) and (5.4) and $\mathcal{X}_c(\overline{z})$ the polyhedron defined by (5.1),(5.2) and (5.5). Then, define $\mathcal{Z}_i$ (respectively $\mathcal{Z}_c$) as the set of vectors $\overline{z} \in \{0,1\}^{|E|}$ such that $\mathcal{X}_i(\overline{z})$ (respectively $\mathcal{X}_c(\overline{z})$) is nonempty, and let $conv(\mathcal{Z}_i)$ (respectively $conv(\mathcal{Z}_c)$) be its convex hull. Since $\mathcal{X}_i(\overline{z}) \subseteq \mathcal{X}_c(\overline{z})$ for every binary $\overline{z}$, we have that $\mathcal{Z}_i \subseteq \mathcal{Z}_c$ so that any valid inequality for $conv(\mathcal{Z}_c)$ is valid for $conv(\mathcal{Z}_i)$. In particular, the Benders cut

$$
K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij\in E} z_{ij}\overline{\sigma}_{ij} \le 0, \tag{5.7}
$$

with $(\overline{\pi},\overline{\sigma}) \in \mathcal{R}^q$ for some commodity $q \in Q$, is valid for $conv(\mathcal{Z}_i)$.

These definitions raise the following question. Are cuts (5.7) together with integrality restrictions (5.3) enough to characterize $\mathcal{Z}_i$? Namely, given a binary vector $\overline{z}$, is it true that $\overline{z}$ belongs to $\mathcal{Z}_i$ if and only if $\overline{z}$ does not violate any cut (5.7)? It is easy to see that this is true when $K = 1$, and we prove in Section 5.4 that this is also the case for $L = 2, 3$ with any $K \ge 2$, and for $L = 4$ with $K = 2$. However, since we do not know the answer for general K and L, we must in general solve the following feasibility problem

$$
\begin{array}{lll}
\min & e & \\
\text{s.t.} & \sum_{j:(j,i,l-1)\in A^q} x_{ji}^{l-1q} - \sum_{j:(i,j,l)\in A^q} x_{ij}^{lq} = \begin{cases} -K+e & \text{if } (i=o(q)) \\ K-e & \text{if } (i=d(q)) \text{ and } (l=L+1) \\ 0 & \text{otherwise} \end{cases}, & \\
FP(q,\overline{z}) & & i \in V^q, l \in \{1,\dots,L+1\}, q \in Q, \\
& \sum_{l\in\{1,\dots,L\}} \left(x_{ij}^{lq} + x_{ji}^{lq}\right) \le \overline{z}_{ij}, & ij \in E, q \in Q, \\
& x_{ij}^{lq} \quad \text{integer}, & (i,j,l) \in A^q, q \in Q \\
& e \ge 0 &
\end{array}
$$

If $e = 0$, $\overline{z} \in \mathcal{Z}_i$. Otherwise, we must add the weak inequality

$$
\sum_{ij\in E^0(\overline{z})} z_{ij} \ge 1, \tag{5.8}
$$

with $E^0(\overline{z}) = \{ij \in E \text{ s.t. } \overline{z}_{ij} = 0\}$, to move away from the current solution, as explained in the next subsection. It is interesting to point out that in our computational experiments, we never needed to add such a cut, see Section 5.5. As a consequence, we did not find a vector $\overline{z} \in \mathcal{Z}_c \backslash \mathcal{Z}_i$.

5.3.2 Algorithmic approach

(BR) contains an exponential number of constraints while only a few of them are active at the optimum. Therefore, we must dynamically generate the required constraints throughout the solution method. First works on Benders decomposition for mixed-integer problems use cutting plane algorithms, cycling many times between master integer problems and continuous subproblems. However, modern developments in branch-and-cut frameworks such as the commercial CPLEX [64] or the noncommercial SCIP [1], among others, have eased the development of a branch-and-cut algorithm to solve the master problem, incorporating the Benders cut separation in the cutting plane callback. Recent works by Fortz and Poss in [44] and by Bai and Rubin in [8] present examples for which there is an important time reduction when using a branch-and-cut algorithm instead of a cutting plane algorithm. First we briefly describe the multi-cut cutting plane algorithm, while we detail later our different branch-and-cut algorithms.

Cutting plane approach

Algorithm 1: "Naive" Benders decomposition algorithm: `cp`

```
repeat
    solve (MP);                                              /* solve an IP */
    let z̄ be an optimal solution;
    foreach q ∈ Q do
        compute s_q = SP(q, z̄);                  /* solve the dual subproblem */
        if s_q > 0 then add (5.7) to (MP);
    if s_q ≤ 0 for each q ∈ Q then /* are all dual suproblems feasible? */
        foreach q ∈ Q do
            compute f_q = FP(q, z̄);   /* solve the feasibility subproblem */
        if f_q > 0 for some q ∈ Q then add (5.8) to (MP);
until s_q, f_q ≤ 0 for each q ∈ Q;
return z̄
```

Given a subset R^q of $\mathcal{R}^q$ for each $q \in Q$, and binary vectors $\overline{z}^s$, $s = 1, \ldots, r$, let us define the master problem

$$
(MP) \qquad \begin{aligned}
\min \quad & \sum_{ij \in E} c_{ij} z_{ij} \\
\text{s.t.} \quad & K\overline{\pi}^{L+1}_{d(q)} - \sum_{ij \in E} z_{ij} \overline{\sigma}_{ij} \leq 0, && (\overline{\pi}, \overline{\sigma}) \in \bigcup_{q \in Q} R^q, \\
& \sum_{ij \in E^0(\overline{z}^s)} z_{ij} \geq 1, && s = 1, \ldots, r, \\
& z_{ij} \in \{0, 1\}, && ij \in E.
\end{aligned}
$$

Algorithm 2: Improved Benders decomposition algorithm: `cp-i`

```
repeat
    solve the linear programming relaxation of (MP);          /* solve a LP */
    let z̄ be an optimal solution;
    foreach q ∈ Q do
        compute s_q = SP(q, z̄);
        if s_q > 0 then add (5.7) to (MP);
until s_q ≤ 0 for each q ∈ Q;
repeat
    solve (MP);                                               /* solve an IP */
    let z̄ be an optimal solution;
    foreach q ∈ Q do
        compute s_q = SP(q, z̄);
        if s_q > 0 then add (5.7) to (MP);
    if s_q ≤ 0 for each q ∈ Q then
        foreach q ∈ Q do compute f_q = FP(q, z̄);
        if f_q > 0 for some q ∈ Q then add (5.8) to (MP);
until s_q, f_q ≤ 0 for each q ∈ Q;
return z̄
```

In algorithms 1 and 2, we describe our cutting plane algorithms, `cp` and `cp-i`. Because the main computational burden is the solution of (MP), we implemented a multi-cut version of the algorithm: we solve the subproblem for each commodity, therefore adding up to $|Q|$ cuts per iteration. The improved version `cp-i` starts by solving the linear programming relaxation of (MP) in a cutting plane fashion. Various works enhance this classical solution algorithm. Among them, Magnanti and Wong in [81] study the effect of using special cuts called "pareto optimal", Tsamasphyrou *et al.* in [101] describe a more subtle version of the multi-cut algorithm, grouping together subsets of subproblems, and Rei *et al.* in [95] use local branching to accelerate the overall algorithm. Sometimes, strong classes of cuts are known for the problem allowing to add even more cuts at each iteration (see Gabrel *et al.* in [46]). Nevertheless we show herein that, at least for our problem, branch-and-cut algorithms are an order of magnitude faster than cutting plane algorithms.

Branch-and-cut approach

An alternative strategy is to solve (MP) only once. We aim at embedding the generation of violated feasibility cuts (5.7) (and (5.8) if needed) into the branch-and-cut framework solving (MP).

It is important to add many cuts early in the tree to avoid exploration of too many infeasible nodes. However, adding too many unnecessary cuts would slow down the linear programming relaxation at each node. Our first branch-and-cut algorithm, `bc-all`, checks for violated Benders cuts (5.7) at every node of the tree, while it tests for violated inequality (5.8) only at integer nodes. As noted by Fortz and Poss in [44], this algorithm is relatively slow, because too many cuts are added and too much time is spent in the solution of $SP(q, \overline{z})$. In `bc-int`, we check for cuts (5.7) and (5.8) only at

integer nodes. Finally, we developed an hybrid algorithm `bc-n`, described in algorithm 3, checking for violated inequality (5.8) at integer nodes and for violated inequality (5.7) at integer nodes and nodes with a depth less than or equal to n. Note that `bc-n` generalizes both frameworks since `bc-int` is the same as `bc-0`, and `bc-all` is the same as `bc-`$|E|$.

In algorithm 3, solving a node $o' \in T$ means solving the linear programming relaxation of (MP), augmented with branching constraints of o', while depth(o') counts the number of branching constraints of o'.

Algorithm 3: Hybrid branch-and-cut algorithm: `bc-n`

begin `/* Initialization */`
  $T = \{o\}$ where o has no branching constraints;
  $UB = +\infty$;
while T *is nonempty* **do**
  select a node $o' \in T$;
  $T \leftarrow T \backslash \{o'\}$; `/* withdraw node` o' `from the tree */`
  solve o';
  let $\overline{z}$ be an optimal solution;
  let $\overline{w}$ be the optimal cost;
  if $\overline{w} < UB$ **then**
    if $\overline{z} \in \{0,1\}^{|E|}$ ***or*** *depth*$(o') \leq n$ **then**
      foreach $q \in Q$ **do** compute $s_q = SP(q, \overline{z})$;
      if $s_q > 0$ **then** add (5.7) to (MP);
    if $\overline{z} \in \{0,1\}^{|E|}$ ***and*** $s_q \leq 0$ *for each* $q \in Q$ **then**
      foreach $q \in Q$ **do** compute $f_q = FP(q, \overline{z})$;
      if $f_q > 0$ *for some* $q \in Q$ **then** add (5.8) to (MP);
      else
        $UB \leftarrow \overline{w}$; `/* define a new upper bound */`
        $z^* \leftarrow \overline{z}$; `/* save current incumbent */`
    if $s_q > 0$ ***or*** $f_q > 0$ *for some* $q \in Q$ **then**
      $T \leftarrow T \cup \{o'\}$; `/* put node` o' `back in the tree */`
    else if $\overline{z} \notin \{0,1\}^{|E|}$ **then**
      branch, resulting in nodes o^* and o^{**};
      $T \leftarrow T \cup \{o^*, o^{**}\}$; `/* add children to the tree */`
return z^*

5.3.3 Heuristic

An intrinsic difficulty of Benders decomposition is that we replace the well-structured problem (P) by a problem (MP) with no straightforward structure. Indeed, it is well known that special structures can help the solution of hard integer programs. For instance, detecting a flow structure within a more complicated problem can be used to add strong cut inequalities see Achterberg and Raack in [2]. Moreover, for many problems Benders cuts have fractional coefficients, yielding numerical instability (Codato and Fischetti in [23] avoid this difficulty for combinatorial problems with certain classes of "big M" constraints). Finally, it will be hard for our default MIP solver (CPLEX 11 in our

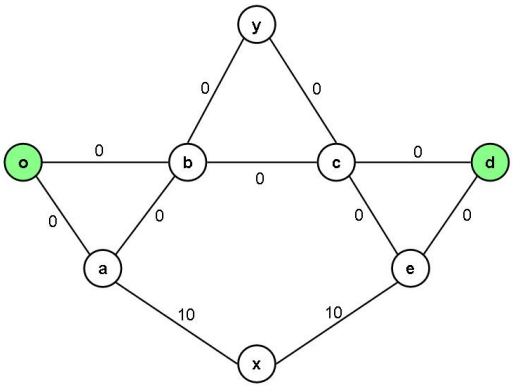


Figure 5.2: Graph obtained from $\overline{z}$

case) to find good upper bounds. We present next a simple, yet efficient, heuristic. First, we solve the linear programming relaxation of the Benders decomposition, resulting in a fractional $\overline{z}$. Then, for each $\overline{z}_{ij} = 0$ we add the constraint $z_{ij} = 0$ to (MP), and we solve the resulting problem with `bc-n`. This allows us to reduce significantly the number of variables of the problem, yielding a very good solution in a limited amount of time. The issue whether or not the heuristic finds a feasible solution is discussed in Section 5.4.

We present in Section 5.5 the heuristic quality and the solution time of a new branch-and-cut algorithm , `bc-n-heur`, starting with the upper bound from the heuristic.

5.4 Feasibility problem

Note that the feasibility problems $SP(q, \overline{z})$ and $FP(q, \overline{z})$ and the Benders cuts (5.7) are independent for each commodity $q \in Q$, so that without loss of generality, we assume in this section that we have a unique commodity q going from o to d. Let us come back to the problem of knowing whether Benders cuts (5.7) together with integrality restrictions on z are sufficient to describe $\mathcal{Z}_i$, or in other words, whether $\mathcal{Z}_i = \mathcal{Z}_c$. Simply stated, are the set of feasible network designs equal for (P) and (P') ? This is equivalent to knowing whether $\mathcal{X}_i(\overline{z}) = \emptyset$ implies that $\mathcal{X}_c(\overline{z}) = \emptyset$, for any binary vector $\overline{z}$. Notice that the inclusion $conv(\mathcal{X}_i(\overline{z})) \subseteq \mathcal{X}_c(\overline{z})$ may be strict. Consider the example with the binary $\overline{z}$ described on Figure 5.2, where each edge $ij \in E$ has a routing cost $\tilde{c}_{ij}$, which refers to an example with $L = 4$ and $K = 2$. There are 5 different 4-paths from o to d: the ones shown on Figure 5.3 and the path $(o - b - c - d)$. There are only two pairs of disjoint paths, $\{P3, P4\}$ from Figure 5.3 and $\{P4, (o - b - c - d)\}$, both with a cost equal to 20. Then, the fractional optimal solution routes 0.5 unit on each path (P_1, P_2, P_3 and P_4) from Figure 5.3 yielding a total routing cost equal to 10. Thus, the cheapest fractional routing is less than the cheapest integer routing, implying $conv(\mathcal{X}_i(\overline{z})) \subset \mathcal{X}_c(\overline{z})$.

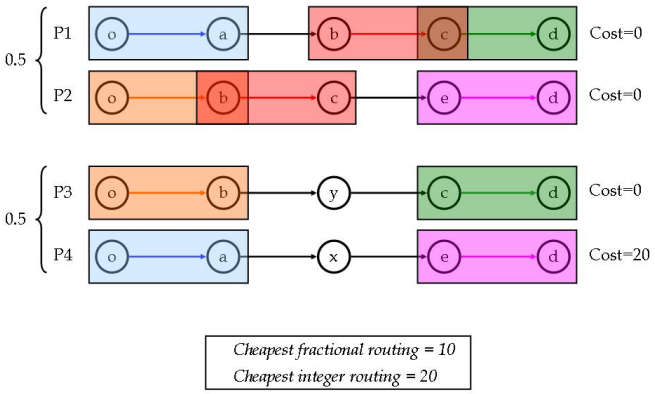


Figure 5.3: Fractional and integer minimum cost routing when $L = 4$ and $K = 2$.

Let us recall some well-known families of cuts used by Huygens *et al.* in [63] and [62] to describe a formulation for (P) using only design variables. In the sequel, we show that cuts (5.7) generalize these families of cuts. Therefore, using results from [30, 34] and Lemmas below, we obtain that Benders cuts together with binary constraints completely describe $\mathcal{Z}_i$ for $L = 2, 3$ and any $K \geq 1$. Then, we give an example (see Figure 5.5) showing that cuts (5.7) may be interesting when $L \geq 5$.

We introduce first some notations. If $W \subset V$ is a node subset, then the set of edges that have one node in W and one node in $V \backslash W$ is called a cut and denoted by $\delta(W)$, and $z(\delta(W)) := \sum_{ij \in \delta(W)} z_{ij}$. For $o, d \in V$, a cut $\delta(W)$ such that $o \in W$ and $d \in V \backslash W$ is called a **od-cut**. Then, let $V_0, V_1, \ldots, V_{L+1}$ be a partition of V such that $o \in V_0$, $d \in V_{L+1}$ and $V_i \neq \emptyset$ for $i = 1, \ldots, L$. A set of edges $T \subset E$ is called a **L-path-cut** if for each $ij \in T$, $i \in V_v$, $j \in V_w$ such that $|v - w| > 1$.

For any L and $K = 1$, Dahl in [27] proves that a binary vector $\overline{z}$ belongs to $\mathcal{Z}_i$ if it satisfies the following inequalities:

$$z(\delta(W)) \geq K, \quad \text{for all } od - cuts\ \delta(W), \tag{5.9}$$

and the $L - path - cut$ inequalities

$$z(T) \geq K, \quad \text{for all } L - path - cuts\ T. \tag{5.10}$$

In [63], Huygens *et al.* extend (5.9) and (5.10) to $L = 2, 3$ and $K = 2$, which together with binary restrictions on z, provide a valid formulation for the problem. In further works [30, 34], the authors prove the formulation to be valid for any $K \geq 2$. Consider now a partition $V_0, V_1, \ldots, V_{L+r}$ of V such that $o \in V_0$, $d \in V_{L+r}$ and $V_i \neq \emptyset$ for $i = 1, \ldots L + r - 1$. Generalizing (5.10), Dahl and Gouveia in [29] introduce the **generalized jump inequality**

$$\sum_{i\in V_v, j\in V_w, v\neq w} \min(|v-w|-1, r) z_{ij} \geq Kr. \tag{5.11}$$

Finally, Huygens and Mahjoub in [62] introduce the **two-layered 4-path-cut** specifically for the case $L = 4$ and $K = 2$. Let $V_0, V_1, \ldots, V_6, W_1, \ldots, W_4$ be a partition of V such that $o \in V_0$, $d \in V_6$ and $V_i \neq \emptyset$ for $i = 1, \ldots 5$. They define the inequality

$$ax \geq 4, \tag{5.12}$$

with

$$\begin{array}{ll}
a_{ij} = \min(|v-w|-1, 2), & i \in V_v, j \in V_w, i \neq j, \\
a_{ij} = 2, & i \in W_v, j \in W_w, |v-w| \geq 2, \\
a_{ij} = 2, & i \in V_v, j \in W_w, w - v \geq 2 \text{ or } v - w \geq 3, \\
a_{ij} = 1, & i \in V_v, j \in W_w, (v,w) = (2,3), (3,1), (3,4), (4,2), \\
a_{ij} = 0, & \text{otherwise.}
\end{array} \tag{5.13}$$

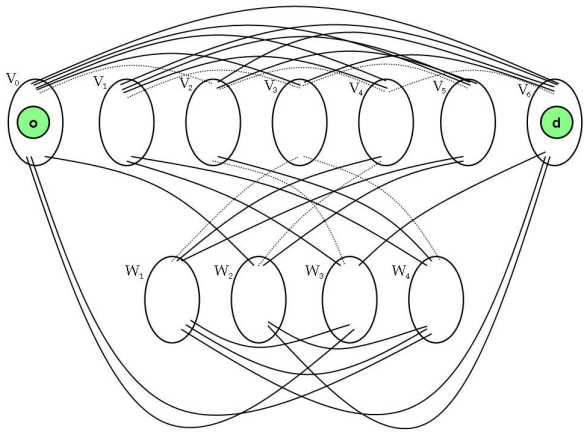


Figure 5.4: Support graph a two-layered 4-path-cut inequality, inspired by Figure 4 from [62].

They have shown that inequalities (5.12), besides (5.9) and (5.10), are needed to obtain a valid formulation for $L = 4$ and $K = 2$.

Next, we prove that cuts (5.7) generalize cuts (5.9), (5.11) and (5.12) (thus (5.10) because it is a special case of (5.11)).

Lemma 1. *Consider some od − cut $\delta(W)$. Inequality* (5.9) *for $\delta(W)$ is a Benders cut* (5.7) *for some vector $(\overline{\pi}, \overline{\sigma})$ feasible for $SP(q, \overline{z})$.*

Proof. Setting $\overline{\pi}_d^{L+1} = 1$, $\overline{\sigma}_{ij} = 1$ for $ij \in \delta(W)$ and 0 otherwise, (5.7) becomes

$$\sum_{ij\in\delta(W)} z_{ij} \geq K.$$

l	1	2	3	4	5
V_0	$-/0$	0	0	0	–
V_1	–	0	0	0	–
V_2	–	0.5	0	0	–
V_3	–	1	0.5	0	–
V_4	–	1	1	0.5	–
V_5	–	1	1	1	–
V_6	–	1	1	1	$-/1$

Table 5.1: Values of $\overline{\pi}$ for the *generalized jump inequality* for $L = 4$ and $r = 2$.

l	1	2	3	4	5
W_1	–	0	0	0	–
W_2	–	1	0	0	–
W_3	–	1	1	0	–
W_4	–	1	1	1	–

Table 5.2: Values of $\overline{\pi}$ for nodes in W for the *two-layered 4-path-cut inequality*.

We are left to set up $\overline{\pi}_i^l$ for $l < L+1$ so that $(\overline{\pi}, \overline{\sigma})$ is feasible for $SP(q, \overline{z})$. It is easy to see that $\overline{\pi}_i^l = 0$ for $i \in W$, and $\overline{\pi}_i^l = 1$ for $i \in V \backslash W$, satisfies this requirement. □

Lemma 2. *Let $(V_0, V_1, \ldots, V_{L+r})$ be some partition of V such that $o \in V_0$, $d \in V_{L+r}$ and $V_i \neq \emptyset$ for $i = 1, \ldots L+r-1$. Inequality* (5.11) *for this partition is a Benders cut* (5.7) *for some vector $(\overline{\pi}, \overline{\sigma})$ feasible for $SP(q, \overline{z})$.*

Proof. First, we must set $\overline{\pi}_d^{L+1} = 1$, $\overline{\sigma}_{ij} = r^{-1} \min(|v-w|-1, r)$ for $i \in V_v, j \in V_w$, so that (5.7) becomes

$$r^{-1} \sum_{i \in V_v, j \in V_w, v \neq w} \min(|v-w|-1, r) z_{ij} \geq K,$$

equal to (5.11) by multiplying both sides by r. We are left to set up $\overline{\pi}_i^l$ for $l < L+1$ so that $(\overline{\pi}, \overline{\sigma})$ is feasible for $SP(q, \overline{z})$. First, set $\overline{\pi}_o^1 = 0$ and $\overline{\pi}_d^l = 1$ for $l = 2, \ldots, L$. For each $0 \leq k \leq L+r$, let $i \in V_v$ and set $\overline{\pi}_i^l = r^{-1} \min(v+1-l, r)$ for $l = 1, \ldots, v$, $\overline{\pi}_i^l = 0$ for $l = v+1, \ldots, L+r-1$, see Table 5.1 for $r = 2$ and $L = 4$. We must check that for any $i \in V_v$, $j \in V_w$, and $1 \leq l \leq L$, the arc (i, j, l) satisfies $\overline{\sigma}_{ij} \geq \overline{\pi}_j^{l+1} - \overline{\pi}_i^l$. By definition of $\overline{\pi}$, $\overline{\pi}_j^{l+1} - \overline{\pi}_i^l = \frac{y}{r}$ for some $1 \leq y \leq r$ implies that $w \geq v+y+1$ so that $\overline{\sigma}_{ij} \geq \frac{y}{r}$. Thus, $(\overline{\pi}, \overline{\sigma})$ satisfies all equations of $SP(q, \overline{z})$. □

Lemma 3. *Let $V_0, V_1, \ldots, V_6, W_1, \ldots, W_4$ be a partition of V such that $o \in V_0$, $d \in V_6$ and $V_i \neq \emptyset$ for $i = 1, \ldots 5$. Inequality* (5.12) *for this partition is a Benders cut* (5.7) *for some vector $(\overline{\pi}, \overline{\sigma})$ feasible for $SP(q, \overline{z})$.*

Proof. We set $\overline{\pi}_d^{L+1} = 1$ and $\overline{\sigma}_{ij} = \frac{1}{2} a_{ij}$, with a defined in (5.13), and $\overline{\pi}$ as in Tables 5.1 and 5.2. The rest of the proof is similar to the proof of Lemma 2. □

As a result of Lemmas 1, 2 and 3, [27], Theorem 2.2 from [63] (and its generalization to any K in [30] and [34]) and Theorem 3 from [62] we obtain:

Proposition 1. *The sets $\mathcal{Z}_c$ and $\mathcal{Z}_i$ are equal for $L = 2, 3$ with any $K \geq 2$, and $L = 4$ with $K = 2$.*

In particular, we obtain that `heuristic` described in subsection 5.3.3 shall find a feasible solution in this context:

Corollary 1. *For $K = 1$ with any $L \geq 1$, $L = 2, 3$ with any $K \geq 2$, and $L = 4$ with $K = 2$, the algorithm* `heuristic` *will always find a feasible design for (P) .*

Proof. Let $(\overline{z}, \overline{x})$ be an optimal solution to the linear programming relaxation of (P). Thus, $\overline{z}$ satisfies all Benders cuts (5.7). Then, since each component of $\overline{\sigma}$ is positive or zero, $\lceil \overline{z} \rceil$ satisfies all (5.7) as well, so that $\lceil \overline{z} \rceil \in \mathcal{Z}_c$. Therefore, Proposition 1 implies that $\lceil \overline{z} \rceil \in \mathcal{Z}_i$. □

It is natural to wonder whether the equality $\mathcal{Z}_c = \mathcal{Z}_i$ holds for $L = 4$ and $K \geq 3$, and $L \geq 5$ and $K \geq 2$. Although, we do not know the complete answer, Theorem 3.3 from [65] leads to the following partial answer.

Proposition 2. *For each $L \geq 5$, there exists a $K \geq 2$ for which the inclusion $\mathcal{Z}_i \subset \mathcal{Z}_c$ holds strictly.*

Proof. We prove this result by contradiction. Consider some $L \geq 5$ and assume that $\mathcal{Z}_i = \mathcal{Z}_c$ for each $K \geq 2$. Consider some $\overline{z} \in \mathcal{Z}_i$ and let $G = (V, E)$ be the graph described by $\overline{z}$, i.e., $ij \in E$ if and only if $\overline{z}_{ij} = 1$. For each $K \geq 2$, we can check whether there exists K edge-disjoint L-paths between o and d by solving $SP(q, \overline{z})$, because $\mathcal{Z}_i = \mathcal{Z}_c$. Thus, this existence question can be answered in polynomial time for any K. Since the maximum number of such paths is bounded by the number of vertices of V, the problem of finding the maximum number of edge-disjoint L-paths between o and d is polynomial, which contradicts Theorem 3.3 from [65]. □

Finally, let us show that cuts (5.7) contain new valid inequalities for the problem for $L \geq 5$. First, note that in [62] the authors introduce the **two-layered L-path-cut inequalities**, extending the **two-layered 4-path-cut inequalities** to general L. It can be easily seen that Lemma 3 can be extended to incoporate these generalized inequalities. More important is the fact that they show on three examples that these new inequalities, together with (5.9), (5.11) and binary restrictions on z, are not sufficient to formulate the problem for $L \geq 5$. For instance, consider the graph shown in Figure 5.5. We see that it is impossible to find two edge-disjoints paths from o to d with length smaller than or equal to 5, but Huygens and Mahjoub were not able to provide any inequality that cuts off this design. We describe next a violated Benders cut that cuts off the solution depicted in Figure 5.5. Let $G = (V, E)$ be the graph in Figure 5.5, $\overline{z}_{ij}$ the binary vector made of 9 consecutive ones, and q a commodity in G from o to d. First, note that a violated Benders cut must exist because it is impossible to find a fractional flow satisfying (5.1),(5.2) and (5.5) in the layered graph constructed from G (so that its dual is unbounded, yielding a violated Benders cut).

Then, define the following dual variables: $\overline{\sigma}_{og} = \overline{\sigma}_{cg} = \overline{\sigma}_{gd} = 0.5$, $\overline{\sigma}_{ij} = 0$ for $ij \in E \backslash \{og, cg, gd\}$, and $\overline{\pi}$ is described in Table 5.3. One can check that $(\overline{\pi}, \overline{\sigma})$ belongs to $SP(q, \overline{z})$ for G.

Moreover, they yield the Benders cut

$$0.5z_{og} + 0.5z_{cg} + 0.5z_{gd} \geq 2, \tag{5.14}$$

violated by $\overline{z}$. This cut can be extended for more general graphs, partitioning the nodes into 8 subsets, see Figure 5.6. Then, setting $(\overline{\pi}, \overline{\sigma})$ accordingly, we obtain a cut involving

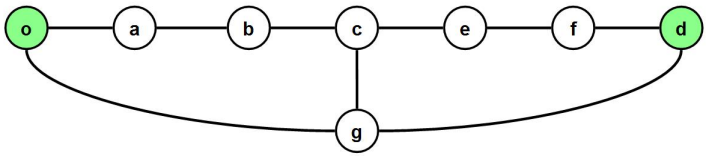


Figure 5.5: Infeasible solutions for the edge THPP with $L = 5$, inspired by Figure 6 from [62].

l	1	2	3	4	5	6
o	0	0	0	0	0	–
a	–	0	1	0	1	–
b	–	1	0	1	0	–
c	–	1	1	0	1	–
e	–	1	1	1	0	–
f	–	1	1	1	1	–
g	–	0.5	0.5	0.5	0.5	–
d	–	1	1	1	1	1

Table 5.3: Values of $\overline{\pi}$ for cut (5.14).

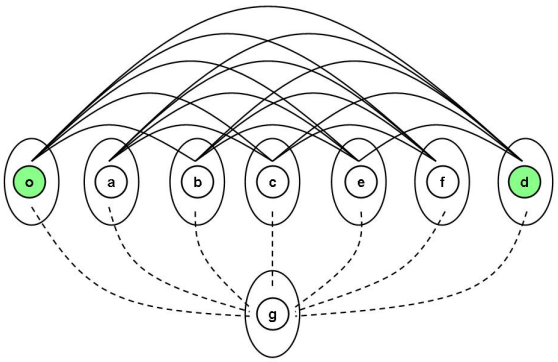


Figure 5.6: New inequality for $L = 5$.

the edges represented in Figure 5.6, with coefficients equal to 1 (respectively 0.5) for plain (respectively dashed) edges. Similar cuts can be obtained in this way for the other examples in [62]. Such cuts were independently discovered by [54].

5.5 Computational results

In this section we compare the solution times of formulations (P) and (P') (see the commentary below), which we denote `layered` and `layered-r`, respectively, cutting plane algorithms `cp` and `cp-i`, and branch-and-cut approaches `bc-all`, `bc-int`, `bc-5`

and `bc-5-heur`. Then, for the branch-and-cut approaches we compare the number of cuts generated and the number of nodes visited in the branch-and-cut tree. Finally, we evaluate the quality of the upper bound given by `heuristic`.

We have discussed in Section 5.4 whether we can relax the integrality restrictions (5.4). We have answered affirmatively for some values of L and K, see Proposition 1, although we have never encountered an instance for which a feasibility cut (5.8) was needed. Thus, Benders cuts were enough to describe the problem for all the instances in our computational experiments, so that models `layered` and `layered-r` coincide for these instances. Therefore, we also present the computational time required by `layered-r`. Note that we tested branch-and-cut algorithms without the feasibility part as well, but the speed-up was insignificant, so that we do not report them in the remainder.

5.5.1 Implementation details

All models have been coded in JAVA using CPLEX 11 MIP solver and run on a DELL Latitude D820 with a processor Intel Core Duo 2 T7200 of 2 GHz and 2.5 GB of RAM memory. We allow CPLEX to store the branch-and-bound tree in a file, setting parameter *IntParam.NodeFileInd* to 2, to avoid from running out of memory. Moreover, for each algorithm we configure CPLEX as follows:

`layered` and `layered-r` All parameters have been kept to their default values, CPLEX chooses to explore the branch-and-bound tree with the dynamic search.

`cp`: We build an empty IP for the master problem, $|Q|$ LP's for the Benders subproblems and $|Q|$ IP for the feasibility subproblems. Then, we cycle between these problems, with all parameters kept to their default values. CPLEX uses the dynamic search for solving all IP's.

`cp-i`: We build an empty LP for the master problem and solve it through a cutting plane algorithm. Then, we build an IP for the master problem, the constraints of which are the Benders cuts just generated; we solve it through a cutting plane algorithm, with all parameters kept to their default values. CPLEX uses the dynamic search.

`bc-all`, `bc-int`, and `bc-n`: Since the model does not contain explicitly all constraints, we must deactivate the dual presolve, setting *BooleanParam.PreInd* to false. Then, we implemented our (global) cuts generation with a *LazyConstraintCallback*, preventing CPLEX from using dynamic search.

`bc-n-heur`: We use the algorithm `bc-n`, providing CPLEX with the upper bound found by `heuristic`. The CPU times reported do not consider the time spent in `heuristic`.

`heuristic`: We first solve the linear programming relaxation by a cutting plane algorithm (in fact, we use a branch-and-cut algorithm with a limit of 0 nodes, setting *IntParam.NodeLim* to 0). Then, we fix some of the variables to 0, and re-solve the resulting problem with `bc-n`.

Name	$\|N\|$	$\|E\|$	$\|Q\|$	# of instances	Rooted demands?
TC-5	21	210	5	5	true
TC-10	21	210	10	5	true
TE-5	21	210	5	5	true
TE-10	21	210	10	5	true
pdh	11	34	27	1	false
di-yuan	11	42	48	1	false
dfn-gwin	11	47	9	1	false
polska	12	18	17	1	false
nobel-us	14	21	33	1	false

Table 5.4: Instances description.

L/K	1	2	3
3	21.40	10.90	5.47
4	24.30	9.45	5.69
5	26.70	6.00	5.16

Table 5.5: Geometric average of gap $\left(\frac{IP^*-LP^*}{IP^*} * 100\right)$ for all instances.

5.5.2 Instances details

We used three different test sets. The sets TC and TE were taken from a class of complete graphs $G = (V, E)$, reported in [51]. They share the following features: $|V| = 21$, $|Q| \in \{5, 10\}$, and all point-to-point demands share one of their extremities (which we call rooted demands in Table 5.4). The cost matrix for each instance considers the integer part of the Euclidean distance between the coordinates of the 21 nodes, randomly placed among the integer points of a grid (100×100). The TC class contains 5 instances with 5 commodities and 5 instances with 10 commodities with the root located in the center of the grid and the TE class contains 5 instances with 5 commodities and 5 instances with 10 commodities with the root located on a corner of the grid. We see in the next section that instances TE are much harder to solve than instances TC. Then, five instances are based on sparse networks from *SNDlib* [89]: pdh, di-yuan, dfn-gwin, polska and nobel-us. Table 5.4 summarizes the size of the instances. We solved the problem for L ranging from 3 to 5 and K from 1 to 3. We set a time limit of 3600 seconds for all instances and algorithms.

5.5.3 Results

First, we look at the quality of the linear programming relaxation of our model (P). Let IP^* and LP^* be the optimal value of (P) and its linear programming relaxation, respectively. Table 5.5 shows that the gap $\left(\frac{IP^*-LP^*}{IP^*} * 100\right)$ decreases as K increases. It means that the extended formulation integrates well the survivability constraints.

Before comparing the different algorithms, we need to determine the "best" value for the depth parameter of branch-and-cut algorithm `bc-n`. We select a group of complicated instances (instances that `layered` can not solve to optimality within 3600 seconds) and

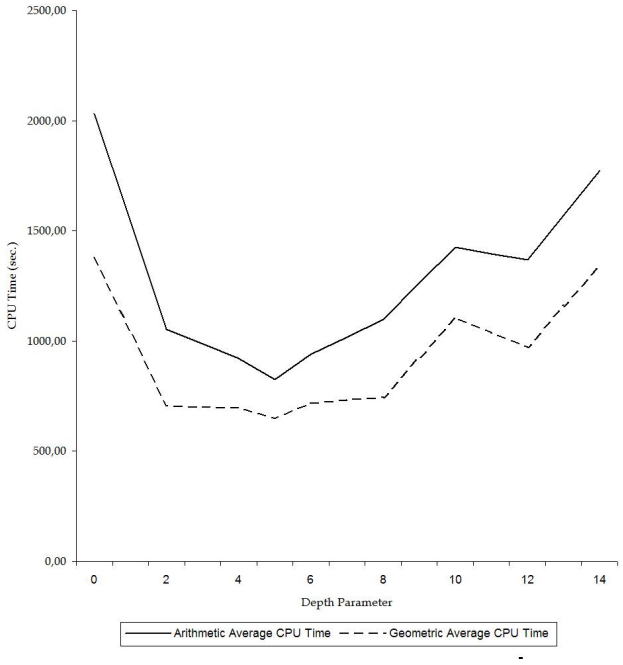


Figure 5.7: `bc-n` depth parameter tuning by average CPU time (sec.)

we test different values of the depth parameter n. On Figure 5.7, we plot the result of this tuning stage. For both curves, the minimum is reached when $n = 5$. Therefore, in the sequel we always consider `bc-5` for the hybrid branch-and-cut algorithm.

We compare the performance in terms of resolution time for the different methods by plotting the performance profile (see for instance [35]) on Figure 5.8. Clearly, algorithms `bc-5`, `bc-int` and `bc-5-heur` are the fastest algorithms. Moreover, we see that the simple and basic implementation of the Benders decomposition `cp` performs much worse than the original model `layered`. Indeed, Benders decomposition suffers from the loss of problem structure, so that each of the iterations of the master problem requires a reasonable amount of time (see Table 5.9 for average numbers of iterations). Thus, a careful implementation is required to make the decomposition efficient.

Table 5.6 indicates, for each algorithm, the number of instances (out of the 213 instances which compose the entire test set) that can not be solved within the 3600 seconds. Among the 13 or 12 instances that `layered` or `layered-r` can not solve to optimality, only 1 can not be solved by `bc-5`. `b-c-heur` can solve all instances to optimality. The reader can find the arithmetic averages of CPU times on Table 5.7 (">" indicates that one or more instances could not be solved to optimality).

Table 5.8 indicates the arithmetic average values of the optimal solutions (IP*), linear programming relaxations (LP*), heuristic solutions (heuristic*), and `heuristic` CPU time in seconds. Moreover, the arithmetic averages of the gap between LP* and IP* are computed, $\frac{IP^*-LP^*}{IP^*}$ and between heuristic* and IP*, $\frac{heuristic^*-IP^*}{IP^*}$. Column "Optimal"

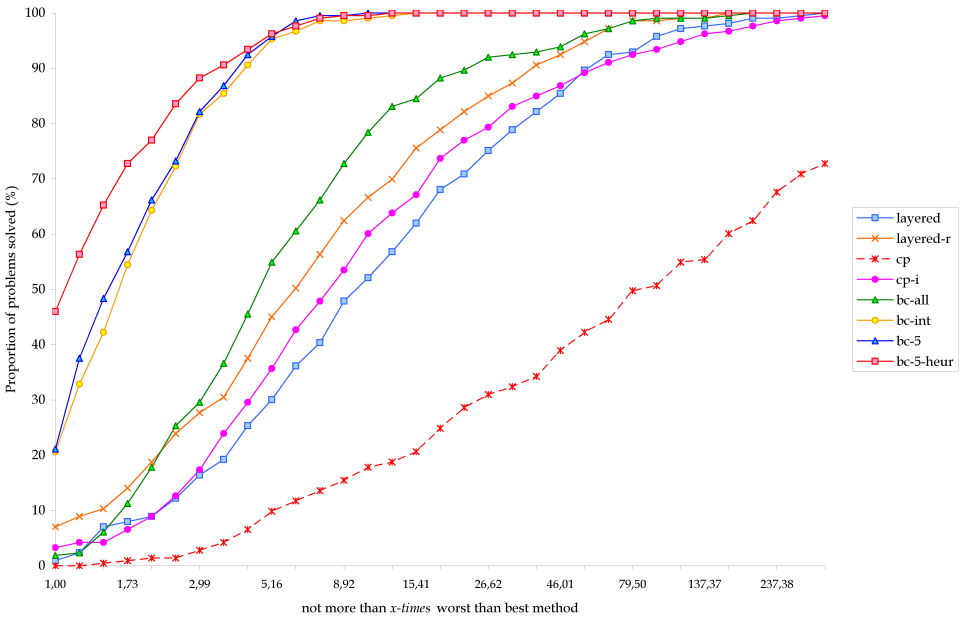


Figure 5.8: Performance profile comparing methods on the entire test set.

layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur
13	12	90	30	20	4	1	0

Table 5.6: Number of instances (out of 213) unsolved within one hour.

provides the number of instances for which the heuristic solution is optimal. It can be seen that `heuristic` always provide a very good solution to the problem.

Furthermore, `heuristic` is also pretty fast, taking around 4 seconds whereas `layered` and `bc-5-heur` take respectively on average 582.05 and 74.03 seconds. In 139 cases out of 213 (around 65%), the solution given by the heuristic is optimal. Finally, Table 5.10 presents arithmetic averages of the number of Benders cuts generated by the branch-and-cut algorithms, and number of nodes explored by branch-and-cut and extended formulations, and Table 5.11 provides means of CPU time spent for solving Benders subproblems and the corresponding fraction in the total CPU time (means have been taken over all instances).

5.6 Summary

In this chapter we give an extended formulation valid for any L and K for the hop-constrained survivable network design problem in the multicommodity case. This formulation is composed of a huge number of variables and constraints that is the reason why

we develop cutting plane approaches and some branch-and-cut algorithms based on the Benders decomposition. We test thanks to the different versions of our branch-and-cut algorithms where, in the branch-and-bound enumeration tree, it is more interesting to add the Benders cuts. Our best algorithm combined and helped by an intuitive heuristic solves satisfactorily all the instances tested (among with a set of real based instances from *SNDlib*) on average faster than CPLEX 11. From a theoretical point of view we prove, in this chapter, that our Benders cuts contain the constraints used by Huygens *et al.* to formulate the problem for $L = 2, 3, 4$, as well as new inequalities when $L \geq 5$.

	layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	heuristic
Arithmetic Mean	> 582.05	> 498.09	> 1817.87	> 680.27	> 494.40	> 151.87	> 89.78	74.03	3.78

Table 5.7: Average CPU times for all approaches.

Instances	**IP***	**LP***	**Gap(%)**	**heuristic***	**Gap(%)**	**Optimal**	**heuristic CPU Time**
TC-5	248.27	231.97	8.04	252.07	1.77	24/45	0.76
TC-10	382.51	343.34	12.96	384.00	0.42	31/45	4.78
TE-5	316.78	291.03	10.67	322.36	2.17	24/45	0.97
TE-10	446.20	379.06	17.72	447.98	0.36	30/45	9.39
pdh	1480322.56	1216582.21	21.25	1480285.35	0.00	9/9	6.51
di-yuan	46156666.67	37860533.01	20.96	46156666.67	0.00	9/9	2.21
dfn-gwin	114280.00	97639.51	16.33	114753.33	0.41	3/6	0.59
polska	302420.00	274481.67	11.11	302420.00	0.00	5/5	0.37
nobel-us	13966000.00	12806125.00	10.00	13966000.00	0.00	4/4	1.39
Arithmetic Mean	2285715.50	1901093.08	13.13	2285729.93	1.01	-	3.78

Table 5.8: LP Relaxation, Integer optimal solution and heuristic performances for the entire test set.

Instances	`cp`	`cp-i`	
		linear (MP)	integer (MP)
TC-5	54.07	20.53	11.11
TC-10	97.00	28.22	22.09
TE-5	229.87	34.71	36.64
TE-10	126.29	37.58	51.51
pdh	174.67	13.33	111.44
di-yuan	81.33	9.89	20.67
dfn-gwin	94.33	17.00	29.50
polska	56.60	11.40	4.80
nobel-us	147.25	11.50	21.75
Arithmetic Mean	124.73	27.52	32.57

Table 5.9: Average numbers of iterations.

Instances	**Number of cuts generated**				**Number of nodes visited**					
	`bc-all`	`bc-int`	`bc-5`	`bc-5-heur`	`bc-all`	`bc-int`	`bc-5`	`bc-5-heur`	`layered`	`layered-r`
TC-5	342.98	120.33	128.20	94.71	135.27	405.69	311.98	917.58	118.27	149.09
TC-10	6616.56	567.80	780.49	378.36	885.60	6825.44	3545.40	13972.93	1241.31	741.11
TE-5	757.13	236.62	248.40	159.73	169.29	1165.84	520.51	2204.31	206.98	177.56
TE-10	17154.53	1693.60	2008.31	1029.40	1310.00	48490.58	13983.40	43201.62	3402.56	2207.69
pdh	6270.33	1219.22	1205.89	1103.22	2652.78	7831.33	6773.78	9663.11	4464.56	2009.89
di-yuan	1349.44	585.22	592.22	541.56	210.89	753.78	549.33	1614.56	1202.89	243.78
dfn-gwin	1221.33	289.67	310.17	200.83	415.00	1029.33	909.17	787.17	632.00	410.00
polska	110.00	110.20	103.80	105.00	16.00	31.40	25.00	88.80	8.20	14.80
nobel-us	224.25	238.75	219.25	234.75	17.75	47.00	28.25	242.50	45.25	14.25
Arithmetic Mean	5617.64	644.65	760.01	433.20	661.60	12411.86	4215.30	13244.02	1308.13	799.38

Table 5.10: Number of cuts generated and of nodes visited for the entire test set.

	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	heuristic
CPU time (Arithmetic mean)	1.39	0.81	86.31	9.94	10.46	6.14	1.91
Fraction of total time (Geometric mean)	0.25	1.34	37.59	23.98	29.81	23.41	17.31

Table 5.11: CPU time spent for solving Benders subproblems.

Chapter 6

On the hop-constrained survivable network design problem with reliable edges

6.1 Introduction

THis chapter focuses on an extension of the previous multicommodity survivable network design problem with hop constraints. We consider an undirected graph $G = (V, E)$, where V represents the vertex set, and E the set of potential edges. We also associate a cost c_{ij} to each edge ij and introduce an auxiliary arc set A which is obtained from each edge ij of E by creating two directed arcs (i, j) and (j, i) with the same cost as the original edge. We consider also a set Q of different commodities that must be routed through the network. A commodity $q \in Q$ can be represented by a pair of node $o(q)$ and $d(q)$ which have to be connected.

In this chapter, we study a different version of the problem studied in the previous chapter where a subset of edges, E_R, of the network represented by the graph $G = (V, E)$, is composed of very reliable edges (as an extension, we could also consider the similar problem with subset of reliable nodes V_R). Essentially, these edges are not prone to failure and the edge-disjointness should be guaranteed only in the remaining edge set $(E \setminus E_R)$. It means that, potentially, all the K paths can use simultaneously each reliable edge ij of the network. In other words, the quantity of information going through a reliable edge ij is no more limited to 1 but is bounded, here, by the value of the parameter K. Of course this advantage as a bad consequence in terms of cost because reliable edges are, in a certain way, more expensive. As reliable edges can be used at most K times, it is obvious that their installation cost is higher that the installation cost of similar "normal" edges. Here, we decide to add an incentive to use reliable edges because this class of edges is used to facilitate the tasks of managers, that is the reason why we propose to integrate a scale economy principle when we consider the cost point of view for reliable edges. The idea is the following, if a quantity of flow of K units transit by a reliable edge ij, the installation cost of edge ij will be less than K times the normal cost of the same

edge. In the following of this study we propose to use for instance $((K-1)+f)$ times the normal cost of edge ij where f (called here the scale economy factor) is a fractional number between $[0,1[$ (then $((K-1)+f) < K$). As an example, assume that $K=2$, $f=0.2$, the edge ij is reliable and the normal installation cost of edge ij is equal to 10. The cost to install reliable edge ij is equal to 12 which is better than simply using a cost equal to K times the normal installation cost of the edge (here 20).

This version of the problem can find a direct application in the field of survivable multi-layer network design. Indeed, nowadays, the architecture of communication networks is more complicated than a simple subset of routers or nodes linked by a subset of edges. Today telecommunications networks are designed with a layered structure according to different technologies and needs in which the different layers interact between each others in a hierarchical way. Simply we could consider a bi-layer network structure composed of a virtual layer over a physical layer (or transport layer) as illustrated on Figure 6.1.

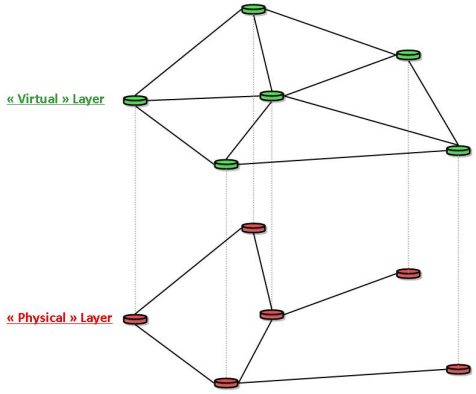


Figure 6.1: Example of a bi-layered architecture.

In those kind of networks, demands are usually given in the virtual layer as quantities of information to send between pairs of nodes. Those demands have to be routed in the virtual layer, leading to the installation of "virtual links" (which are routers or other devices) on the edges belonging to the paths used for the routing. Consequently, virtual links allocated to the edges used in the virtual layer define new demands between extremities of those edges for the lower level layer, here the transport layer, leading to the installation of links (optical fibers, copper links, etc.), in the physical layer. Therefore, when a demand is routed through a path in the virtual layer (composed of many virtual edges), each edge corresponds to a path in the layer underneath (also called a "grooming path").

We first provide an example of multi-layer network design without reliable edges. Consider the bi-layered network composed of six connected routers represented on previous Figure 6.1. Figure 6.2 (a.) illustrates a single demand between two nodes in the virtual layer which is represented by the blue curve between the two blue end nodes. This

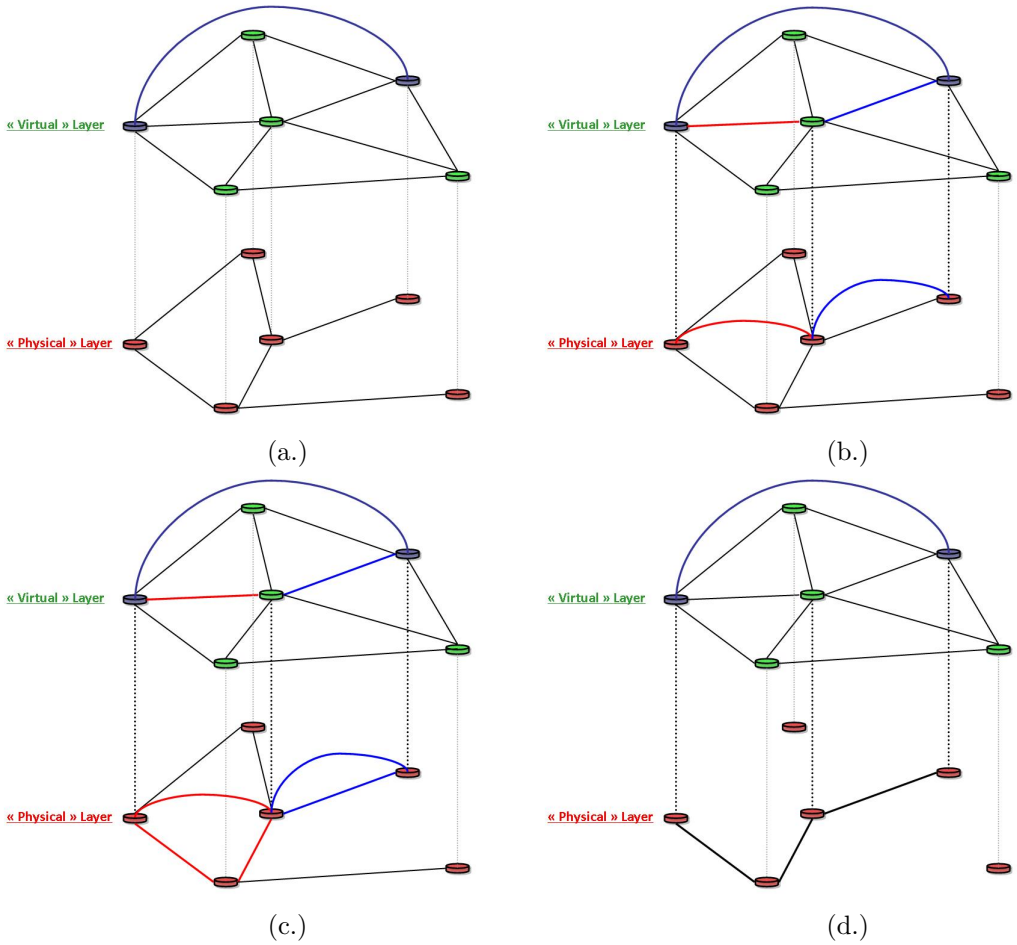


Figure 6.2: Links between the various layers of a bi-layered network architecture.

demand is routed on the virtual layer using the selected red and blue edges on which virtual links are now installed (see Figure 6.2 (b.)). These two virtual links become in the physical layer two new demands (the red and the blue curves) (see Figure 6.2 (c.)). For these demands the physical layer determine a feasible routing using respectively the red and the blue paths defining links for the three selected edges (see Figure 6.2 (d.)).

In the literature, ([70, 9, 44, 69]) multi-layer network design problems also consider modular capacity on the links which is a generalization of the fixed-charge network design problems studied in this chapter. As the single-layer capacitated network design problem is complicated enough, many approaches for the bi-layer problem consider the optimization of each layer separately: First, a network design problem is solved for the virtual layer only; then, virtual capacities or virtual links to be installed in the virtual layer define demands for another network design problem, for the physical layer this time. Though much easier to solve, this relaxed approach might provide solutions far from the optimal solution of the problem. Therefore, an integrated approach should be considered. Although network design has been widely studied for many years [106], the

interest in multi-layered network design is more recent and can be traced back to a paper by Dahl and Stoer [32].

In this multi-layer network design problem area, survivability considerations can be added as well as the quality of service aspect via, for instance, a hop-limit approach. Assuming that the survivability component is managed, as it was already deeply studied in this thesis, by searching for each commodity at least K edge-disjoint paths, it could be interesting to consider, for the virtual layer, a subset of "reliable" edges for which the survivability aspect is already integrated in the structure of the network (in other words in the structure of the physical layer). To ease the work of network providers, we can assume that if an edge in the virtual layer is assumed to be reliable, there exists already a physical structure (routers and transmission links for instance) in the physical layer which guarantees that there exist at least K edge-disjoint paths between the extremities of the reliable edge in the physical layer. Consequently the capacity of each reliable edge is equal to the security factor K. This assumption eases the workload of network providers because, for instance, if all the edges of the network are reliable, then the survivability constraints are naturally satisfied. Indeed and as we will show it later, searching K edge-disjoint paths in this kind of reliable network is similar to look for only one path in a network without reliable edges, which is an easier problem.

Assume the original situation of the previous example represented in Figure 6.1, and look at the virtual layer optimization problem. Figure 6.3 (a.) represents a single demand between blue nodes in the virtual layer. Consider now that a survivability constraint is added to the problem which leads to searching for each demand at least 2 edge-disjoint paths taking into account that the blue edge of Figure 6.3 (a.) is reliable. By definition, if an edge is reliable, it is not prone to failure and therefore the edge-disjointness should be guaranteed only for the non-reliable edge set. It means that although the survivability constraint imposes to find at least K edge-disjoint paths, these paths can share reliable edges. This aspect is represented in the physical layer on Figure 6.3 (a.) by the two edge-disjoint paths existing between the extremities of the reliable edge. Figures 6.3 (b.) and (c.) show a feasible solution which does not use the reliable blue edge whereas Figures 6.3 (d.), (e.) and (f.) show a feasible solution using the reliable edge. Figure 6.3 (f.) shows a feasible solution of the network design optimization problem of the physical layer taking into account the reliable nature of the blue edge.

In the remainder of this chapter, because most integrated problems are too complicated to solve and because it is not our main objective to focus on new techniques or approaches to solve multi-layer network design problems, we limit our analysis to the network design optimization problem of the virtual layer only together with quality of service constraints via the hop-limit consideration. About the hop-limit, we consider only the length of the path not taking into account the "inside" length of the reliable edges. By "inside" length of the reliable edge we mean the length, expressed in terms of hops, of the existing structure composed of K edge-disjoint paths inside a reliable edge. The length of one reliable edge is counted as only one hop.

To model this problem we use a generalization of the hop indexed-model used in Gouveia (see [52]) and in Chapters 4 and 5. This model has been considered for related telecom-

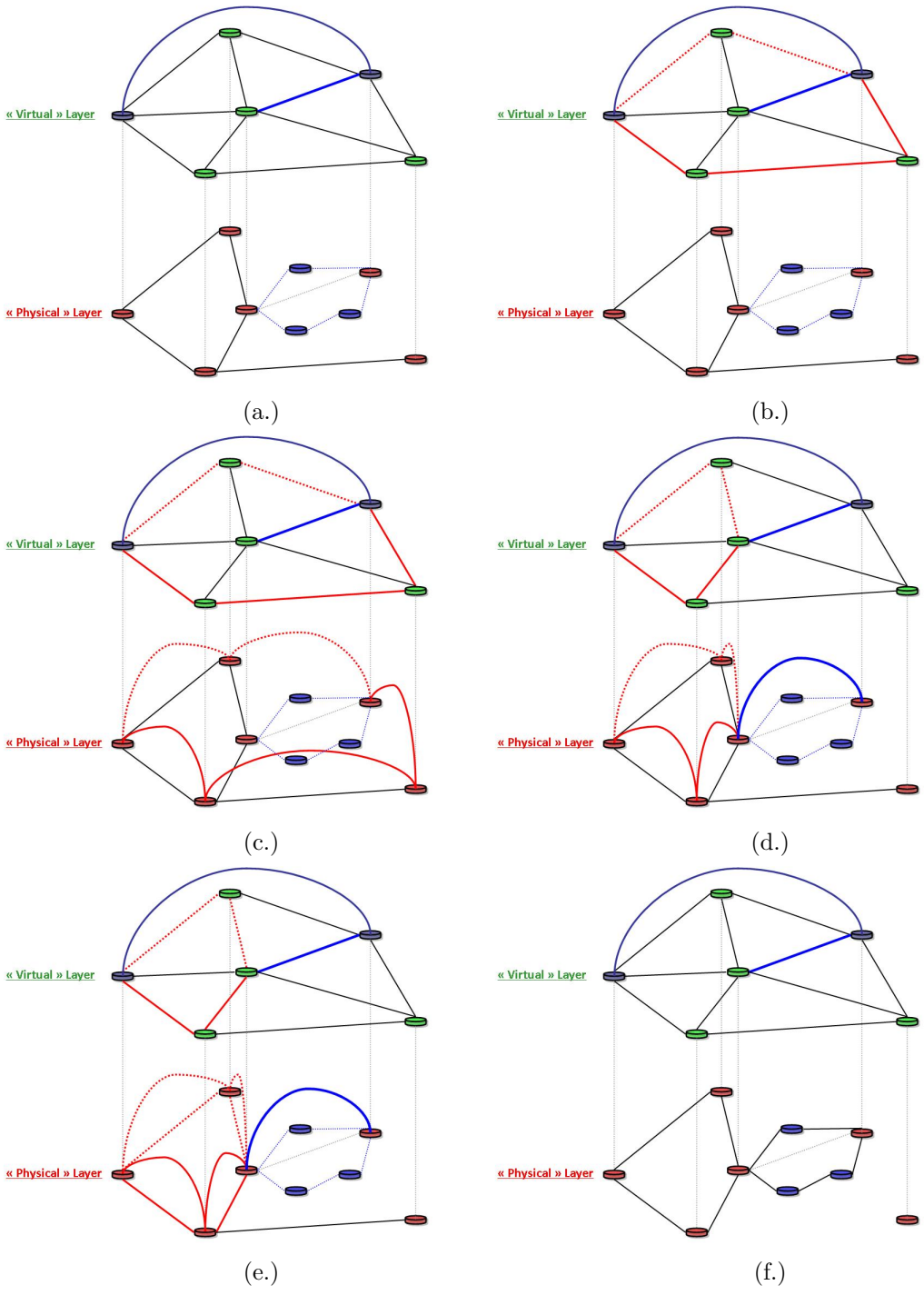


Figure 6.3: Bi-layered architecture and impact of reliable edges.

munication problems as in (see [57],[56]). As far as we know, the variation used here has not been studied yet. As we shall show next, the hop-indexed model is quite adequate to model the variation described here with a slight modification of the model parameters.

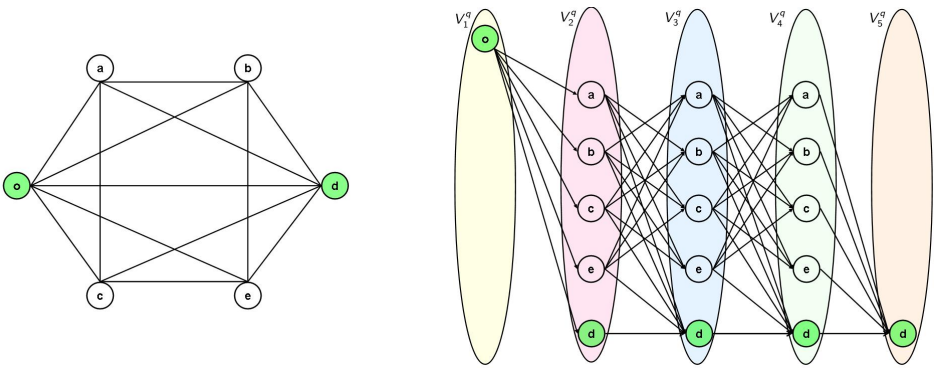
This chapter is organized as follows: In Section 6.2 we propose the basic model (called here Hop) for the traditional survivable network design problem with hop constraints, then we show how slight modifications can be brought in order to introduce the "reliable" edge specificities. We propose two variants of the problem (called HopR1 and HopR2) and we conclude this section by giving some properties linking the original problem and the two proposed variants.

In Section 6.3 and because the three models are large-scale problems (due to the huge number of variables and constraints used) we propose, as it was already done in Chapter 5, to use the Benders decomposition approach. We propose an adapted branch-and-cut algorithm in Section 6.4 before showing in Section 6.5 how the properties of Chapter 5 related to the feasibility problem could be easily extended for this specific problem.

Section 6.6 summarizes the results of a considerable testing phase. Firstly we show how difficult this variation of the problem is compared to the original problem without reliable edges, then we show how fast our branch-and-cut algorithm is compared to a basic implementation in CPLEX of the adapted extended formulations. Finally, we conclude this last chapter with an economical analysis which tries to measure the relative effects or benefits linked to the utilization of reliable edges in the network design step.

6.2 Problem statement

In this section, we first propose an extended formulation for the original survivable network design problem with hop constraints. Then, we propose two adapted formulations for the same problem where we integrate reliable edges. The main idea of [52] is to model the subproblem associated with each commodity with a directed graph composed of $(L+1)$ layers as illustrated in Figure 6.4. Namely, from the original non-directed



(a) (b)

Figure 6.4: Basic Network (a) and its Layered Representation (b) when $L = 4$

graph $G = (V, E)$, we create a directed layered graph $G^q = (V^q, A^q)$ for each commodity, where $V^q = V_1^q \cup \ldots \cup V_{L+1}^q$ with $V_1^q = \{o(q)\}$, $V_{L+1}^q = \{d(q)\}$ and $V_l^q = V \backslash \{o(q)\}$, $l = 2, \ldots, L$. Let v_l^q be the copy of $v \in V$ in the $l - th$ layer of graph G^q. Then, the arcs sets are defined by $A^q = \{(i_l^q, j_{l+1}^q) \mid i_l^q \in V_l^q, j_{l+1}^q \in V_{l+1}^q, l \in \{1, \ldots, L\}\}$, see Figure 6.4. In the sequel, an (undirected) edge in E with endpoints i and j is denoted ij while a (directed) arc between $i_l^q \in V_l^q$ and $j_{l+1}^q \in V_{l+1}^q$ is denoted by (i, j, l) (the commodity q is omitted in the notation as it is often clear from the context).

Note that each path between $o(q)$ and $d(q)$ in layered graph G^q is composed of exactly L arcs (hops), which corresponds to a maximum of L edges (hops) in the original graph. In fact this is the main idea of this transformation since in the layered graph, any path is feasible with respect to the hop constraints.

6.2.1 Extended formulation for the survivable network design problem with hop constraints

The following extended model **Hop** formulates the traditional survivable network design problem with hop constraints. Variables x_{ij}^{lq} model the quantity of flow sent on arc (i, j) when arc (i, j) is used in position l in one of the paths for commodity $q \in Q$. Variables z_{ij} are binary variables which take value 1 if edge ij is used in the optimal topology (if at least one path for at least one commodity uses this edge ij) and 0 otherwise. Constraints (6.1) are flow conservation constraints. To ensure K L-paths we send K units of flow for each commodity $q \in Q$ from the origin vertice $o(q)$ to the destination vertice $d(q)$. Constraints (6.2) model the edge-disjointness.

$$
\begin{aligned}
(Hop) \quad & \min \quad \sum_{ij \in E} c_{ij} z_{ij} & & & \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1) \in A^q} x_{ji}^{l-1q} - \sum_{j:(i,j,l) \in A^q} x_{ij}^{lq} = \begin{cases} -K & \text{if } (i = o(q)) \\ K & \text{if } (i = d(q)) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, & & \\
& & i \in V^q, l \in \{1, \ldots, L+1\}, q \in Q, & (6.1) \\
& \sum_{l \in \{1, \ldots, L\}} \left(x_{ij}^{lq} + x_{ji}^{lq} \right) \leq z_{ij}, & ij \in E, q \in Q, & (6.2) \\
& z_{ij} \in \{0, 1\}, & ij \in E, & (6.3) \\
& x_{ij}^{lq} \geq 0 \text{ and integer}, & (i, j, l) \in A^q, q \in Q. & (6.4)
\end{aligned}
$$

6.2.2 Extended formulation for the survivable network design problem with hop constraints and reliable edges.

Now, we need to consider the presence of a group of reliable edges. By definition we call a graph G which is composed of at least two kinds of edges an **heterogeneous graph**. On the opposite, if a graph G is only composed of the same kind of edges (either normal or reliable) we call this graph G an **homogeneous graph**. If the edges are all normal we call the graph G **an homogeneous normal graph** and if the edges are all reliable we call the graph G **an homogeneous reliable graph**. In the following we assume that $E_R \subset E$, in other words, there exist edges which are not reliable, $(E \backslash E_R \neq \emptyset)$ or G is heterogeneous.

Given a graph $G = (V, E)$ where all edges ij in E have an associated installation cost c_{ij} we propose to formulate two variants to integrate reliable edges. The first variant **HopR1** proposes to split the set of edges E in two mutually exclusive subset E_R, the set of reliable edges and $E_{\overline{R}}$ ($E_{\overline{R}} = E \backslash E_R$) the complementary set of normal edges with ($E = E_R \cup E_{\overline{R}}$) and ($E_R \cap E_{\overline{R}} = \emptyset$). In this first approach an edge ij belongs either to the normal set or to the reliable one which could be a limit of the model. That is the reason why a second variant, **HopR2**, considers the case where all edges ij in E are normal ($E_{\overline{R}} = E$) and a subset $E_R \subset E$ contains normal edge ij which could, in some cases, also be used as reliable edge. Figure 6.5 shows the difference between the two variants. For all normal edges ij in the optimal design the associated cost is c_{ij} and for all reliable edge ij used the associated cost is $((K-1)+f) * c_{ij}$ to model a kind of scale economy. At the end of this section we show some properties linking models Hop, HopR1 and HopR2.

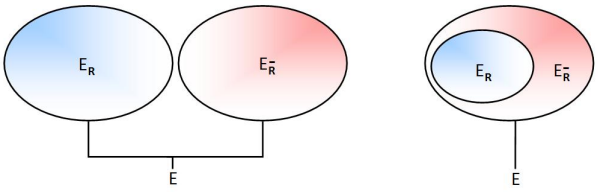


Figure 6.5: Edges sets for variant HopR1 (left) and variant HopR2 (right).

Model HopR1

$$
\begin{aligned}
& \min \quad \sum_{ij \in E} p_{ij} c_{ij} z_{ij} \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1) \in A^q} x_{ji}^{l-1q} - \sum_{j:(i,j,l) \in A^q} x_{ij}^{lq} = \begin{cases} -K & \text{if } (i = o(q)) \\ K & \text{if } (i = d(q)) \text{ and } (l = L+1) \\ 0 & \text{otherwise} \end{cases}, \\
& (HopR1) \qquad i \in V^q, l \in \{1, \ldots, L+1\}, q \in Q, \quad (6.5) \\
& \sum_{l \in \{1,\ldots,L\}} (x_{ij}^{lq} + x_{ji}^{lq}) \leq v_{ij} z_{ij}, \qquad ij \in E, q \in Q, \quad (6.6) \\
& z_{ij} \in \{0, 1\}, \qquad ij \in E, \quad (6.7) \\
& x_{ij}^{lq} \geq 0 \text{ and integer}, \qquad (i, j, l) \in A^q, q \in Q. \quad (6.8)
\end{aligned}
$$

This model is very close to model Hop, variables x_{ij}^{lq} and z_{ij} have the same signification. Constraints (6.1) and (6.5) are the same flow conservation constraints. To ensure K L-paths we send K units of flow from the origin vertice $o(q)$ to the destination vertice $d(q)$. Constraints (6.6) model the edge-disjointness. A new parameter v_{ij} (the capacity of the edge ij) takes the value 1 if the edge ij is a non-reliable edge (if the edge $ij \in E_{\overline{R}}$) and the value K if the edge is reliable (if the edge $ij \in E_R$). c_{ij} is the normal cost of adding an edge ij in the solution. This cost is multiply by a new parameter p_{ij} for each edge. As said in the introduction, if a reliable edge is used in the optimal solution it is normal to pay more for this edge since a reliable structure exists already in the physical layer between the extremities of the reliable edge ensuring the presence of K edge-disjoint paths. For instance p_{ij} takes the value 1 for normal edges and the value $((K-1)+f)$

(where f is a fractional number between 0 and 1) for reliable edges. The structure of this model HopR1 is equivalent to the one of model Hop, consequently properties of model Hop for $L = 2, 3$ and any K, and for $L = 4$ and $K = 2$ (see Chapter 4) for the single commodity case are also valid.

Figure 6.6 shows a sparse graph G composed of 7 nodes and 9 potential edges. The normal installation cost c_{ij} is indicated along each edge. On this network 5 rooted commodities (all commodities have the same origin node, the blue node 21), have to be routed with parameters $L = 3$ and $K = 2$. The five destination nodes (the red nodes) are nodes 3, 4, 7, 11 and 15. First Figure 6.7 represents the optimal topology if we consider that G is an homogeneous normal graph with traditional model Hop. The associated cost is here equal to 262 € (simply the sum of the normal installation costs of all the selected edges).

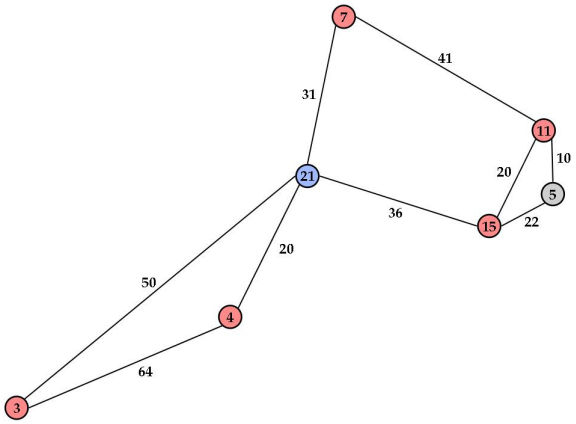


Figure 6.6: Example of a network to connect: 1 source node and 5 destination nodes.

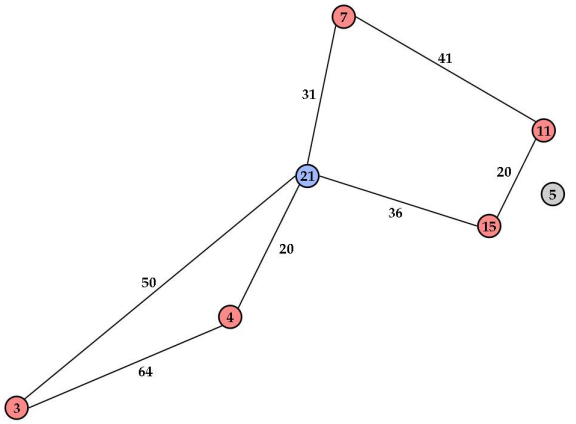


Figure 6.7: Example of optimal topology with model Hop: Optimal design cost: 262 €.

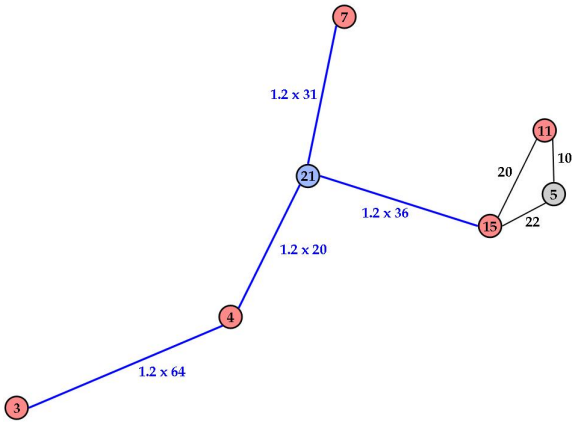


Figure 6.8: Example of optimal topology with model HopR1: Optimal design cost: 216,4 €.

Assume now that a subset of the 9 potential edges are reliable. Here are considered as reliable the edge $3-4$ and all edges which have as one of their extremities the source node 21. It means that 5 edges out of 9 are reliable and have a capacity of K (here $K=2$). We propose to use, as an example, 0.2 as scale economy factor f. It means that if a reliable edge is selected in the optimal topology, his cost is 1.2 times the normal installation cost c_{ij}. Figure 6.8 shows the optimal topology with the model HopR1. Blue bold lines represent reliable edges on which a maximum flow of 2 could be sent for each commodity. The difference between solutions with models Hop and HopR1 is significant in terms of topology first, of cost (262 € versus 216.4 €) secondly and also in terms of simplicity for the network provider.

Model HopR2

$$
\begin{aligned}
& \min \quad \sum_{ij\in E}\sum_{s=1}^{2} p^s c_{ij} z_{ij}^s \\
& \text{s.t.} \quad \sum_{j:(j,i,l-1)\in A^q} x_{ji}^{l-1q} - \sum_{j:(i,j,l)\in A^q} x_{ij}^{lq} = \begin{cases} -K & \text{if} \quad (i=o(q)) \\ K & \text{if} \quad (i=d(q)) \text{ and } (l=L+1) \\ 0 & \text{otherwise} \end{cases}, \\
& (HopR2) \qquad\qquad i\in V^q, l\in\{1,\dots,L+1\}, q\in Q, \qquad (6.9)\\
& \sum_{l\in\{1,\dots,L\}} (x_{ij}^{lq}+x_{ji}^{lq}) \le \sum_{s=1}^{2} v_{ij}^s z_{ij}^s, \qquad ij\in E, q\in Q, \qquad (6.10)\\
& \sum_{s=1}^{2} z_{ij}^s \le 1, \qquad ij\in E, \qquad (6.11)\\
& z_{ij}^s \in \{0,1\}, \qquad ij\in E, s\in\{1,2\}. \qquad (6.12)\\
& x_{ij}^{lq} \quad \ge 0 \text{ and integer}, \qquad (i,j,l)\in A^q, q\in Q. \qquad (6.13)
\end{aligned}
$$

This model is also close to previous models Hop and HopR1 first because variables x_{ij}^{lq} have the same meaning. Nevertheless there is a small modification concerning variable z. Indeed, now we duplicate variables z by adding a new index s. z_{ij}^s are binary variables taking the value 1 if edge ij is used with the scenario s in the optimal topology. We consider here two kinds of scenario ($S=\{1,2\}$), if $z_{ij}^1=1$, edge ij is used as normal

edge (edge's capacity = 1) or if $z^2_{ij} = 1$ edge ij is used as reliable one (edge's capacity = K). Remember that, in this model, all edges are normal edges ($E_{\overline{R}} = E$) and it exists a subset of reliable edges such that ($E_R \subset E_{\overline{R}}$). On one hand, if an edge is used as normal then we have to pay the normal installation cost of this edge ij, c_{ij}. On the other hand, if an edge is used as reliable then we have to pay $((K-1)+f)$ times the normal installation cost c_{ij}. We add thus a new parameter p^s which represents the coefficient depending on the scenario by which we multiply the normal installation cost. By definition, $p^1 = 1$ and $p^2 = ((K-1)+f)$. Then all constraints are similar to previous models except new constraints (6.11) which force that if an edge is used, the two scenario are mutually exclusive, either it is as normal or as reliable edge that the edge is used. The edge-disjointness constraints (6.10) are also modified, parameter v^s_{ij} is the capacity of edge ij associated with scenario s. For all edges, $v^1_{ij} = 1$ because all edges exist at least in the normal version, then if an edge $ij \in E_R$, $v^2_{ij} = K$ else $v^2_{ij} = 1$. Note that for all $ij \notin E_R$ capacities of both scenario are similar ($v^1_{ij} = v^2_{ij} = 1$) but not the price: scenario 1 is thus always preferred by the model in terms of costs.

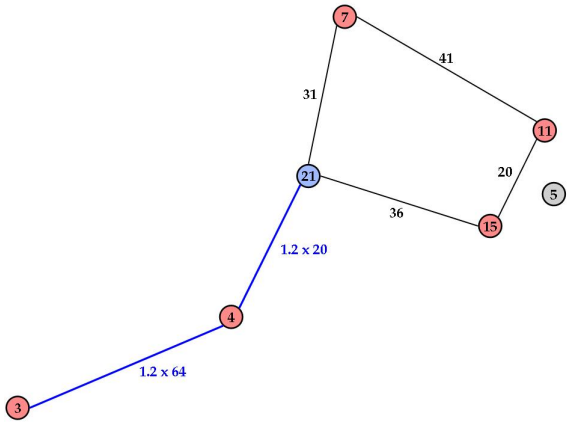


Figure 6.9: Example of optimal topology with model HopR2: Optimal design cost: 212 €.

Figure 6.9 presents the HopR2 optimal version of previous example. Here the 5 edges that were previously considered as reliable (the edges $21-3$, $21-4$, $21-7$, $21-15$ and $3-4$) can either be used as normal or as reliable edges. We see, as an illustration, that edges $21-4$ and $3-4$ are used in their reliable version as edges $21-7$ and $21-15$ are used in their normal version. We show a positive effect linked to the utilization of reliable edges. Indeed model HopR1 proposes a first significant improvement in terms of costs (216,4 € compared to 262 € for normal Hop model) of approximately 18%. Model HopR2 goes further and improves still the optimal cost (212 € which is around 20% of improvement). A deeper analysis of the cost effect linked to the utilization or not of reliable edges is presented in section 6.6.

6.2.3 Properties of the reliable models HopR1 and HopR2

As said before, we assume that $E_R \subset E$, in other words, there exist edges which are not reliable, ($E \backslash E_R \neq \emptyset$) or G is heterogeneous. Indeed, consider the following homogeneous reliable graph $G_R = (V, E_R)$ where all edges are reliable. Then for each commodity

and in terms of design, looking for K edge-disjoint paths in G_R is equivalent to looking for only one single path in the associated homogeneous normal graph G defined as G_R, without edges being reliable.

Proposition 5. *The optimal solution of the survivable network design problem which looks for K edge-disjoint paths in an homogeneous reliable graph $G_R = (V, E_R)$ where all edges $ij \in E_R$ are reliable edges is always composed of one single path for each commodity.*

Proof. As in an homogeneous reliable graph G_R all edges are considered as reliable, the capacity of all these edges is fixed to K. Then the capacity constraints (6.6) or (6.10) are redundant and the problems (HopR1 or HopR2) become similar to an uncapacitated multicommodity network flow problem for which it is well known that the optimal solution consists in sending, for each commodity, the total flow K on the shortest path. □

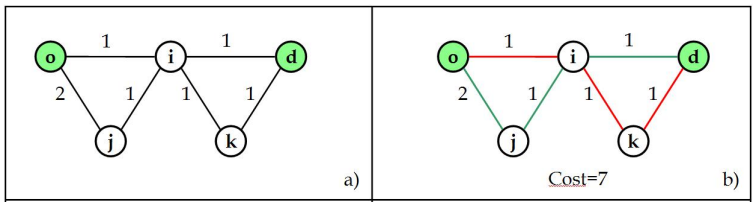


Figure 6.10: An homogeneous normal network G and the optimal solution with $L = 3$ and $K = 2$.

Figures 6.10 and 6.11 illustrate this proposition. Indeed Figure 6.10 a.) shows an homogeneous normal graph G in which we want to find 2 edge-disjoint paths with maximum 3 hops between source node o and destination node d. Figure 6.10 b.) depicts the optimal solution (with a total cost $= 7$) which consists in the two paths $(o - i - k - d)$ (with cost $= 3$) and $(o - j - i - d)$ (with cost $= 4$). Note that for this example, if we use the shortest path $(o - i - d)$, no feasible solution exists.

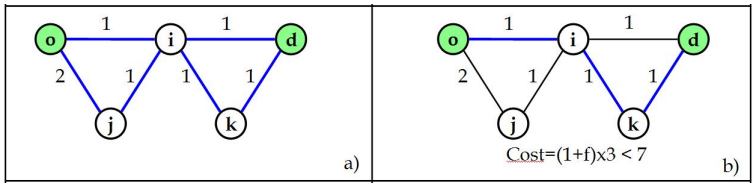


Figure 6.11: An homogeneous reliable network G_R and a feasible solution with $L = 3$ and $K = 2$.

Consider now the homogeneous reliable graph G_R represented on Figure 6.11 a.). For this graph there exist four feasible solutions. One way to build a group of feasible solutions consists in sending a flow K on only one path. As a good upper bound we can take the optimal solution of the basis problem with the homogeneous normal graph G which consists of the less expensive pair of paths $(o - i - k - d)$ and $(o - j - i - d)$ and we send a flow K on the shortest of these two paths (here $(o - i - k - d)$). This good feasible

solution is presented on Figure 6.11 b.) but the better solution (see Figure 6.12) consists in sending a flow K on the shortest path $(o - i - d)$ (cost=$(1 + f) \times 2$).

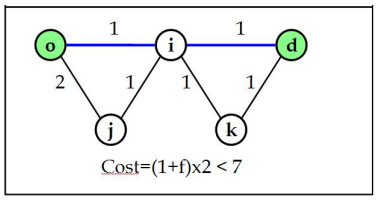


Figure 6.12: Optimal solution for the homogeneous reliable network G_R with $L = 3$ and $K = 2$.

As stated previously, the three models are linked together. Indeed it is possible to obtain models Hop and HopR1 from the relaxed model HopR2. Let $\mathcal{Z}_{Hop}$ (respectively $\mathcal{Z}_{HopR1}$, $\mathcal{Z}_{HopR2}$) be the set of all feasible solutions of model Hop (respectively HopR1 and HopR2).

Proposition 6. *Model HopR2 is a relaxation of model Hop and* $\mathcal{Z}_{Hop} \subset \mathcal{Z}_{HopR2}$

Proof. Model Hop is similar to Model HopR2 after adding this new group of constraints:

$$z_{ij}^2 = 0, ij \in E. \tag{6.14}$$

□

Proposition 7. *Model HopR2 is a relaxation of model HopR1 and* $\mathcal{Z}_{HopR1} \subset \mathcal{Z}_{HopR2}$

Proof. Model HopR1 is similar to Model HopR2 after adding this new group of constraints:

$$z_{ij}^1 = 0, ij \in E_R. \tag{6.15}$$

□

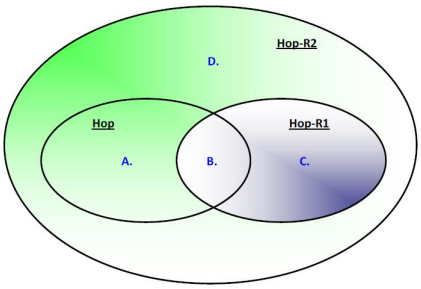


Figure 6.13: Feasible solutions subsets depending of the model.

Figure 6.13 shows the three sets of feasible solutions depending of the models. As an illustration we can consider the simple example of the graph depicted on Figure 6.14. This graph is composed of 6 nodes, one origin node o and one destination node d, a set of 10 non-reliable edges (in black) and a set of three bivalent edges (these edges can be used as reliable or not depending on the model) in blue. For $L = 4$ and $K = 2$, four points (A,B,C,D) in Figure 6.13 describe four cases of feasible solutions detailed on Figure 6.15.

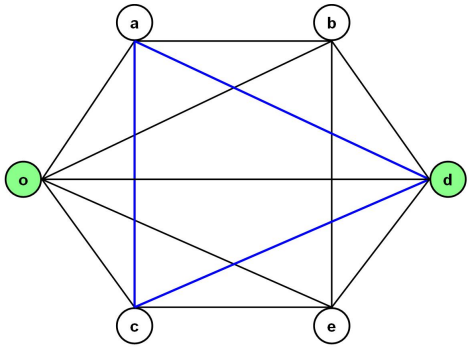


Figure 6.14: Example of a network to design.

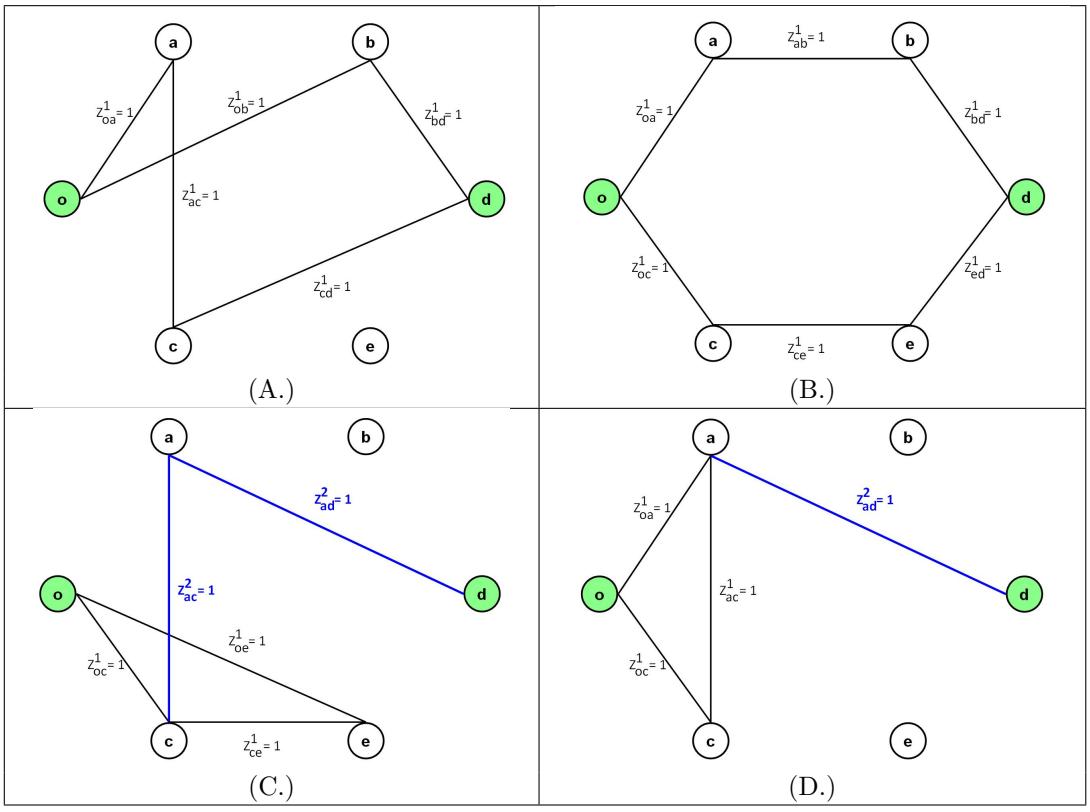


Figure 6.15: Illustration of feasible solutions for each model.

All four sub-figures of Figure 6.15 are feasible for model HopR2. Figure 6.15 (A.) shows a feasible solution for model Hop which is not feasible for model HopR1 because potential reliable edge ac is used as normal edge ($z^1_{ac} = 1$) which is not possible in model HopR1 because ($z^1_{ij} = 0, ij \in E_R$). On Figure 6.15 (B.) we find a solution which is feasible for both models Hop and HopR1. Figure 6.15 (C.) represents a feasible solution for model HopR1 but not feasible for model Hop because it uses reliable edges ac and ad. Finally, Figure 6.15 (D.) depicts a feasible solution only for model HopR2, it is not a feasible solution for model Hop because reliable edges are used and it is not a feasible solution for model HopR1 because edge ac which is a bivalent edge is used in a normal way.

6.3 Benders reformulation

For the same reasons stated before in the Benders reformulation subsection of Chapter 5, we propose to use an alternative formulation of our problems (HopR1 and HopR2) by projecting out the flow variables x^{lq}_{ij}. The remaining problem called master problem is thus only dependent of design variables z. Here, once again, when we project out the flow variables, the subproblems become independent linear programs for each commodity leading to $SP(q, \overline{z})$, thus reducing significantly the size of the linear programs to solve.

Here again, because flow variables are integer and because it is not allowed to project out integer variables we propose to get round this difficulty by introducing new formulations **HopR1′** and **HopR2′** , where we relax the integrality restrictions on x variables in HopR1 and HopR2, replacing (6.8) and (6.13) by

$$x^{lq}_{ij} \geq 0, \qquad (i, j, l) \in A^q, q \in Q. \tag{6.16}$$

We discuss in Section 6.5 whether HopR1′ and HopR2′ provide the same optimal design $\overline{z}$ as HopR1 and HopR2. Master problems of models HopR1 and HopR2 are slightly different because in both models design variables z have not exactly the same meaning. But in terms of subproblems the structure of both models is similar. Indeed the goal of subproblems is to ensure that, for a given design vector $\overline{z}$ (for a given graph), the feasibility problem which consists in finding K edge-disjoint paths with maximum L hops is not violated. And except for small details this problem is similar for both reliable models.

6.3.1 Master problem of model HopR1

$$(BR - HopR1') \qquad \begin{array}{lll} \min & \displaystyle\sum_{ij \in E} p_{ij} c_{ij} z_{ij} & \\ \text{s.t.} & K\overline{\pi}^{L+1}_{d(q)} - \displaystyle\sum_{ij \in E} \overline{\sigma}_{ij} v_{ij} z_{ij} \leq 0, & (\overline{\pi}, \overline{\sigma}) \in \displaystyle\bigcup_{q \in Q} \mathcal{R}^q, \\ & z_{ij} \in \{0, 1\}, & ij \in E, \end{array}$$

6.3.2 Master problem of model HopR2

$$
(BR-HopR2') \qquad
\begin{array}{lll}
\min & \displaystyle\sum_{ij\in E}\sum_{s=1}^{2} p^s c_{ij} z_{ij}^s & \\
\text{s.t.} & K\overline{\pi}_{d(q)}^{L+1} - \displaystyle\sum_{ij\in E}\sum_{s=1}^{2} \overline{\sigma}_{ij} v_{ij}^s z_{ij}^s \leq 0, & (\overline{\pi},\overline{\sigma}) \in \displaystyle\bigcup_{q\in Q} \mathcal{R}^q, \\
& \sum_{s=1}^{2} z_{ij}^s \leq 1, & ij \in E, \\
& z_{ij} \in \{0,1\}, & ij \in E,
\end{array}
$$

For both master problems $\mathcal{R}^q$ contains the extreme points of the feasibility polyhedron for the dual subproblem $SP(q,\overline{z})$ described next.

6.3.3 Subproblem of model HopR1 and HopR2

Given commodity $q \in Q$, let us introduce a dual variable π_i^l, associated with node $i \in V$ and layer l, for each constraint (6.5) or (6.9) and a dual variable σ_{ij} for each constraint (6.6) or (6.10). Defining $o := o(q)$ and $d := d(q)$, and adding the constraint $\pi_d^{L+1} \leq 1$ to normalize the dual cone we get our dual subproblem $SP(q,\overline{z})$

$$
SP(q,\overline{z}) \qquad
\begin{array}{lll}
\max & \mathit{objective\ function} & (6.17) \\
\text{s.t.} & \pi_i^2 - \pi_o^1 - \sigma_{oi} \leq 0, & i \in V, \\
& \pi_i^{l+1} - \pi_j^l - \sigma_{ij} \leq 0, & i,j \in V\backslash\{o,d\}, i<j, l \in \{2,\ldots,L\}, \\
& \pi_j^{l+1} - \pi_i^l - \sigma_{ij} \leq 0, & i,j \in V\backslash\{o,d\}, i<j, l \in \{2,\ldots,L\}, \\
& \pi_d^{l+1} - \pi_i^l - \sigma_{id} \leq 0, & i \in V\backslash\{o,d\}, l \in \{2,\ldots,L\}, \\
& \pi_d^{l+1} - \pi_d^l \leq 0, & l \in \{2,\ldots,L\}, \\
& \pi_d^{L+1} \leq 1, & \\
& \sigma_{ij} \geq 0, & ij \in E.
\end{array}
$$

Of course the objective function of subproblems linked to each of our two reliable variants is different. For model HopR1 we replace (6.17) by:

$$
\max \qquad K\pi_d^{L+1} - \sum_{ij\in E} v_{ij}\overline{z}_{ij}\sigma_{ij} \tag{6.18}
$$

For model HopR2, we replace (6.17) by:

$$
\max \qquad K\pi_d^{L+1} - \sum_{ij\in E}\sum_{s=1}^{2} v_{ij}^s \overline{z}_{ij}^s \sigma_{ij} \tag{6.19}
$$

For both models and for a particular commodity $q \in Q$, given a vector $\overline{z}$ defining a graph, if it is possible to find a feasible flow in terms of continuous x variables leading to K edge-disjoint paths composed of maximum L hops, then the value of the objective function of $SP(q,\overline{z})$ is equal to 0. Otherwise, a new Benders cut is added to the adequate master problem ((BR-HopR1$'$) or (BR-HopR2$'$)). Depending on the nature of the problem the Benders cuts are slightly different. For model HopR1, Benders cuts are:

$$K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E} \overline{\sigma}_{ij} v_{ij} z_{ij} \leq 0 \tag{6.20}$$

and for model HopR2, Benders cuts look like:

$$K\overline{\pi}_{d(q)}^{L+1} - \sum_{ij \in E} \sum_{s=1}^{2} \overline{\sigma}_{ij} v_{ij}^{s} z_{ij}^{s} \leq 0 \tag{6.21}$$

6.4 Algorithms

Even if Benders reformulation is nice from a theoretical point of view, it is not the only motivation for using this technique in this research area. Benders reformulation is, as said before, useful here because the three extended models (Hop, HopR1 and HopR2) are composed with two groups of variables among with one (x variables) is big. A first natural approach consists in solving this kind of huge extended models with a commercial solver like CPLEX. But the big number of variables and constraints can have a negative effect on the performance (CPU time) of this kind of techniques. That is the reason why a decomposition like the Benders one can help and improve the performance. In the next section we develop and explain some algorithms based on this reformulation.

A second approach consists in using cutting plane techniques. The main idea is to solve, in an alternative manner, and at each iteration, the master problem and the subproblems for each commodity $q \in Q$. At each iteration firstly a new branch-and-bound tree is investigate to solve the master problem. Then the solution of the master problem in terms of design variables z gives new parameters $\overline{z}$ for the different subproblems. For each commodity $q \in Q$ a new subproblem is generated and solved to optimality using the new parameters from the master problem. Depending on the optimal solution of each subproblem, a new cut is generated or not and introduced in the master problem for the next iteration. This process stops when the optimal solution of the master problem's branch-and-bound tree is a vector of z variables for which no more Benders cut is added[1].

This cutting plane approach could converge slowly because a new branch-and-bound tree has to be explored at each iteration. This exploration of the tree is time consuming that is the reason why we developed, in Chapter 5, branch-and-cut algorithms based also on the Benders decomposition. The main idea of branch-and-cut techniques is to explore only one branch-and-bound tree and to add Benders cuts inside the different nodes of the tree. We have already deeply investigated this algorithmic aspect before. The goal of this section is not to do the same work again. We just mention here that, for this chapter, we decide to use the fastest branch-and-cut algorithm developed before for the experiment phase.

[1]This stop criterion is not completely correct. Indeed if the vector z do not violate any Benders cut, it means that there exists a continuous vector x satisfying flow conservation constraints, edge-disjointness constraints,... but not necessary an integer one. However all the optimal solutions obtained with extended model Hop, HopR1 and HopR2 are integer solutions in terms of z but also in terms of x variables. This remark will be developed in the following section.

Algorithm 4: Hybrid branch-and-cut algorithm: `bc-n`

```
begin                                                    /* Initialization */
    T = {o} where o has no branching constraints;
    UB = +∞;
while T is nonempty do
    select a node o' ∈ T;
    T ← T\{o'};                          /* withdraw node o' from the tree */
    solve o';
    let z̄ be an optimal solution;
    let w̄ be the optimal cost;
    if w̄ < UB then
        if z̄ ∈ {0,1}^|E| or depth(o') ≤ n then
            foreach q ∈ Q do compute s_q = SP(q, z̄);
            if s_q > 0 then add (6.20) or (6.21) to (MP);
        if z̄ ∈ {0,1}^|E| and s_q ≤ 0 for each q ∈ Q then
            foreach q ∈ Q do compute f_q = FP(q, z̄);
            if f_q > 0 for some q ∈ Q then add (5.8) to (MP);
            else
                UB ← w̄;                          /* define a new upper bound */
                Z* ← z̄;                            /* save current incumbent */
        if s_q > 0 or f_q > 0 for some q ∈ Q then
            T ← T ∪ {o'};                    /* put node o' back in the tree */
        else if z̄ ∉ {0,1}^|E| then
            branch, resulting in nodes o* and o**;
            T ← T ∪ {o*, o**};                          /* add children to the tree */
return Z*
```

In a basic branch-and-cut algorithm, the main idea is to add cuts (here Benders cuts) at each node of the branch-and-bound tree. This basic method is time consuming because lots of cuts are added at each node. Here we propose to use an improved branch-and-cut algorithm called `bc-5-heur` (see Chapter 2.4) which add Benders cuts only at "strategic points" of the exploration tree. By "strategic points" we mean either "when the optimal solution of a node n is integer" (it means that this solution could be a candidate to optimality) or "when the depth of the node n is less or equal to a given parameter". By definition, in a branch-and-bound tree, the depth of the root node is equal to 0 and the depth of a given node n is equal to the depth of its parent node + 1. In others words, the depth of a given node n represents the number of branching constraints added to the linear relaxation problem (the root node) to obtain the subproblem linked to node n. While the depth of a given node is less or equal to a given parameter we propose to add lots of Benders cuts, then we let CPLEX freely branch.

Furthermore, we can speed up this algorithm by introducing a good upper bound which can be easily found by the heuristic described in Chapter 5. The main idea of this heuristic is based on the following statement that the solution of the root node can be computed very quickly. This solution is composed of a big number of design z variables equal to 0 (let those variables be z_1) and a small number of z variables > 0 (let those

variables be z_2). We build and solve a modified reduced problem in which all $z_1 = 0$ and $z_2 \in \{0, 1\}$. Because this new problem is largely reduced we can expect to obtain quickly its optimal solution. This optimal solution is a solution of a smaller feasible set, and consequently, is an upper bound of our original problem.

6.5 Feasibility problem

Given $\overline{z} \in \{0,1\}^{|E|}$, let $\mathcal{X}_i(\overline{z})$ be the set of binary vectors defined by (6.5) (respectively (6.9)),(6.6) (respectively (6.10)) and (6.8) (respectively (6.13)) and $\mathcal{X}_c(\overline{z})$ the polyhedron defined by (6.5) (respectively (6.9)),(6.6) (respectively (6.10)) and (6.16). Then, define $\mathcal{Z}_i$ (respectively $\mathcal{Z}_c$) as the set of vectors $\overline{z} \in \{0,1\}^{|E|}$ such that $\mathcal{X}_i(\overline{z})$ (respectively $\mathcal{X}_c(\overline{z})$) is nonempty, and let $conv(\mathcal{Z}_i)$ (respectively $conv(\mathcal{Z}_c)$) be its convex hull.

As said before, in this chapter, because the edges can be normal and\or reliable, we call, by definition, an homogeneous graph G, a graph only composed of one variety of edge (either normal or reliable). An heterogeneous graph G is thus a graph in which both varieties are present.

Proposition 1 from Chapter 5 prove that: *The sets $\mathcal{Z}_c$ and $\mathcal{Z}_i$ are equal for $L = 2, 3$ with any $K \geq 2$, and $L = 4$ with $K = 2$.* This proposition is similar to say that given a graph G (given a binary vector $\overline{z}$ which represents the graph G), it exists a fractional routing in G if and only if it exists an integer routing in G for adequate values of L and K. This proposition is true for a normal and homogeneous graph G without reliable edges. But here models HopR1 and HopR2 leads to optimal design which can be either homogeneous or heterogeneous. Thus a natural question now is to know if this proposition holds for heterogeneous graphs.

Proposition 8. *Given a binary vector $\overline{z} \in \{0,1\}^{|E|}$, which represents an heterogeneous graph G where some edges $ij \in E_{\overline{R}}$ and some edges $ij \in E_R$, there exists a fractional routing in G if and only if there exists an integer routing in G for $L = 2, 3$ with any $K \geq 2$, and $L = 4$ with $K = 2$.*

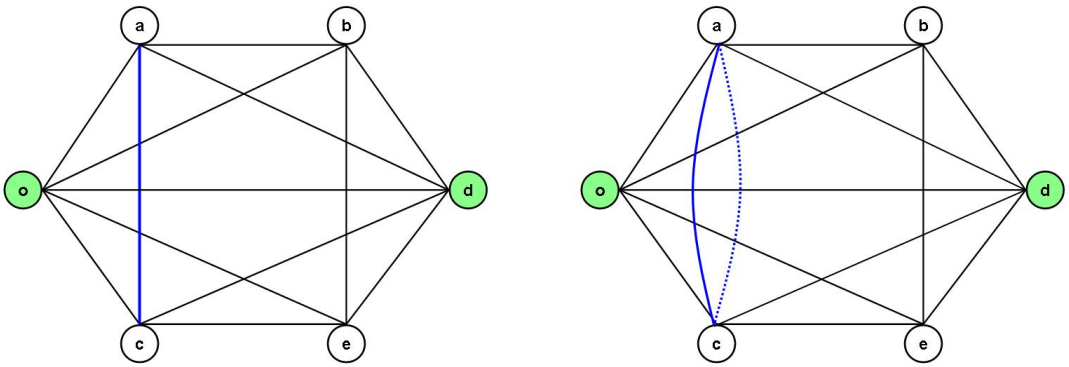
Proof.

- ⋄ If there exists an integer routing in G then there exists a fractional routing in G.

  Since $\mathcal{X}_i(\overline{z}) \subseteq \mathcal{X}_c(\overline{z})$ for every binary $\overline{z}$, we have that $\mathcal{Z}_i \subseteq \mathcal{Z}_c$.

- ⋄ If there exists a fractional routing in G then there exists an integer routing in G.

  To prove this property we need to introduce a new representation of our heterogeneous graph G. From an heterogeneous graph $G = (V, E)$, where $E = (E_{\overline{R}} \cup E_R)$ we create an alternative graph $G' = (V, E')$ where $E' = (E_{\overline{R}} \cup E_{R'})$ and $E_{R'} = \{ij_k, ij \in E_R, k \in \{1, 2, ..., K\}\}$. The idea is to separate a particular reliable edge ij with a capacity $= K$ in K different normal edges ij_k with each a capacity $= 1$. G' is thus a graph only composed of normal edges and consequently G' is homogeneous.



(a.) (b.)

Figure 6.16: Representation of graph G (a.) and graph $G^{'}$ (b.) with one reliable edge ac.

Figure 6.16 (a.) represents an heterogeneous graph G with only one reliable edge ac and the alternative homogeneous representation of G, G' (see Figure 6.16 (b.)) with exactly K (here 2) copies ac_1 and ac_2 between nodes a and c. Note that from G' it is easily possible to build the layered associated graph $G^{''}$ (see for instance Figure 6.17).

- If there exists a fractional routing in G then there exists a fractional routing in G'.

  Because:

$$x_{ij}^{lq} = \sum_{k=1}^{K} x_{ij_k}^{lq} \tag{6.22}$$

$$z_{ij} = \sum_{k=1}^{K} z_{ij_k} \tag{6.23}$$

  it is trivial to prove this properties by substitution.

- If there exists a fractional routing in G' then there exists an integer routing in G'.

  This property is always true thanks to Proposition 1 from Chapter 5 and also because Theorem 2.2 from [63] (and its generalization to any K in [30] and [34]) and Theorem 3 from [62] are also valid for graphs with parallel edges (here the case of the built homogeneous graph G').

- If there exists an integer routing in G' then there exists an integer routing in G.

  This property is also trivial. Indeed if for each reliable edge ij we add all the integer flows of the K different copies ij_k, we obtain of course a number which is also integer. If x_{ijk}^{lq} is integer then x_{ij}^{lq} is also integer.

$$\sum_{k=1}^{K} x_{ijk}^{lq} = x_{ij}^{lq} \tag{6.24}$$

$$\sum_{k=1}^{K} z_{ijk} = z_{ij} \tag{6.25}$$

□

For $L = 4$ and $K > 2$ or for $L > 4$, this proposition is not always satisfied. We propose, as it was already done in previous chapter, to add a feasibility model in the branch-and-cut algorithm `bc-5-heur` to check if the optimal design given by the branch-and-cut process (which satisfies the fractional routing existence condition) admits also an integer routing. This feasibility model is almost similar to previous $FP(q, \overline{z})$ with the exception of constraints (**??**) which is replaced respectively by constraints (6.6) for model HopR1 and (6.10) for model HopR2.

If the objective value of the feasibility problem $e = 0$, $\overline{z} \in \mathcal{Z}_i$. Otherwise, we must add the weak inequality

$$\sum_{ij \in E^0(\overline{z})} z_{ij} \geq 1, \tag{6.26}$$

with $E^0(\overline{z}) = \{ij \in E \text{ s.t. } \overline{z}_{ij} = 0\}$, to move away from the current solution, as explained in the next subsection. It is interesting to point out that in our computational experiments, we never needed to add such a cut.

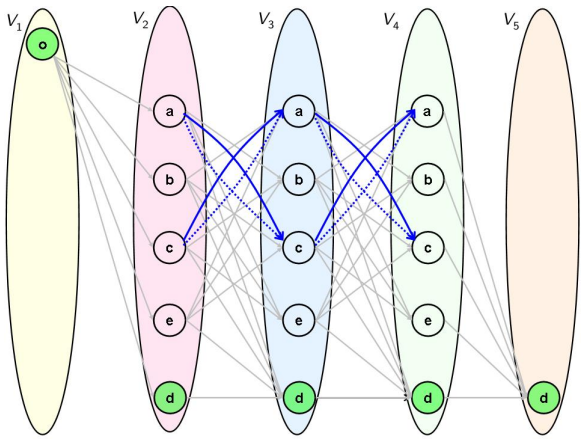


Figure 6.17: Representation of associated layered graph G''.

6.6 Computational experiments

In this section we test our two extended models HopR1 and HopR2 first directly used with CPLEX and then with adapted branch-and-cut algorithms `bc-5-heurR1` and `bc-5-heurR2`. The objective of this section is twofold, first we want to study how the problem becomes more and more complex with the addition of a subset of reliable edges. This first aspect will be studied by looking at the gap between the linear relaxation and the optimal solution but also by looking at the CPU time needed to reach optimality. We study also how fast our branch-and-cut algorithm (also developed in previous chapter) is.

The second objective of this section is also to analyze the potential economical impact of reliable edges for a network manager. Is it really interesting from a managerial as an economical point of view to introduce reliable edges in network design? What could be a good estimation of the economical improvement linked to reliable edges?...

6.6.1 Implementation details

All models have been coded in JAVA using CPLEX 11 MIP solver and run on a DELL Latitude D820 with a processor Intel Core Duo 2 T7200 of 2GHz and 2.5 GB of RAM memory. We allow CPLEX to store the branch-and-bound tree in a file, setting parameter *IntParam.NodeFileInd* to 2, to avoid from running out of memory. Moreover, for each algorithm we configure CPLEX as follows :

> `bc-n-heur:` We use the algorithm `bc-n`, providing CPLEX with the upper bound found by `heuristic`. The CPU times reported do not consider the time spent in `heuristic`.
>
> `heuristic:` We first solve the linear programming relaxation by a cutting plane algorithm (in fact, we use a branch-and-cut algorithm with a limit of 0 node, setting *IntParam.NodeLim* to 0). Then, we fix some of the variables to 0, and resolve the resulting problem with `bc-n`.

6.6.2 Instances details

We used two different test sets. The set TC is taken from a class of complete graphs $G = (V, E)$, reported in [51]. They share the following features: $|V| = 21$, $|Q| \in \{5, 10\}$, and all point-to-point demands share one of their extremities (which we call rooted demands in Table 6.1). The cost matrix for each instance considers the integer part of the Euclidean distance between the coordinates of the 21 nodes, randomly placed among the integer points of a grid (100×100). The TC class contains 5 instances with 5 commodities and 5 instances with 10 commodities with the root located in the center of the grid. Then, five instances are based on sparse networks from *SNDlib* [89]: pdh, di-yuan, dfn-gwin, polska, nobel-us. Table 6.1 summarizes the characteristics of the instances. We solved the problem for L ranging from 3 to 5 and K from 2 to 3. We set a time limit of 3600 seconds for all instances and algorithms.

Moreover, three additional parameters are used in this testing phase. First how can we model the scale economy associated to reliable edges. As said before, if the optimal topology uses a reliable edge ij, we propose to multiply the normal installation cost of

Name	$\|N\|$	$\|E\|$	$\|Q\|$	# of instances	Rooted demands?
TC-5	21	210	5	5	true
TC-10	21	210	10	5	true
pdh	11	34	27	1	false
di-yuan	11	42	48	1	false
dfn-gwin	11	47	9	1	false
polska	12	18	17	1	false
nobel-us	14	21	33	1	false

Table 6.1: Instances description.

the edge ij by $((K-1)+f)$, where f is a fractional number between 0 and 1. In this computational study we decide to test different values of the parameter $f \in \{0.2, 0.4, 0.6, 0.8\}$ to see how the optimal design is sensitive to this scale economy parameter.

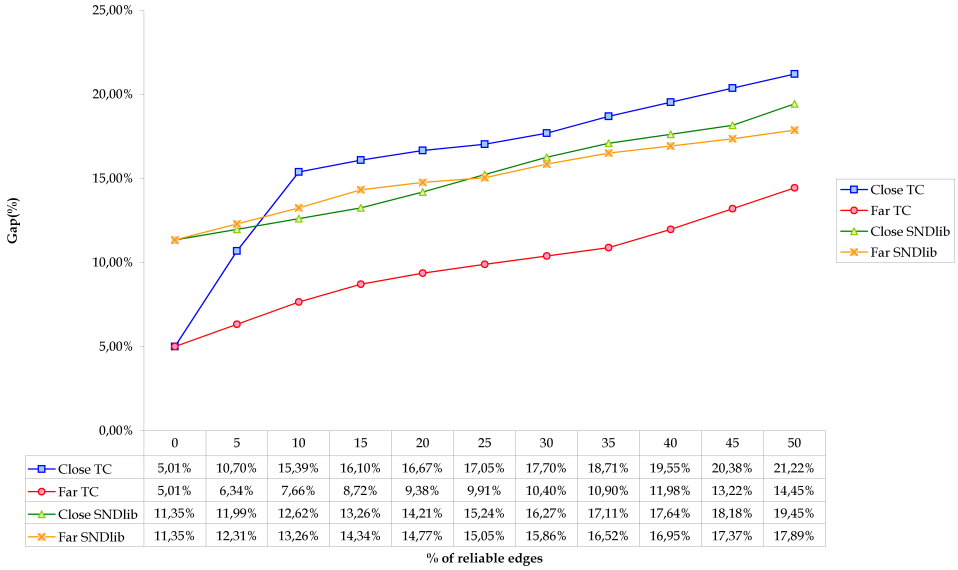
The second additional parameter is linked to the proportion of reliable edges in the whole edge set E. Note that if the proportion of reliable edges is equal to 0, this problem is equivalent to the problem studied in the previous chapter.

Last but not least, the third additional parameter is linked to the way we used to decide if an edge $ij \in E$ is potentially reliable or not. We propose to use two different processes called **Close** and **Far**. The Close process depends of two parameters cg and c. cg is one node of the network which is considered arbitrarily as a center of gravity. And if $c = 2$ it means that we create a subset of $(c+1)$ nodes C composed of cg and of 2 additional nodes which are the two closest nodes from the center of gravity node cg. All edges $ij \in E_R \Leftrightarrow i \in C$ or $j \in C$. We do tests for different values of $c \in \{0, 1, 2, 3\}$. In other words, the Close process sets as reliable roughly edges which are close to a center of gravity. On the opposite the second process, called Far, is also composed of two parameters cg and f. cg is also a center of gravity and if $f = 2$, for instance, it means that we create a subset of f nodes F composed of the 2 nodes which are the two farther nodes from the center of gravity node cg. All edges $ij \in E_R \Leftrightarrow i \in F$ and $j \in F$. Note that set F forms a clique. In other words are considered as reliable all edges ij belonging to the farther clique (composed of c nodes) from a center of gravity cg.

6.6.3 Complexity of the problem and quality of Hop models

We know that the problem is strongly NP-hard. Although the extended model Hop is really easy to use as starting point in order to model the survivable network design with hop constraints and reliable edges problem[2], we need now to study the difficulty of solving this new problem. Our first criterion is the root gap (expressed in %) which represents the relative difference between the optimal linear relaxation solution and the optimal integer solution. This gap is obtained by using the following formula: $\frac{(IP^*-LR^*)}{IP^*}$. Figure 6.18 depicts the upward trend of the gap when the proportion of reliable edges in the network increases. Indeed without any reliable edges, we showed in the previous chapter that the extended model Hop integrates well the edge-disjointness constraints because even if the gap is high for $K = 1$ (around 25%), for $K = 2, 3$ the average gap

[2]Indeed only simple and localized modifications have to be introduced



	0	5	10	15	20	25	30	35	40	45	50
Close TC	5,01%	10,70%	15,39%	16,10%	16,67%	17,05%	17,70%	18,71%	19,55%	20,38%	21,22%
Far TC	5,01%	6,34%	7,66%	8,72%	9,38%	9,91%	10,40%	10,90%	11,98%	13,22%	14,45%
Close SNDlib	11,35%	11,99%	12,62%	13,26%	14,21%	15,24%	16,27%	17,11%	17,64%	18,18%	19,45%
Far SNDlib	11,35%	12,31%	13,26%	14,34%	14,77%	15,05%	15,86%	16,52%	16,95%	17,37%	17,89%

Figure 6.18: Evolution of the gap (%) in function of the percentage of reliable edges.

decreases to reach approximately the threshold of 5% for the TC test set.

On Figure 6.18 we show clearly that as the proportion of reliable edges increases (for both kind of network TC and *SNDlib* and for both reliable process Close and Far) the gap increases also.

Models	Average CPU time (sec.)	Not solved to optimality	Average final gap (%)
Hop	214.67	0 \ 18	0
bc-5-heur	5.73	0 \ 18	0
HopR1	262.41	0 \ 660	0
bc-5-heurR1	4.95	0 \ 660	0
HopR2	496.53	22 \ 660	5.51
bc-5-heurR2	31.96	0 \ 660	0

Table 6.2: Average CPU time, problem not solved and final gap for *SNDlib* test set.

Tables 6.3, 6.4 and 6.5 show that here again the gap increase between the original model Hop and the reliable models HopR1 and HopR2.

Instance		Models		
Name	$\|Q\|$	Hop	HopR1	HopR2
TC	5	3.22	12.54	9.27
	10	6.80	18.84	13.30
pdh	27	12.54	27.91	18.70
di-yuan	48	13.62	28.04	18.49
dfn-gwin	9	9.90	21.17	14.36
polska	11	3.18	9.90	6.43
Average gap		6.39	17.92	12.56

Table 6.3: Average gap (%) depending of the model and the number of commodities.

K	Models		
	Hop	HopR1	HopR2
2	7.63	17.86	14.17
3	4.94	17.98	10.78
Average gap	6.39	17.92	12.56

Table 6.4: Average gap (%) depending on the number of edge-disjoint paths K.

L	Models		
	Hop	HopR1	HopR2
3	7.48	18.93	13.37
4	6.58	17.76	12.57
5	5.19	17.10	11.77
Average gap	6.39	17.92	12.56

Table 6.5: Average gap (%) depending on the hop limit L.

About the analysis of the CPU time, we did tests only on the *SNDlib* test set. Table 6.2 shows that basically models `Hop` and `HopR1` take on average the same amount of time to reach optimality. Of course, here again the associated branch-and-cut algorithms `bc-5-heur` and `bc-5-heurR1` outperform the standard extended formulations. The improvement is really significant and reached ratios between extended and branch-and-cut around 40, meaning that the branch-and-cut algorithms are 40 times faster than the extended formulations with CPLEX. A big difference concerns model `HopR2` which seems more difficult to solve for both the extended formulation with CPLEX and the branch-and-cut algorithm. Roughly for the extended formulation, the average CPU time to reach optimality is more that twice as big as the one for models `extended` and `HopR1`. Concerning the branch-and-cut algorithms here again the average time increases considerably. But the ratio between extended formulation and branch-and-cut stays reasonably correct and turns around 16. The HopR2 formulation seems really to be more complicated also because 22 instances out of 660 are not solved to optimality by the associated extended formulation during the 3600 seconds time window and an average gap of approximately 5% remains to close between the bests lower and upper bound of the branch-and-bound tree. Fortunately all these 22 instances are easily solved to optimality by our branch-and-cut `bc-5-heurR2`. This huge difference in term in CPU time performance is illustrated on the performance profile graph depicted onf Figure 6.19.

We remark also that for sparse graphs, as for instance the graphs from the *SNDlib* test set, and for some values of the parameters L and K, no feasible solution exists for model Hop. Of course, this problem does not occur when we consider the graphs from the TC test set because these are all complete graphs. Thus, for a given sparse graph G and for some values of L and K, it happens that when no feasible solution exists for model Hop, introducing reliable edges, can make the problem feasible.

On Table 6.6 we show that the Close process, used to determine which edges are reliable or not, induces a higher average gap for both models HopR1 and HopR2 than the Far process. We can understand this property because with the Far process, the reliable edge clique defined is difficult to reach, thus the probability to use a reliable edge in

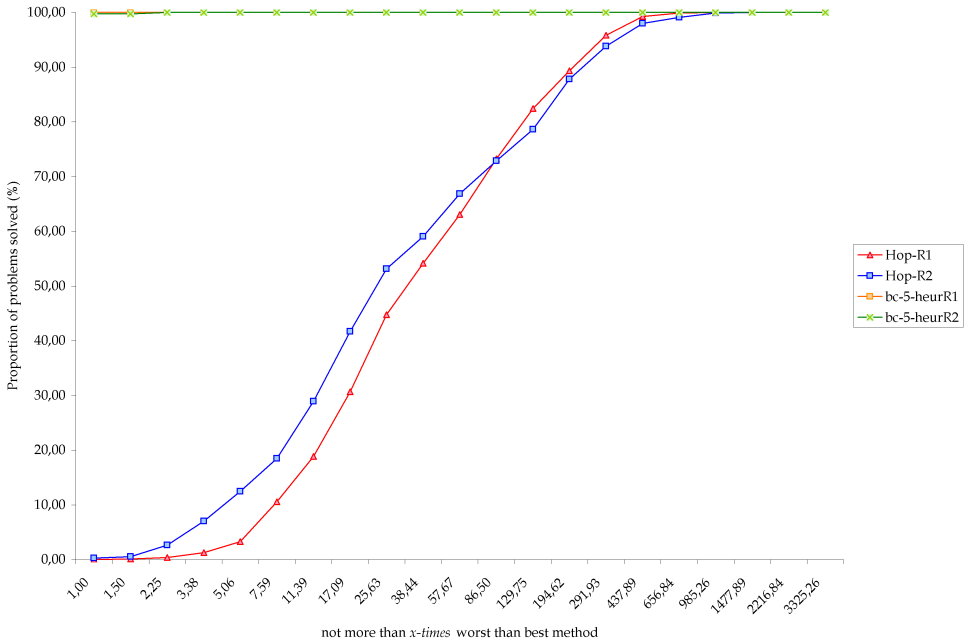


Figure 6.19: Performance profile comparing methods on the *SNDlib* test set.

the optimal design is low because by definition reliable edges are far, and consequently expensive to use, which can be close to a situation without reliable edges as the situation of model Hop. On the opposite, if the useful reliable edges are close, the probability to use a reliable edge in the optimal design increases and, by the way, the complexity of the problem to solve.

Process	**Models**	
	HopR1	HopR2
Close	19.53	13.41
Far	16.33	11.72
Average gap	17.92	12.56

Table 6.6: Average gap (%) depending on the process used to generate reliable subset of edges.

Another property is linked to the scale economy factor f. On Table 6.7 we show the average gap (%) and on Table 6.8 the average IP CPU time (sec.) for TC and *SNDlib* test sets for $K = 2, 3$ for models HopR1 and HopR2 depending on the scale economy factor f. We show on Figure 6.20 that for model HopR2, as the factor f increases, the average gap decreases and also the average IP CPU time. This observation is not a surprise because when f increases it means that the economical incentive to use reliable edges decreases. Thus if it is less profitable to use reliable edges, it is easier to find the

optimal topology because the problem is close to the original problem without reliable edges. We could say that when f increases the model HopR2 approaches traditional extended model Hop without reliable edges, which is, as we know, an easier problem, that is the reason why the trend of gap and the IP CPU time decreases.

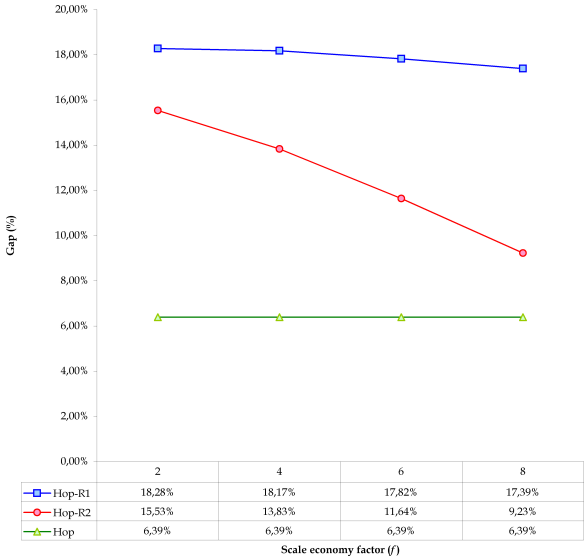


Figure 6.20: Evolution of the average gap (%) depending on the scale economy factor f.

			Scale economy factor f			
	Network	**K**	**0.2**	**0.4**	**0.6**	**0.8**
HopR1	TC	2	15.79	15.78	15.32	14.83
	TC	3	16.53	16.15	15.79	15.31
	SNDlib	2	23.72	23.97	23.82	23.46
	SNDlib	3	25.52	25.51	25.34	25.11
Average gap			18.28	18.17	17.82	17.39
HopR2	TC	2	15.03	13.50	10.93	8.05
	TC	3	13.60	11.73	9.79	7.64
	SNDlib	2	22.36	20.91	18.91	16.66
	SNDlib	3	14.16	12.06	10.06	8.15
Average gap			15.53	13.83	11.64	9.23

Table 6.7: Impact on the gap of the scale economy factor f.

			Scale economy factor f			
	Network	**K**	**0.2**	**0.4**	**0.6**	**0.8**
HopR1	TC	2	29.58	23.32	19.02	16.67
	TC	3	26.41	23.83	23.19	20.94
	SNDlib	2	8.55	7.12	5.82	4.38
	SNDlib	3	3.11	2.95	2.44	2.17
Average IP CPU Time			22.46	18.94	16.85	14.89
HopR2	TC	2	76.96	66.98	62.70	54.79
	TC	3	63.62	56.73	53.84	45.39
	SNDlib	2	84.59	62.13	35.62	27.79
	SNDlib	3	1.34	1.13	1.03	0.96
Average IP CPU Time			65.43	55.68	48.94	41.64

Table 6.8: Impact on the IP CPU Time of the scale economy factor f.

6.6.4 Economic advantages of reliable edges

The objective of this last section is to determine if there is a real interest for network providers to use the concept of reliable edges when they want to model the problem of topology in the virtual layer. As developed in the introduction of this chapter, the first advantage of this simplification is to ease the tasks of managers because a pool of edges in the physical layer can be reduced to a smaller set of edges in the virtual layer. It is clear that if the set of edges is smaller then the optimization work is less difficult. But concurrently to this aspect is it possible to find another economical incentive for the use of this kind of abstraction?

	HopR1 < Hop	HopR2 < Hop	HopR2 < HopR1
(%) of all the instances	23.89	47.67	83.25
Average Economy realized (%)	5.97	4.46	7.15
Maximum Economy realized (%)	27.23	27.23	34.02

Table 6.9: Economy proposed by the utilization of reliable edges.

Table 6.9 shows that in roughly 23% of the simulations (globally around 2500 simulations have been done for this chapter) model HopR1 gives a less expensive optimal design compared to model Hop, and the average saving is around 6% of the total design cost. In approximately 48% of all the instances, model HopR2 proposes a cheaper optimal design compared to model Hop with an average saving of 4.46% of the total design cost. Finally Model HopR2 is better (in terms of cost) than model HopR1 in 83.25% of the cases with an average saving around 7.17% of the total cost. It means that the improvements proposed by the utilization of reliable edges in terms of cost are not huge, indeed only small percentages of the total design cost can be saved. The question for managers now is to know if the increase in the time needed to reach optimality can be balanced by the saves in design cost. With a simple extended model the answer can be no, but with a fast branch-and-cut algorithm as the one developed in this chapter it could be worthwhile to integrate reliable edges.

As a summary, the main conclusion of this testing phase, which is not a surprise, is that "the problem becomes more difficult when the proportion of reliable edges increases as the gap increases". A second conclusion says that "for model HopR2, when the scale economy factor associated to the reliable edges increases, in other words, when it is not economically interesting enough to use reliable edges, the trend of the gap decreases because the HopR2 model approaches the original Hop model without reliable edges."

6.7 Summary

In this chapter we study the hop-constrained network design problem with reliable edges which are a class of edges not subject to failure. We propose two variants and their associated formulations valid for any L and K and we show that adding reliable edges make the problem more complex. Here again the huge number of variables and constraints linked to our extended formulations is a drawback. Then we propose to use the Benders decomposition to make easier the solving process. We use our best branch-and-cut

algorithm developed in Chapter 5 with our heuristic and we improve considerably the CPU time. From a numerical point of view we try to show how the original problem becomes more complex when reliable edges are added and we conclude with an economical analysis which evaluates the incentive to use reliable edges in network design.

Chapter 7

Conclusion

7.1 Contributions

In our everyday life, telecommunications are omnipresent. From giving a phone call to a friend to booking the next holidays, people, and more largely the entire society are totally dependent of the big panel of applications "offered" by the telecommunications industry. All this variety of proposed services is possible only thanks to small and large interconnected networks. In this thesis we represent networks by graphs like $G = (V, E)$ where a set of vertices V are potentially linked by a set of edges E. Generally a set of commodities Q must be routed on this graph G, that is, we have to determine for each commodity $q \in Q$ the optimal routes used to send the information from the origin node $o(q)$ to the destination node $d(q)$. Because our life depends so much on networks, those need to be as survivable as possible to ensure a certain level of reliability. Indeed networks need to work correctly all the time even in case of potential failure. The impact of a node or an edge failure could be detrimental for customers. In this work to ensure a certain level of reliability or survivability we propose to find for each commodity at least K edge-disjoint (or arc-disjoint) paths, that is, the K paths do not share any edge. As a consequence $(K-1)$ simultaneous edge or arc failures can affect the network without deteriorating the proposed service as the Kth path remains functional to send information. However survivability considerations alone does not guarantee an irreproachable quality of service, as it could be possible that one or many of the K selected paths can be too long. That can involve long waiting time for customers as long paths are used to route the information. This fear justifies that we impose a limit on the length of the routing paths. This limit is expressed in terms of the maximum number of edges (here called hop) belonging to feasible paths, i.e. a L-path is a path composed of at most L edges (or arcs). This thesis studies deeply **the survivable network design problem with quality of service constraints** which consists in, given a cost c_{ij} associated to each edge ij, finding among all the potential edges $ij \in E$, the subset on which install capacity allowing between each commodity $q \in Q$, at least K edge-disjoint (or arc-disjoint) paths composed of at most L hops.

The contributions of this thesis are varied. First we approach the survivability problem from an intuitive perspective. Indeed we consider that each edge $ij \in E$ can break

down independently, i.e. for each edge $ij \in E$ we associate one specific failure scenario in which the broken edge ij is removed from the set E. We build $|E|$ scenario or $|E|$ different problems. The objective of this research is thus to find the optimal capacity for each edge allowing connectivity in each failure scenario for all the $|Q|$ commodities. We propose an independent as well as a global formulation using two kinds of variables (the flow variables x as the capacity variables y). Because the proposed models include many variables (complicated or not), we develop a cutting plane algorithm based on the Benders decomposition to reduce the time needed to reach optimality. A second improvement proposed consists in grouping all the commodities with the same destination together, which reduces considerably the number of variables as the size (and the difficulty) of the problems. We show that for small instances a basic implementation of the model using the solver CPLEX 8.0 is better that the cutting plane algorithm. But things deeply change when the instances become larger. Improvement ratios around 10 between the basic implementation with CPLEX and the cutting plane algorithm mean that the Benders decomposition algorithm goes 10 times more quickly than CPLEX.

Then we forget about failure scenario and especially about the capacity definition problem. Indeed the objective of the second part of this thesis is more to determine which edges are include in the optimal topology rather than determining the capacity (of these edges) needed to ensure survivability. Here, we ensure survivability by looking for K edge (or arc)-disjoint paths for each commodity. Moreover we introduce the quality of service constraints by imposing a hop limit. We study the single commodity case of the problem which consists in finding K edge-disjoint L-paths between only one pair of terminals. This problem was largely studied before using only design variables (here the z variables). The main weakness of this approach is the difficulty to find valid formulations for any K and any L only using design variables. Dahl and later on Dahl and Gouveia proposed first the generalized jump-inequalities. Later Huygens *et al.* in ([30], [61], [62], [63]) proposed some nice cuts like the two-layered L-path-cut inequalities and the two-layered 4-path-cut inequalities. But additional and complex cuts are needed to propose a valid formulation for $L \geq 5$. In this thesis we give an extended arc-flow formulation valid for any K and L based on the new extended flow variables x. The additional index associated to this new variable x gives us additional important information about the position occupied by the edge (or the arc) in the routing paths. We propose and study two versions of the disjointness problem, the arc-disjoint as the edge-disjoint cases. In the literature, both problems are often confused since the arc-disjoint and the edge-disjoint cases are closely related. But when the costs can be negative, things become more complicated and a light difference is to be considered between both approaches. We prove that this extended formulation gives the complete description of the polyhedron for some values of the parameters K and L, e.g. for the arc-disjoint case, when $L = 3$ and $K \geq 1$ and for the edge-disjoint case when $L = 3$ and $K = 2$. For this last case, we add two new classes of valid inequalities and to prove that we have the complete description of the polyhedron, we transpose our problem to a stable set problem of size 2 for which we made a polyhedral study as a preliminary. For greater values of L, for instance $L = 4$, even if the formulation remains valid, we need new cuts to completely describe the polyhedron since some extreme points of the polyhedron are fractional. Then, we study the problem from a numerical point of view for large values of parameters K and L. We show that our formulation integrates well the survivability (K) as the quality of service (L) aspects

as out of 29.000 tests only 6 instances have a fractional optimal solution.

Naturally we extend this study to the multicommodity case in which the flow for several commodities has to be routed in the network. Indeed now it is a set of commodities Q for which we need to find at least K edge-disjoint L-paths leading to the minimum design cost. Here again, in the literature, most of the studies use the only category of the design variables z and here again we propose to use our extended formulation based on the extended flow variables x. Of course this extended formulation is valid for any K and L. Nevertheless one big disadvantage of this formulation is linked to the huge number of variables and constraints. Here, the additional index of flow variables x is shown firstly as a drawback since it increases considerably the number of variables compared with formulations with a small set of design variables z. However, the structure of the matrix of the extended formulation is suitable to justify the use of decomposition methods. As, in our formulation, we use two groups of variables (z and x) we choose the Benders decomposition to split the main problem into master problem and subproblems. From a computational point of view, the objective of this research is also to study the performance of the basic implementation with CPLEX of the extended formulation. As the Benders decomposition is a method based on a cutting-plane approach we implement two cutting-plane algorithms and evaluate also their performances. One weakness of the cutting-plane algorithms is that a new branch-and-bound tree is solved at each iteration since the master problem is an integer problem. Then we study in depth also when to generate the Benders cuts and we develop four branch-and-cut algorithms. Moreover one intuitive heuristic is developed to build a good upper bound. We present a thorough computational study of various cutting plane and branch-and-cut algorithms on a large set of instances including the real based instances from *SNDlib*. Our best branch-and-cut algorithm combined with an efficient heuristic is able to solve the instances significantly faster than CPLEX 11. Furthermore, our heuristic is also pretty fast, taking around 4 seconds whereas the extended formulation and our best branch-and-cut algorithm take respectively on average 582.05 and 74.03 seconds. In 139 cases out of 213 (around 65%), the solution given by the heuristic is the optimal one. Then, we show that our Benders cuts contain the constraints used by Huygens *et al.* ([30], [61], [62], [63]) to formulate the problem for $L = 2, 3, 4$, as well as new inequalities when $L \geq 5$.

Then we consider a new variation of the survivable network design problem with quality of service constraints in which we introduce a subset of edges considered as reliable edges. These reliable edges are not prone to failure, i.e. there exists already for these edges an underlying reliable structure. As a consequence the edge-disjointness has to be guaranteed only for the subset of non-reliable edges and no more for the whole set of edges. This could ease the work of network managers as a smaller set of edges (the set of the non-reliable edges) has to be considered for the survivability problem. The main idea of the reliable edges is that, if necessary, all the K paths can use the same reliable edge, i.e. the capacity of the reliable edges is assumed to be equal to K. From a cost point of view, an incentive to use reliable edges is linked with a scale economy factor f. Indeed, as all reliable edges can be used K times, there does not exist any real economical motivation if the installation cost of this kind of edges is equal to K times the normal installation cost. Therefore, we decide to introduce an idea similar to the scale economy principle by permitting a certain saving if reliable edges are used in the optimal

topology. As a consequence the reliable cost of an edge is assumed to be $((K-1)+f)$ times the normal cost of the same edge. In this thesis we model this new problem with again an extended formulation valid for any K and L. We derive two ways (HopR1 and HopR2) to formulate the presence of reliable edges. After studying the link between the two models we apply again the Benders decomposition approach for this problem and we developed a branch-and-cut algorithm and a heuristic to improve the CPU time. We present a thorough computational study on a large set of instances including the real based instances from *SNDlib*. From a numerical point of view, the main conclusion of this study is that when the proportion of reliable edges increases, the gap increases as well so that the problem becomes more difficult. A second conclusion says that for model HopR2, when the scale economy factor associated to the reliable edges increases, in other words, when it is not economically interesting enough to use reliable edges, the trend of the gap decreases because the HopR2 model approaches the original Hop model without reliable edges. The tests show also that in roughly 23% of the simulations the model HopR1 gives a less expensive optimal design compared to the model Hop, and the average economy is around 6% of the total design cost. In approximatively 48% of the case, model HopR2 is better than model Hop with an average economy of 4.46% of the total design cost. Finally Model HopR2 is better than model HopR1 in 83.25% of the cases with an improvement of 7.17% of the total cost.

7.2 Limits and future work

Concerning the limits of this research we can first point out the limited size of the networks used in our computational experiments. Even if real-based networks from *SNDlib* were used, the size of these networks, determined among others by $|V|, |E|$ and $|Q|$, stays far from reality. Future research could focus on computational experiments on larger sized networks for instance. We could anticipate the potential results saying that we could expect nice performance of our branch-and-cut algorithms if $|Q|$ increases since the real impact of the decomposition is strongly linked to the number of commodities. For instance, some tests were done for TC and TE networks where $|V| = 40$. For this kind of networks, depending on the value of parameters K and L of course, the basic implementation of the extended models with CPLEX took almost the entire 3600 seconds CPU time limit to close only 60% of the gap.

Concerning the K edge (arc)-disjoint L-paths problem, we limit our analysis to $L = 3$. A possible source of future research could be to look at cases where $L = 4$, $L = 5$ for instance. The idea consists in finding new valid inequalities missing to completely describe the associated polyhedron. Even if the gap (proposed by our extended formulation) remaining to close is microscopic (on average less than 1%), this problem of finding new valid inequalities remains challenging. First attempts were already approached during this thesis using the software PORTA. For instance, for a small network with 6 nodes when $L = 4$ and $K = 2$ the number of facets identified by PORTA [21] reach roughly 2000 inequalities.

We could also extend our work by looking at the survivable network design problem with quality of service constraints in which we introduce a subset of nodes considered as reliable nodes. We could compare from the point of view of network managers which

alternative (reliable nodes or reliable edges) is preferable. Or moreover we could consider a more general problem in which a subset of reliable edges as a subset of reliable nodes are considered together. Even if the problem seems to be more complex, the nature of the inequalities is already well known and an adaptation of our branch-and-cut could be direct. The main idea is always to stick our model to real situations. As in practise nodes as edges can both be broken simultaneously, it could be interesting to propose models which ensure survivability when both situations of failure can occur.

We could also modify the nature of the cost function and consider for instance a congestion level. We could also consider stochastic demand, stochastic scenario of failure,... Many more perspectives could be present. We just cite here some of them.

Bibliography

[1] T. Achterberg. SCIP: Solving constraint integer programs. *Math. Programming Comput.*, 1:1–41, 2009. [cited at p. 73]

[2] T. Achterberg and C. Raack. The MCF-separator – detecting and exploiting multi-commodity flows in MIPs. ZIB-Report 09-38, Konrad Zuse Zentrum fuer Informationstechnik Berlin, December 2009. [cited at p. 75]

[3] Y.K. Agarwal. Design of capacited multicommodity networks with multiple facilities. *Oper. Res.*, 50(2):333–344, 2002. [cited at p. 4]

[4] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows, theory, algorithms and applications*. Prentice Hall, 1993. [cited at p. 15]

[5] K. Altinkemer. Special issue of telecommunications systems: Wide area networks design and analysis foreword. *Telecommunications systems*, 20, 2002. [cited at p. 4, 5]

[6] Y.P. Aneja, R. Chandrasekaran, and K.P.K. Nair. Maximizing residual flow under an arc destruction. *Networks*, 38(4):194–198, 2001. [cited at p. 5]

[7] L. Bai, D. W. Hearn, and S. Lawphongpanich. Decomposition techniques for the minimum toll revenue problem. *Networks*, pages 142–150, 2004. [cited at p. 13]

[8] L. Bai and P. A. Rubin. Combinatorial Benders cuts for the minimum toll booth problem. *Oper. Res.*, 57(6):1510–1522, 2009. [cited at p. 73]

[9] G. Baier, T. Engel, A.M.C.A. Koster, S. Orlowski, C. Raack, and R. Wessäly. Single-layer cuts for multi-layer network design problems. ZIB Report ZR-07-21, August 2007. [cited at p. 93]

[10] A. Balakrishnan and K. Altinkemer. Using a hop-constrained model to generate alternative communication network design. *ORSA J. Comput.*, 4:192–205, 1992. [cited at p. 37, 67]

[11] A. Balakrishnan, T.L. Magnanti, and P. Mirchandani. *Annotated Bibliographies in Combinatorial Optimization*, chapter Network Design, pages 311–334. John Wiley and Sons, New York, 1997. [cited at p. 4]

[12] C. Barnhart, A. Farahat, and S. Shen. Network design in transportation and logistics: Models and algorithms. Technical report, Massachusetts Institute of Technology, 2002. [cited at p. 4, 13]

[13] W. Ben-Ameur and L. Gouveia. Some recent contributions to network optimization. *Networks*, pages 27–30, 2004. [cited at p. 4, 5]

[14] A. Benchakroun, J.A. Ferland, and V. Gascon. Benders decomposition for network design problems with underlying tree structure. *Investigacion Operativa*, 6(1,2 and 3):165–180, 1998. [cited at p. 6, 13]

[15] J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962. [cited at p. 71]

[16] D. Bienstock, S. Chopra, O. Günlük, and C-Y. Tsai. Minimum cost capacity installation for multicommodity network flows. Technical report, CORE, Université catholique de Louvain, 1995. [cited at p. 4, 5]

[17] J.R. Birge and F.V. Louveaux. *Introduction to Stochastic programming (2nd edition)*. Springer Verlag, New-York, 2008. [cited at p. 71]

[18] A. Bley. Node-disjoint length-restricted paths. Master's thesis, TU Berlin, 1997. [cited at p. 37, 40, 68]

[19] Q. Botton and B. Fortz. Decomposition methods for the optimization of survivable telecommunications networks. Technical Report 135, UCL/IAG, 2005. [cited at p. 20]

[20] C.C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Math. Programming*, 83(3, Ser. A):451–464, 1998. [cited at p. 71]

[21] T. Christof and A. Löbel. Polyhedron representation transformation algorithm. *http://www.zib.de/Optimization/Software/Porta/*. [cited at p. 124]

[22] M. Chudnovsky and P. Seymour. The structure of claw-free graphs. In London Mathematical Society, editor, *Proceedings of the 20th British Combinatorial Conference*, pages 153–171, 2005. 327. [cited at p. 30]

[23] G. Codato and M. Fischetti. Combinatorial Benders' cuts for mixed-integer linear programming. *Oper. Res.*, 54(4):756–766, 2006. [cited at p. 75]

[24] A.M. Costa. A survey on Benders decomposition applied to fixed-charge network design problems. *Comput. Oper. Res.*, 32(6):1429–1450, 2005. [cited at p. 69, 71]

[25] A.M. Costa, J.-F. Cordeau, and B. Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42:371–392, 2009. [cited at p. 71]

[26] G. Dahl. The 2-hop spanning tree problem. *Oper. Res. Lett.*, 23:21–26, 1998. [cited at p. 36]

[27] G. Dahl. Notes on polyhedra associated with hop-constrained walk polytopes. *Oper. Res. Lett.*, 25:97–100, 1999. [cited at p. 36, 68, 77, 79]

[28] G. Dahl, N. Foldnes, and L. Gouveia. A note on hop-constrained walk polytopes. *Oper. Res. Lett.*, 32(4):345–349, 2004. [cited at p. 36, 68]

[29] G. Dahl and L. Gouveia. On the directed hop-constrained shortest path problem. *Oper. Res. Lett.*, 32:15–22, 2004. [cited at p. 36, 77]

[30] G. Dahl, D. Huygens, A.R. Mahjoub, and P. Pesneau. On the k edge-disjoint 2-hop-constrained paths polytope. *Oper. Res. Lett.*, 34(5):577–582, 2006. [cited at p. 8, 38, 77, 79, 110, 122, 123]

[31] G. Dahl and B. Johannessen. The 2-path network problem. *Networks*, 43:190–199, 2004. [cited at p. 37, 68]

[32] G. Dahl, A. Martin, and M. Stoer. Routing through virtual paths in layered telecommunication networks. *Oper. Res.*, 47(5):693–702, 1999. [cited at p. 94]

[33] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS J. Comput.*, 10(1):1–11, 1998. [cited at p. 68]

[34] I. Diarrassouba. *Survivable network design problems with high survivability requirements*. PhD thesis, Université Blaise Pascal - Clermond-Ferrand II, 2009. [cited at p. 38, 77, 79, 110]

[35] E.D. Dolan and J.J. More. Benchmarking optimization software with performance profiles. *Math. Programming*, 91(2):201–213, 2002. [cited at p. 84]

[36] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards (B)*, 69:125–130, 1965. [cited at p. 29]

[37] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. [cited at p. 29]

[38] F. Eisenbrand, G. Oriolo, G. Stauffer, and P. Ventura. The stable set polytope of quasi-line graphs. *Combinatorica*, 28(1):45–67, 2008. [cited at p. 30]

[39] C. Feremans, M. Labbé, and G. Laporte. Generalized network design problems. *Eur. J. Oper. Res.*, (148), 2002. [cited at p. 4]

[40] M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of Benders' cuts. Mathematical Programming B - (to appear), 2008. [cited at p. 69, 71]

[41] B. Fortz and M. Labbé. Polyhedral results for two-connected networks with bounded rings. *Math. Programming*, 93(1):27–54, 2002. [cited at p. 67]

[42] B. Fortz and M. Labbé. Two-connected networks with rings of bounded cardinality. *Comput. Optim. and Appl.*, 27(2):123–148, 2004. [cited at p. 67]

[43] B. Fortz, A.R. Mahjoub, S.T. Mc Cormick, and P. Pesneau. Two-edge connected subgraphs with bounded rings: Polyhedral results and branch-and-cut. *Math. Programming*, 105:85–111, 2006. [cited at p. 67]

[44] B. Fortz and M. Poss. An improved Benders decomposition applied to a multi-layer network design problem. *Oper. Res. Lett.*, 37:359–364, 2009. [cited at p. 13, 69, 73, 74, 93]

[45] V. Gabrel, A. Knippel, and M. Minoux. A comparaison of heuristics for the discrete cost multicommodity network optimization problem. *J. of Heuristics*, 9(5):429–445, 2003. [cited at p. 4]

[46] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Oper. Res. Lett.*, 25:15–23(9), August 1999. [cited at p. 74]

[47] A.M. Geoffrion and G.W. Graves. Multicommodity distribution system design by benders decomposition. *Management Science*, 20(5):822–844, 1974. [cited at p. 3, 6, 13]

[48] R. Giles and L. Trotter. On stable set polyhedra for k1,3-free graphs. *Journal of Combinatorial Theory, Series B*, 31:313–326, 1981. [cited at p. 31]

[49] L. De Giovanni, B. Fortz, and M. Labbé. A branch-and-cut algorithm for the internet protocol network design problem. Technical report, Université libre de Bruxelles, 2004. [cited at p. 4, 5]

[50] A. Girard, B. Sanso, and L. Dadjo. A tabu search algorithm for access network design. *Annals of Oper. Res.*, (106):229–262, 2001. [cited at p. 6]

[51] L. Gouveia. Multicommodity flow models for spanning trees with hop constraints. *Eur. J. Oper. Res.*, 95:178–190, 1996. [cited at p. 36, 83, 112]

[52] L. Gouveia. Using variable redefinition for computing lower bounds for minimum spanning and steiner trees with hop constraints. *INFORMS J. Comput.*, 10:180–188, 1998. [cited at p. 36, 38, 68, 69, 94, 96]

[53] L. Gouveia and T.L. Magnanti. Network flow models for designing diameter-constrained minimum-spanning and steiner trees. *Networks*, 41(3):159–173, 2003. [cited at p. 68]

[54] L. Gouveia, A.R. Mahjoub, and P. Pesneau. Personal communication, 2009. [cited at p. 81]

[55] L. Gouveia, P.F. Patrício, A.F. de Sousa, and R. Valadas. MPLS over WDM Network Design with Packet Level QoS Constraints based on ILP Models. In *Proceedings of IEEE Infocom*, pages 576–586, 2003. [cited at p. 68]

[56] L. Gouveia, P.F. Patricio, and A.F. Sousa. *Telecommunications Planning: Innovations in Pricing, Network Design and Management*, chapter Compact models for hop-constrained node survivable network design, pages 167–180. Springer, New York, 2006. [cited at p. 38, 40, 68, 96]

[57] L. Gouveia, P.F. Patricio, and A.F. Sousa. Hop-contrained node survivable network design: An application to mpls over wdm. *Networks Spatial Econom.*, 8(1):3–21, 2008. [cited at p. 38, 40, 68, 96]

[58] L. Gouveia and C. Requejo. A new lagrangean relaxation approach for the hop-constrained minimum spanning tree problem. *Eur. J. Oper. Res.*, 132:539–552, 2001. [cited at p. 36]

[59] M. Grotschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1993. [cited at p. 28]

[60] K. Holmberg and D. Yuan. Optimization of internet protocol network, design and routing. *Networks*, 43(1):39–53, 2004. [cited at p. 6]

[61] D. Huygens, M. Labbé, A.R. Mahjoub, and P. Pesneau. The two-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut. *Networks*, 49(1):116–133, 2007. [cited at p. 8, 38, 68, 69, 122, 123]

[62] D. Huygens and A.R. Mahjoub. Integer programming formulations for the two 4-hop-constrained paths problem. *Networks*, 49(2):135–144, 2007. [cited at p. xiv, 8, 38, 68, 69, 77, 78, 79, 80, 81, 110, 122, 123]

[63] D. Huygens, A.R. Mahjoub, and P. Pesneau. Two edge-disjoint hop-constrained paths and polyhedra. *SIAM J. Discrete Math.*, 18(2):287–312, 2004. [cited at p. 8, 37, 38, 68, 69, 77, 79, 110, 122, 123]

[64] ILOG. *ILOG CPLEX 11.0 Reference Manual*. ILOG CPLEX Division, Gentilly, France, 2007. [cited at p. 73]

[65] A. Itaí, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 2:277–286, 1982. [cited at p. 68, 80]

[66] J. Janssen and K. Kilakos. Bounded stable sets: Polytopes and colorings. *SIAM J. Discrete Math.*, 12(2):262–275, 1999. [cited at p. 27, 32, 33]

[67] E. Kalvelagen. Benders decomposition withs Gams. Technical report, 2002. [cited at p. 13]

[68] H. Kerivin and A.R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005. [cited at p. 67]

[69] A. Knippel and B. Lardeux. The multi-layered network design problem. *Eur. J. Oper. Res.*, 127(1):87–99, November 2007. [cited at p. 93]

[70] A.M.C.A. Koster, S. Orlowski, C. Raack, and R. Wessäly. Two-layer network design by branch-and-cut featuring MIP-based heuristics. In *Proceedings of INOC, Spa, Belgium*, April 2007. [cited at p. 93]

[71] M. Labbé, R. Séguin, P. Soriano, and C. Wynants. Networks synthesis with non-simultaneous multicommodity flow requierements: bounds and heuristics. Technical report, Institut de statistique et de recherche opérationnelle, Université Libre de Bruxelles, 1999. [cited at p. 4, 5]

[72] T.M. Liebling, G. Oriolo, B. Spille, and G. Stauffer. On non-rank facets of the stable set polytope of claw-free graphs and circulant graphs. *Mathematical Methods of Operations Research*, 59:25–35, 2004. [cited at p. 31]

[73] A. Lisser, R. Sarkissian, and J. Vial. Optimal joint synthesis of base and reverse telecommunication networks. Technical report, Departement of management studies, University of Geneva, 1995. [cited at p. 5]

[74] A. Lisser, R. Sarkissian, and J.P. Vial. Survivability in telecommunication networks. Technical report, HEC-Genève, 1995. [cited at p. 5, 12]

[75] Y. Liu and K.S. Trivedi. A general framework for network survivability quantification. In *12th GI/ITG Conference on measuring, modelling and evaluation of computer and communication systems*, 2001. [cited at p. 5]

[76] I. Ljubic, P. Putz, and J.J. Salazar. Exact approaches to the single-source network loading problem. Technical Report 2009-05, University of Vienna, 2009. [cited at p. 69, 71]

[77] E. Loute. *Cours de Programmation mathématique*. CORE, 2000-2001. [cited at p. 12]

[78] L. Lovász. Stable sets and polynomials. *Discrete Mathematics*, 124:137–153, 1994. [cited at p. 28]

[79] C. L. Monma M. Grötschel and M. Stoer. Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM J. Optim.*, 2(3):274–504, 1992. [cited at p. 67]

[80] T.L. Magnanti, P. Mireault, and R.T. Wong. Tailoring Benders decomposition for uncapacited network design. *Math. programming study*, 26:112–154, 1986. [cited at p. 6, 13]

[81] T.L. Magnanti and R.T. Wong. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Oper. Res.*, 29(3):464–484, 1981. [cited at p. 74]

[82] J-F Maurras and S. Vanier. Network synthesis under survivability constraints. *4OR, Quaterly journal of the Belgian, French and Italian operations research societies*, 2(1):53–67, 2004. [cited at p. 4, 5, 6, 13]

[83] M. Minoux. Optimum synthesis of network with nonsimultaneous multicommodity flow requieremnts. *Annals of Discrete Mathematics*, 11:269–277, 1981. [cited at p. 4, 5]

[84] M. Minoux. Network synthesis and optimum network design problems: Models, solutions, methods and applications. *Network*, 19:313–360, 1989. [cited at p. 4]

[85] M. Minoux. Discrete cost multicommodity network optimization problems and exact solution methods. *Annals of operations research*, 106(1):19–46, 2001. [cited at p. 4]

[86] G.J. Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, pages 284–304, 1980. [cited at p. 29]

[87] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1988. [cited at p. 70]

[88] G. Oriolo. Clique family inequalities for the stable set polytope of quasi-line graphs. *Discrete Applied Mathematics*, 132:185–201, 2004. [cited at p. 30]

[89] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of INOC, Spa, Belgium*, April 2007. http://sndlib.zib.de. [cited at p. 69, 83, 112]

[90] H. Pirkul and S. Soni. New formulations and solution procedures for the hop constrained network design problem. *Eur. J. Oper. Res.*, 148:126–140, 2003. [cited at p. 37, 68]

[91] D. Rajan and A. Atamtürk. A directed cycled-based column-and-cut generation method for capacited survivable network design. *Networks*, 43(4):201–211, 2004. [cited at p. 4, 6]

[92] C.D. Randazzo and H.P.L. Luna. A comparison of optimal methods for local acess uncapacited network design. *Annals of operations reserach*, 106(1):263–286, 2001. [cited at p. 4]

[93] C.D. Randazzo, H.P.L. Luna, and P. Mahey. Benders decomposition for local access network design with two technologies. *Discrete mathematics and theoretical computer science*, 4:235–246, 2001. [cited at p. 4, 6, 13]

[94] A. Ben Rebea. *Etude des stables dans les graphes quasi adjoints*. PhD thesis, Université de Grenoble, 1981. [cited at p. 30]

[95] W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders decomposition by local branching. *INFORMS J. Comput.*, 21(2):333–345, 2009. [cited at p. 74]

[96] E. Rosenberg. Capacity requirements for node and arc survivable networks. *Telecommunication Systems*, 20(1-2):107–131, 2002. [cited at p. 5]

[97] J. Shaio. Constraint generation for network reliability problems. *Annals of Oper. Res.*, 106:155–180, 2001. [cited at p. 5]

[98] F.B. Shepherd. *Near-Perfection and Stable Set Polyhedra*. PhD thesis, University of Waterloo, 1990. [cited at p. 29]

[99] S. Soni and H. Pirkul. Design of survivable networks with connectivity requirements. *Telecommunication Systems*, 20(1-2):133–149, 2002. [cited at p. 5]

[100] M. Stoer. *Design of survivable networks*. Springer, 1993. [cited at p. 67]

[101] P. Tsamasphyrou, A. Renaud, and P. Carpentier. Transmission network planning under uncertainty with Benders decomposition. *Lecture Notes Econ. Math. Systems*, 481:457–472, 2000. [cited at p. 74]

[102] L.A. Wolsey. *Integer Programming*. John Wiley and Sons, New York, 1998. [cited at p. 32, 46, 47]

[103] K.A. Woolston and S.L. Albin. The design of centralized networks with reliability and availability constraints. *Comput. & OR*, 15(3):207–217, 1988. [cited at p. 36]

[104] T-H. Wu. Emerging technologies for fiber network survivability. *IEEE communications magazine*, 1995. [cited at p. 3, 4]

[105] C. Wynants. *Network synthesis problems*. PhD thesis, Université libre de Bruxelles-Institut de statistique et de recherche opérationnelle, 1999. [cited at p. 6, 9]

[106] D. Yuan. *An annotated bibliography in communication network design and routing*. PhD thesis, Institute of Technology, Linköpings Universitet, 2001. [cited at p. 93]

Appendices

Appendix A

Benders decomposition for the hop-constrainted survivable network design problem: Detailed Results

Instances			CPLEX		Cutting plane algorithms		Branch-and-cut algorithms				Times Ratio	
Name	L	K	layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	layered / bc-5-heur	layered-r / bc-5-heur
TC-5	3	1	0.77	0.41	1.172	0.88	0.48	0.25	0.39	0.19	4.07	2.16
	3	1	2.75	1.61	64.234	14.2	1.63	1.33	0.83	1.05	2.63	1.54
	3	1	0.28	0.31	0.812	0.61	0.19	0.17	0.17	0.09	2.99	3.32
	3	1	2.03	0.86	1.609	1.19	0.66	0.34	0.48	0.3	6.84	2.89
	3	1	1.83	0.91	4.359	2.14	1.42	0.55	0.67	0.45	4.04	2
	3	2	0.11	0.09	2.406	0.64	0.16	0.16	0.16	0.2	0.54	0.46
	3	2	0.38	0.25	4.328	0.83	0.2	0.17	0.16	0.23	1.6	1.07
	3	2	0.22	0.08	0.75	0.52	0.09	0.09	0.11	0.06	3.53	1.26
	3	2	0.59	0.42	1.625	0.59	0.22	0.11	0.23	0.14	4.21	2.99
	3	2	0.94	0.39	4.828	1.09	0.22	0.22	0.23	0.22	4.28	1.79
	3	3	2.34	0.89	15.516	3.14	0.72	0.39	0.53	0.44	5.35	2.03
	3	3	2.73	1.67	174.672	16.5	1.2	0.86	0.73	1.25	2.19	1.34
	3	3	1.3	0.2	0.5	0.41	0.13	0.11	0.14	0.08	16.63	2.6
	3	3	1.67	1.02	7.438	1.69	0.47	0.27	0.28	0.38	4.46	2.71
	3	3	2.25	0.86	24.516	0.8	0.81	0.22	0.38	0.36	6.27	2.39
	4	1	5.52	2.36	1.859	1.64	2.59	0.95	1.58	0.53	10.39	4.44
	4	1	41.55	54.92	420.875	41.97	21.5	5.39	5.61	7.23	5.74	7.59
	4	1	0.86	0.8	1.828	1	1.67	0.42	0.94	0.22	3.92	3.64
	4	1	7.14	3.09	2.828	1.98	3.02	2.19	1.48	0.95	7.49	3.25
	4	1	13.56	8.31	9.312	3.83	10.67	4.45	2.42	2.28	5.95	3.64
	4	2	1.67	1.2	10.188	1.8	0.67	0.78	0.69	1	1.67	1.2
	4	2	5.83	2.98	141.766	3.47	2.98	1.25	2.41	2.84	2.05	1.05
	4	2	0.8	1.64	1.594	0.92	0.44	0.45	0.42	0.31	2.55	5.26
	4	2	3.25	2.36	9.375	4.83	2.77	1.38	1.48	1.48	2.19	1.59
	4	2	8.11	3.77	42.844	9.69	4.14	2.7	2.09	1.33	6.11	2.84
	4	3	3.11	2.14	25.719	1.11	1.97	0.59	0.89	0.63	4.97	3.43
	4	3	54.66	23.42	3206.422	99.89	31.91	3.78	6.03	14	3.9	1.67
	4	3	0.52	1.25	1.438	0.73	0.59	1.22	0.61	0.28	1.84	4.45
	4	3	6.88	2.8	4.609	3.17	1.69	0.92	1.33	1.33	5.18	2.11
	4	3	8.36	5.44	342.062	19.38	3.59	0.72	3.95	2.08	4.02	2.62
	5	1	2.45	4.45	2.156	1.94	1.63	2.03	1.64	2.59	0.95	1.72
	5	1	61.67	209.77	129.453	60.03	138.34	5.67	19.7	14.56	4.24	14.41
	5	1	1.17	1.8	2.297	1.14	2.2	1.34	1.64	0.48	2.42	3.71
	5	1	11.66	9.83	3.469	1.75	15.61	4.19	3.47	2.39	4.87	4.11
	5	1	13.81	10.39	6.609	7.42	31.64	7.77	8.05	4.3	3.21	2.42
	5	2	0.22	0.16	6.656	1.64	0.78	0.75	0.83	0.69	0.32	0.23
	5	2	0.53	0.48	21.703	1.47	1.64	1.63	1.81	1.64	0.32	0.29
	5	2	0.31	0.25	2.328	1.17	0.59	0.61	0.63	0.59	0.53	0.42
	5	2	0.28	0.2	1.516	0.72	0.75	0.75	0.75	0.92	0.3	0.22
	5	2	0.2	0.16	2.094	0.95	1.02	1	1.02	0.34	0.59	0.45
	5	3	9.8	9.36	6.25	2.05	2.72	2.02	1.16	1.11	8.83	8.44
	5	3	389.7	445.73	2437.734	176.86	162.88	14.44	11.16	9.69	**40.23**	**46.01**
	5	3	0.59	1.25	1.703	0.95	0.39	0.34	0.41	0.5	1.19	2.5
	5	3	7.75	3.53	12.734	3.14	5.89	4.27	2.34	1.58	4.91	2.24
	5	3	17.3	27.72	71.938	9.05	29.17	6.05	8.48	4.16	4.16	6.67
Arithmetic Mean			15.54	18.92	160.89	11.35	10.98	1.90	2.23	1.94	4.77	3.85
Geometric Mean			2.40	1.75	9.12	2.56	1.70	0.89	1.01	0.80	3.00	2.19

Table A.1: CPU Time (sec.)/Gap on TC test set.

Instances			CPLEX		Cutting plane algorithms		Branch-and-cut algorithms				Times Ratio	
Name	L	K	layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	$\frac{\text{layered}}{\text{bc-5-heur}}$	$\frac{\text{layered-r}}{\text{bc-5-heur}}$
TC-10	3	1	18.91	17.08	85.703	25.13	16.78	2.19	4	3.61	5.24	4.73
	3	1	89.25	106.17	2927.312	242.48	56.56	17.16	24.59	13.98	6.38	7.59
	3	1	34.13	36.27	327.562	67.92	53.33	5.84	10.28	6.22	5.49	5.83
	3	1	17.45	19.61	48.828	6.94	7.19	1.88	2.44	2.77	6.31	7.09
	3	1	35.25	21.48	381.703	25.81	37.25	6.55	7.88	3.13	11.28	6.87
	3	2	20.91	15.34	1047.344	151.88	8.61	3.88	3.53	6.13	3.41	2.51
	3	2	111.27	70.44	-	807.34	35.3	16.83	7.42	40.55	2.74	1.74
	3	2	120.83	20.09	1186.219	85.23	12.38	5.11	3.11	7.39	16.35	2.72
	3	2	10.88	8.03	101.156	13.52	3.47	1.36	1.61	1.53	7.1	5.25
	3	2	30.38	15.05	418.75	90.11	6.5	2.92	5.42	2.44	12.46	6.17
	3	3	4.14	5.17	553.109	4.28	1.86	0.91	1.22	1.02	4.08	5.09
	3	3	8.7	5.19	1308	12.14	2.83	1	1.28	1.52	5.74	3.42
	3	3	14.75	6.72	737.391	4.53	2.28	1.8	1.28	3.42	4.31	1.96
	3	3	169.42	94.77	-	1724.25	23.97	9.41	9.25	25.91	6.54	3.66
	3	3	12.05	14.3	1729.297	4.56	3.56	2	1.3	2.36	5.11	6.06
	4	1	902.95	781.61	117.219	50.33	306.63	45.22	31.55	28.61	31.56	27.32
	4	1	1314.91	1591.61	1122.797	411.13	3427.22	147.55	174.47	69.28	18.98	22.97
	4	1	2447.97	1352.09	-	346.69	2454.55	60.34	132.53	51.89	47.18	26.06
	4	1	194.39	391.16	19.703	10.86	212.64	8.83	18.33	6.88	28.28	56.9
	4	1	1075.73	927.42	232.078	48.58	1067.77	27.83	74.08	20.56	52.32	45.1
	4	2	486.14	111.47	1803.734	59.17	50.95	12.7	8.39	32.8	14.82	3.4
	4	2	179.94	92.84	667.297	47.53	35.02	21.7	14.88	13.98	12.87	6.64
	4	2	269.5	181.72	1669.844	71.77	25.48	17.23	18.98	52.77	5.11	3.44
	4	2	128.02	83.84	343.891	26.91	17.14	4.72	6.91	15.25	8.39	5.5
	4	2	1428.61	510.33	2037.75	173.66	100.75	49	66.41	46.08	31	11.08
	4	3	212.94	132.19	998.406	7.27	39.77	4.97	7.47	18.25	11.67	7.24
	4	3	160.61	76.31	553.406	5.61	22.64	2.48	3.08	14.63	10.98	5.22
	4	3	786.06	123.33	-	403.81	135.64	18.28	15.52	75.13	10.46	1.64
	4	3	457.94	212.7	2249.516	13.3	193.92	15.02	16.36	24.55	18.66	8.67
	4	3	423.14	347.13	-	71.69	180.84	27.19	23.8	48.48	8.73	7.16
	5	1	1872.11	3145.39	59.719	59.56	*0.984%*	23.75	102.59	21.08	**88.82**	**149.23**
	5	1	3478.7	*9.594%*	1478.984	647.8	*21.573%*	178.84	268.33	72.22	48.17	49.74
	5	1	3415.08	*8.296%*	2459.641	693.48	*14.331%*	129.17	254.27	63.97	53.39	56.17
	5	1	784.41	1468.8	31.016	9.92	786.47	27.05	72.56	14.08	55.72	104.33
	5	1	2547.72	*4.018%*	180.172	190.19	*10.931%*	81.47	216.52	47.64	53.48	75.41
	5	2	208.73	71.94	194.906	9.39	13.45	11.36	9.41	14.58	14.32	4.93
	5	2	257.92	240.23	1581.578	53.75	82.92	24.56	16.52	27.31	9.44	8.8
	5	2	974.03	136.08	-	17.34	66.88	21.03	33.75	34.39	28.32	3.96
	5	2	1620.97	346.94	1673.109	68.03	173.16	38.59	29.31	25.03	64.76	13.86
	5	2	2441.41	1338.09	-	1575.86	517.52	79.47	24.89	172.23	14.17	7.77
	5	3	576.05	313.72	477.953	30.83	335.34	23.63	26.36	10.34	55.69	30.33
	5	3	800.89	316.56	1665	19.59	91.3	14.48	8.8	12.97	61.75	24.41
	5	3	81.59	43.61	166.547	5.77	13.03	5.67	7.52	5.48	14.88	7.95
	5	3	645.81	283.78	665.047	4.45	717.48	18.86	47.3	27.77	23.26	10.22
	5	3	690.28	460.23	-	79.3	237.86	29.44	20.44	36.42	18.95	12.64
Arithmetic Mean			701.40	>584.77	>1388.95	188.44	>575.32	27.76	40.80	27.26	22.64	19.31
Geometric Mean			225.34	>143.26	>703.72	48.25	>77.96	12.53	14.71	15.06	14.96	9.51

Table A.2: CPU Time (sec.)/Gap on TC test set.

Instances			CPLEX		Cutting plane algorithms		Branch-and-cut algorithms				Times Ratio	
Name	L	K	layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	layered / bc-5-heur	layered-r / bc-5-heur
TE-5	3	1	6.66	2.02	-	16.19	4.34	2.02	2.27	0.83	8.04	2.43
	3	1	2.47	1.19	226.73	5.91	2.06	0.92	1.27	0.42	5.85	2.82
	3	1	5.02	2.34	2808.03	102.23	3.83	2.02	2.30	2.06	2.43	1.14
	3	1	2.80	1.33	-	2.44	3.11	1.83	1.36	0.98	2.84	1.35
	3	1	3.88	2.95	-	38.27	3.27	1.95	1.59	1.36	2.85	2.17
	3	2	1.98	1.22	-	1.78	0.80	0.97	0.83	0.94	2.12	1.30
	3	2	1.45	0.72	433.63	1.19	0.44	0.94	0.36	0.48	3.00	1.49
	3	2	5.55	1.56	-	204.50	4.20	3.52	1.55	3.69	1.50	0.42
	3	2	3.23	1.19	-	6.48	2.70	0.56	1.25	1.22	2.65	0.97
	3	2	14.52	3.61	-	11.94	3.89	4.16	1.91	2.36	6.15	1.53
	3	3	0.09	0.08	-	0.88	0.30	0.30	0.30	0.58	0.16	0.13
	3	3	0.22	0.13	131.70	0.81	0.25	0.25	0.28	0.16	1.40	0.80
	3	3	0.23	0.25	1415.34	1.08	0.34	0.33	0.34	0.25	0.94	1.00
	3	3	0.09	0.08	429.38	0.77	0.25	0.23	0.25	0.30	0.32	0.26
	3	3	0.11	0.08	-	1.08	0.36	0.33	0.36	0.39	0.28	0.20
	4	1	31.69	17.81	-	1045.08	23.31	4.47	5.61	4.98	6.36	3.57
	4	1	7.41	4.28	123.59	5.31	9.88	2.52	4.70	0.89	8.31	4.80
	4	1	97.31	74.61	-	558.42	40.02	8.25	7.20	11.39	8.54	6.55
	4	1	32.88	18.58	-	39.72	17.69	4.58	7.86	4.19	7.85	4.44
	4	1	31.53	25.95	-	70.38	53.61	13.73	16.72	5.16	6.12	5.03
	4	2	11.31	5.92	-	413.34	9.52	3.94	3.83	5.06	2.23	1.17
	4	2	4.16	2.38	-	6.80	1.98	1.59	1.30	1.53	2.71	1.55
	4	2	3.89	2.34	-	5.23	2.08	2.00	1.52	1.61	2.42	1.46
	4	2	3.39	0.70	-	2.91	2.92	2.59	2.06	1.44	2.36	0.49
	4	2	15.95	8.95	-	117.94	17.75	8.34	5.67	15.09	1.06	0.59
	4	3	5.83	1.66	-	7.63	3.83	1.25	1.61	2.61	2.23	0.63
	4	3	3.77	1.64	-	4.42	2.16	0.97	1.34	1.30	2.90	1.27
	4	3	41.52	16.97	-	128.59	20.31	5.22	3.69	9.72	4.27	1.75
	4	3	3.13	3.52	-	9.16	2.30	1.17	1.41	1.28	2.44	2.74
	4	3	26.81	12.41	-	129.00	12.45	6.78	7.25	15.06	1.78	0.82
	5	1	171.42	69.66	-	-	43.02	12.16	9.94	8.77	**19.56**	7.95
	5	1	9.19	30.91	146.11	6.98	24.38	5.36	11.47	4.14	2.22	7.46
	5	1	270.33	196.78	-	617.86	131.30	16.64	16.70	18.66	14.49	10.55
	5	1	46.69	39.31	-	22.81	54.19	6.78	7.58	6.55	7.13	6.00
	5	1	65.48	63.41	-	135.03	123.03	18.00	18.02	7.16	9.15	8.86
	5	2	0.42	0.47	-	2.13	0.66	0.64	0.66	0.63	0.68	0.75
	5	2	0.20	0.16	66.27	1.89	0.33	0.33	0.33	0.39	0.52	0.40
	5	2	0.34	0.30	3123.61	2.66	1.50	1.47	1.48	1.22	0.28	0.24
	5	2	0.23	0.17	-	2.78	0.44	0.44	0.44	0.44	0.53	0.39
	5	2	0.33	0.27	1280.48	2.06	0.97	0.95	0.95	1.17	0.28	0.23
	5	3	22.89	12.25	-	66.84	6.61	4.73	5.69	5.89	3.89	2.08
	5	3	20.45	18.16	-	21.72	6.11	5.09	2.75	3.67	5.57	4.94
	5	3	59.23	193.14	-	437.06	133.28	9.41	8.92	9.23	6.41	**20.92**
	5	3	25.42	18.98	1881.30	27.23	26.34	4.95	3.63	2.84	8.94	6.68
	5	3	88.64	45.94	-	174.75	41.67	25.66	16.69	11.08	8.00	4.15
Arithmetic Mean			25.56	20.14	>2928.77	179.39	18.75	4.45	4.29	3.98	4.22	3.03
Geometric Mean			5.15	3.23	>2128.94	17.19	4.62	2.25	2.19	2.03	2.54	1.59

Table A.3: CPU Time (sec.)/Gap on TE test set.

Instances			CPLEX		Cutting plane algorithms		Branch-and-cut algorithms				Times Ratio	
Name	L	K	layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	$\frac{\text{layered}}{\text{bc-5-heur}}$	$\frac{\text{layered-r}}{\text{bc-5-heur}}$
TE-10	3	1	204.33	356.42	-	1213.70	225.75	83.36	74.48	70.11	2.91	5.08
	3	1	145.08	145.45	-	474.13	152.34	34.72	31.58	10.75	13.50	13.53
	3	1	133.06	49.30	-	251.05	148.63	36.77	30.67	12.83	10.37	3.84
	3	1	298.45	262.11	-	1008.14	402.69	57.31	44.05	18.09	16.49	14.49
	3	1	152.05	126.17	-	617.61	104.95	39.88	44.31	20.70	7.34	6.09
	3	2	1068.72	344.83	-	-	561.25	1390.19	250.30	226.00	4.73	1.53
	3	2	406.78	255.03	-	-	418.13	506.95	135.25	70.20	5.79	3.63
	3	2	209.34	64.34	-	1355.34	130.00	99.50	37.59	10.41	20.12	6.18
	3	2	929.31	380.14	-	-	901.48	509.38	228.33	197.59	4.70	1.92
	3	2	222.20	54.28	-	2157.88	32.31	24.81	34.78	31.89	6.97	1.70
	3	3	1337.38	316.56	-	-	759.38	1358.48	115.41	106.78	12.52	2.96
	3	3	428.67	223.48	-	-	170.27	196.25	33.11	36.05	11.89	6.20
	3	3	284.09	140.70	-	-	130.22	186.06	30.17	14.08	20.18	9.99
	3	3	532.25	228.41	-	-	246.75	303.95	71.58	100.83	5.28	2.27
	3	3	243.53	159.17	-	-	158.23	51.08	18.36	68.58	3.55	2.32
	4	1	*3.004%*	*7.1%*	-	-	*20.755%*	934.47	1218.25	1003.97	3.58	3.58
	4	1	*3.189%*	3208.55	-	-	*15.05%*	801.28	780.91	217.80	16.50	14.73
	4	1	1376.64	1223.31	-	525.80	*22.731%*	299.89	269.64	60.78	22.65	20.13
	4	1	*6.743%*	3320.08	-	-	*8.836%*	460.58	372.25	213.08	16.86	15.58
	4	1	858.53	1774.08	-	445.08	1897.95	82.97	102.31	46.64	18.41	38.04
	4	2	*10.531%*	2944.84	-	-	*18.625%*	*10.247%*	*3.736%*	680.78	5.28	4.33
	4	2	*5.227%*	*2.393%*	-	-	*8.649%*	*4.467%*	1247.91	964.98	3.73	3.71
	4	2	2067.00	1418.67	-	-	*0.942%*	1596.58	427.66	223.86	9.23	6.34
	4	2	*3.088%*	*1.896%*	-	-	*6.934%*	2956.81	406.28	3428.88	1.05	1.05
	4	2	1450.31	958.64	-	-	506.31	110.86	73.31	159.81	9.08	6.00
	4	3	2721.97	2201.00	-	-	3458.55	537.09	550.64	194.03	14.03	11.34
	4	3	1765.05	1116.25	-	-	889.13	242.31	53.66	112.55	15.68	9.92
	4	3	3244.56	1850.36	-	-	2085.77	342.13	104.47	86.59	37.47	21.37
	4	3	1028.31	842.83	-	-	282.20	70.56	45.77	545.94	1.88	1.54
	4	3	1866.00	1453.67	-	1981.47	414.42	110.36	62.38	377.36	4.94	3.85
	5	1	*18.232%*	*16.572%*	-	-	*24.534%*	796.23	1542.19	1704.95	2.10	2.11
	5	1	*17.29%*	*10.863%*	-	1530.45	*23.769%*	422.86	776.53	147.33	24.39	24.43
	5	1	*12.145%*	*12.099%*	-	3197.44	*27.486%*	212.08	582.16	209.61	17.07	17.17
	5	1	*14.862%*	*9.835%*	-	1879.23	*33.676%*	338.41	370.27	65.59	54.24	54.86
	5	1	1564.27	3249.44	874.38	167.80	*3.044%*	83.44	211.86	42.63	36.70	76.23
	5	2	*19.046%*	*2.951%*	-	-	*12.152%*	*2.285%*	1572.27	1097.52	3.27	3.28
	5	2	*8.392%*	*6.052%*	-	-	*8.477%*	*4.583%*	734.98	784.69	4.58	4.59
	5	2	2112.33	1905.20	-	-	1189.95	333.75	98.69	55.89	37.79	34.09
	5	2	3040.08	3313.55	-	-	547.17	349.61	80.34	235.56	12.91	14.07
	5	2	2436.67	928.13	-	-	428.06	264.11	54.67	193.64	12.58	4.79
	5	3	*3.836%*	3179.77	-	-	*2.019%*	298.69	235.52	148.27	24.21	21.45
	5	3	727.23	74.44	815.03	7.44	30.89	19.34	8.53	18.36	39.61	4.05
	5	3	2506.97	1465.09	-	445.52	548.31	55.36	73.02	13.95	**179.67**	**105.00**
	5	3	893.83	311.48	-	32.42	162.17	17.78	19.13	19.31	46.28	16.13
	5	3	745.66	1544.55	-	213.17	275.41	29.78	26.02	25.22	29.57	61.25
Arithmetic Mean			>1858.95	>1638.99	>3538.85	>2588.67	>1660.34	>679.88	>372.85	312.77	18.93	15.26
Geometric Mean			>1200.05	>847.64	>3432.45	>1656.82	>780.88	>246.15	>138.14	108.22	11.09	7.83

Table A.4: CPU Time (sec.)/Gap on TE test set.

Instances			CPLEX		Cutting plane algorithms		Branch-and-cut algorithms				Times Ratio	
Name	L	K	layered	layered-r	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	layered / bc-5-heur	layered-r / bc-5-heur
pdh	3	1	73.5	48.63	-	671.13	16	5.03	3.22	15.33	4.8	3.17
	3	2	146.3	62.45	-	1879.48	43.72	13.75	5.63	13.23	11.05	4.72
	3	3	14.63	6.28	579.53	8.3	2.92	0.53	1.03	1.09	13.37	5.74
	4	1	418.92	328.41	-	685.91	231.78	24.94	26.41	24.17	17.33	13.59
	4	2	703	373.92	-	-	299.86	26.63	32.61	37.61	18.69	9.94
	4	3	22.95	8.64	17.67	1.53	2.02	0.67	0.69	0.25	**91.81**	34.56
	5	1	1463.02	1874.63	693.44	469.19	2429.36	71.34	79.16	36.47	40.12	51.4
	5	2	1419.48	846.03	-	-	166.7	38.08	24.06	32.8	43.28	25.8
	5	3	8.2	9.36	0.84	0.86	1.58	0.42	0.53	0.16	52.58	**59.99**
Arithmetic Mean			474.44	395.37	>2160.88	>1216.18	354.88	20.15	19.26	17.90	32.56	23.21
Geometric Mean			137.96	87.49	>538.13	>156.97	38.43	6.43	6.07	5.95	23.17	14.69
di-yuan	3	1	210.7	50.7	-	148.11	29.44	4.22	2.64	8.41	25.07	6.03
	3	2	210.53	30.55	1201.48	44.7	13.19	3.73	2.45	6.25	33.68	4.89
	3	3	92.67	12.84	236.19	2.56	3.03	0.95	0.97	0.98	94.18	13.05
	4	1	181.27	165.59	154.55	15.95	14.84	2.19	6.09	2.25	80.56	**73.6**
	4	2	421.59	142.58	460.72	80.61	26.02	3.22	3.33	4.3	98.11	33.18
	4	3	9.94	8.16	3.7	1.56	1.5	0.3	0.81	0.25	39.75	32.62
	5	1	573.31	129.23	28.3	9.03	21.59	3.73	6.3	3.05	188.16	42.41
	5	2	738.73	93.23	56.89	21.02	12.28	5	3.78	3.11	237.61	29.99
	5	3	25.23	2.27	0.75	0.66	0.38	0.22	0.3	0.08	**323.51**	29.05
Arithmetic Mean			273.78	70.57	>640.71	36.02	13.59	2.62	2.96	3.19	124.52	29.43
Geometric Mean			147.66	36.12	>88.33	11.74	7.42	1.72	2.08	1.65	89.31	21.84
dfn-gwin	3	1	3.19	1.09	19.61	1.41	2.02	0.48	0.67	0.17	18.53	6.36
	3	2	9.95	2.38	329.94	11.38	5.8	0.84	1.14	0.88	11.37	2.71
	4	1	9.31	4.41	155.19	5.39	5.97	1.69	1.13	0.59	15.68	7.42
	4	2	6.77	2.66	114.94	2.98	5.45	1.05	1.47	0.52	13.11	5.15
	5	1	37.72	19.66	5.72	4.77	16.67	1.69	2.03	0.89	**42.33**	**22.06**
	5	2	26.34	4.48	38.05	10.92	4.69	1.5	1.47	0.98	26.77	4.56
Arithmetic Mean			15.55	5.78	110.57	6.14	6.77	1.21	1.32	0.67	21.30	8.04
Geometric Mean			11.21	3.73	54.11	4.87	5.57	1.11	1.25	0.59	19.14	6.36
polska	3	1	0.14	0.13	5.44	0.66	0.14	0.11	0.14	0.06	2.27	2.02
	4	1	0.97	0.59	25.41	1.17	0.34	0.25	0.28	0.33	2.95	1.81
	4	2	0.25	0.16	1.47	0.5	0.11	0.13	0.11	0.03	8.06	**5.03**
	5	1	7	2.47	19.05	2.34	0.95	0.67	0.56	0.84	8.29	2.93
	5	2	1.66	0.61	2.58	1.02	0.42	0.33	0.39	0.13	**13.25**	4.87
Arithmetic Mean			2.00	0.79	10.79	1.14	0.39	0.30	0.30	0.28	6.97	3.33
Geometric Mean			0.83	0.44	6.31	0.98	0.29	0.24	0.25	0.15	5.69	3.05
nobel-us	3	1	0.45	0.34	29.44	1.38	0.27	0.22	0.25	0.06	7.31	5.55
	4	1	3.5	2.34	179.77	3.28	0.89	0.59	0.77	0.58	6.06	4.06
	5	1	64	26.77	270.94	24.77	3.89	3.45	2.56	5.17	**12.37**	5.18
	5	2	5.83	3.55	7.84	2.48	1.14	1.03	1.25	0.48	12.04	**7.33**
Arithmetic Mean			18.45	8.25	122.00	7.98	1.55	1.32	1.21	1.57	9.44	5.53
Geometric Mean			4.93	2.96	57.91	4.08	1.01	0.82	0.88	0.55	9.01	5.40

Table A.5: CPU Time (sec.)/Gap on SNDlib test set.

Instances Name	L	K	IP*	LP*	Gap(%)	heuristic*	Gap(%)	heuristic CPU Time
TC-5	3	1	116	105	9.48	119	2.59	0.28
	3	1	166	127.92	22.94	166	0	0.30
	3	1	102	93.83	8.01	102	0	0.20
	3	1	160	130.33	18.54	163	1.88	0.23
	3	1	157	126.17	19.64	157	0	0.38
	3	2	210	210	0	210	0	0.17
	3	2	260	255.83	1.6	260	0	0.22
	3	2	188	187.67	0.18	188	0	0.16
	3	2	269	260.67	3.1	269	0	0.14
	3	2	262	252.33	3.69	262	0	0.27
	3	3	336	322.2	4.11	336	0	0.31
	3	3	416	392	5.77	419	0.72	0.30
	3	3	299	296.25	0.92	299	0	0.17
	3	3	425	404.42	4.84	425	0	0.23
	3	3	404	387.61	4.06	406	0.5	0.33
	4	1	116	103.67	10.63	119	2.59	0.75
	4	1	166	124.25	25.15	166	0	0.84
	4	1	102	92.25	9.56	102	0	0.50
	4	1	160	125.5	21.56	163	1.88	0.48
	4	1	157	121	22.93	162	3.18	0.58
	4	2	210	207.33	1.27	210	0	0.63
	4	2	260	248.5	4.42	260	0	1.06
	4	2	188	184.5	1.86	188	0	0.41
	4	2	269	251	6.69	279	3.72	0.53
	4	2	262	242	7.63	273	4.2	0.61
	4	3	327	320	2.14	335	2.45	0.56
	4	3	410	379.8	7.37	419	2.2	1.50
	4	3	299	292	2.34	299	0	0.41
	4	3	413	395	4.36	417	0.97	0.98
	4	3	400	375.5	6.13	406	1.5	0.72
	5	1	116	103	11.21	136	17.24	1.13
	5	1	166	122	26.51	180	8.43	2.41
	5	1	102	91	10.78	102	0	0.94
	5	1	160	124.5	22.19	177	10.63	0.73
	5	1	157	116	26.11	162	3.18	0.88
	5	2	206	206	0	206	0	0.83
	5	2	244	244	0	244	0	1.75
	5	2	182	182	0	182	0	0.72
	5	2	249	249	0	249	0	0.81
	5	2	232	232	0	232	0	1.03
	5	3	327	320	2.14	335	2.45	0.91
	5	3	410	375.25	8.48	416	1.46	3.23
	5	3	299	292	2.34	299	0	0.47
	5	3	413	393	4.84	417	0.97	1.28
	5	3	400	374.5	6.38	427	6.75	2.63
Arithmetic Mean			248.27	231.97	8.04	252.07	1.76	0.76
Geometric Mean			226.98	207.85	0.00	230.87	0.00	0.56

Table A.6: LP Relaxation. Integer optimal solution and heuristic performances on TC test set.

Instances								
Name	**L**	**K**	**IP***	**LP***	**Gap(%)**	**heuristic***	**Gap(%)**	**heuristic CPU Time**
TC-10	3	1	224	175.98	21.44	224	0	0.80
	3	1	241	174.75	27.49	241	0	1.05
	3	1	228	178.1	21.89	228	0	1.31
	3	1	235	180.83	23.05	235	0	1.64
	3	1	237	178.36	24.74	237	0	1.19
	3	2	392	351.95	10.22	392	0	0.97
	3	2	405	349.49	13.71	410	1.23	1.30
	3	2	402	356.26	11.38	402	0	1.39
	3	2	400	361.66	9.59	400	0	0.86
	3	2	408	356.73	12.57	415	1.72	1.36
	3	3	557	539.3	3.18	557	0	0.98
	3	3	559	537.45	3.85	559	0	1.00
	3	3	563	539.28	4.21	563	0	0.98
	3	3	616	565.33	8.23	621	0.81	2.03
	3	3	591	559.4	5.35	591	0	1.17
	4	1	218	164.77	24.42	219	0.46	5.08
	4	1	229	163.98	28.39	229	0	4.05
	4	1	224	167.32	25.3	224	0	4.14
	4	1	222	172.62	22.24	225	1.35	4.33
	4	1	224	166.8	25.54	228	1.79	4.55
	4	2	359	329.54	8.21	359	0	3.39
	4	2	354	327.97	7.35	354	0	3.50
	4	2	360	334.64	7.04	360	0	3.75
	4	2	369	345.23	6.44	369	0	3.69
	4	2	371	333.59	10.08	371	0	5.03
	4	3	529	504.66	4.6	529	0	2.80
	4	3	525	506.36	3.55	531	1.14	2.81
	4	3	540	510.24	5.51	542	0.37	3.64
	4	3	565	536.98	4.96	567	0.35	3.67
	4	3	562	521.93	7.13	562	0	4.05
	5	1	216	160.31	25.78	216	0	8.61
	5	1	228	159.28	30.14	228	0	10.39
	5	1	223	161.12	27.75	223	0	20.17
	5	1	222	169.18	23.79	225	1.35	9.67
	5	1	224	162.68	27.38	229	2.23	10.22
	5	2	333	320.63	3.72	342	2.7	6.17
	5	2	342	318.55	6.86	348	1.75	8.25
	5	2	341	322.24	5.5	341	0	13.86
	5	2	361	338.36	6.27	361	0	9.33
	5	2	361	325.36	9.87	361	0	8.50
	5	3	521	496.47	4.71	521	0	5.38
	5	3	523	496.88	5	523	0	5.56
	5	3	505	491.08	2.76	505	0	6.47
	5	3	557	526	5.57	566	1.62	5.63
	5	3	547	510.77	6.62	547	0	10.28
Arithmetic Mean			382.51	343.34	12.96	384.00	0.42	4.78
Geometric Mean			358.32	310.07	9.96	359.82	0.00	3.38

Table A.7: LP Relaxation. Integer optimal solution and heuristic performances on TC test set.

Instances								
Name	**L**	**K**	**IP***	**LP***	**Gap(%)**	**heuristic***	**Gap(%)**	**heuristic CPU Time**
TE-5	3	1	200	158.83	20.58	200	0	0.77
	3	1	157	125.57	20.02	157	0	0.39
	3	1	193	148.00	23.32	198	2.59	0.39
	3	1	183	153.80	15.96	183	0	0.42
	3	1	224	174.25	22.21	224	0	0.67
	3	2	335	317.67	5.17	335	0	0.66
	3	2	267	251.13	5.94	275	3	0.34
	3	2	337	296.00	12.17	355	5.34	0.45
	3	2	330	307.60	6.79	330	0	0.44
	3	2	384	348.50	9.24	416	8.33	0.56
	3	3	479	479.00	0.00	479	0	0.34
	3	3	383	383.00	0.00	383	0	0.28
	3	3	449	448.57	0.10	449	0	0.45
	3	3	463	463.00	0.00	463	0	0.28
	3	3	526	526.00	0.00	526	0	0.38
	4	1	199	147.75	25.75	200	0.5	0.86
	4	1	149	118.88	20.22	149	0	0.88
	4	1	190	136.00	28.42	191	0.53	1.03
	4	1	180	144.50	19.72	205	13.89	0.77
	4	1	211	160.75	23.82	219	3.79	1.44
	4	2	328	295.50	9.91	328	0	0.63
	4	2	257	237.75	7.49	257	0	0.69
	4	2	287	272.00	5.23	287	0	0.77
	4	2	305	289.00	5.25	305	0	0.45
	4	2	353	321.50	8.92	361	2.27	0.92
	4	3	476	453.75	4.67	479	0.63	1.09
	4	3	383	365.00	4.70	386	0.78	0.48
	4	3	449	416.13	7.32	449	0	1.45
	4	3	463	447.00	3.46	463	0	0.78
	4	3	525	492.50	6.19	533	1.52	1.09
	5	1	199	144.00	27.64	218	9.55	1.44
	5	1	149	113.00	24.16	168	12.75	1.17
	5	1	189	133.50	29.37	195	3.17	1.38
	5	1	180	142.50	20.83	207	15	0.66
	5	1	211	154.00	27.01	216	2.37	1.09
	5	2	288	288.00	0.00	288	0	0.72
	5	2	226	226.00	0.00	226	0	0.38
	5	2	267	267.00	0.00	267	0	1.56
	5	2	285	285.00	0.00	285	0	0.50
	5	2	308	308.00	0.00	308	0	1.02
	5	3	475	447.25	5.84	478	0.63	1.78
	5	3	383	364.10	4.93	386	0.78	1.53
	5	3	449	413.00	8.02	449	0	4.48
	5	3	463	445.00	3.89	503	8.64	1.41
	5	3	518	487.30	5.93	527	1.74	4.33
Arithmetic Mean			316.78	291.04	10.67	322.36	2.17	0.97
Geometric Mean			294.79	261.73	0.00	300.99	0.00	0.77

Table A.8: LP Relaxation. Integer optimal solution and heuristic performances on TE test set.

Instances								
Name	**L**	**K**	**IP***	**LP***	**Gap(%)**	**heuristic***	**Gap(%)**	**heuristic CPU Time**
TE-10	3	1	306	211.54	30.87	307	0.33	3.38
	3	1	281	214.40	23.70	281	0	2.80
	3	1	296	217.29	26.59	296	0	2.05
	3	1	266	194.27	26.97	266	0	4.17
	3	1	257	182.25	29.09	257	0	1.86
	3	2	547	423.08	22.65	554	1.28	12.91
	3	2	520	428.92	17.52	520	0	4.73
	3	2	515	434.57	15.62	515	0	2.56
	3	2	473	388.54	17.86	473	0	5.52
	3	2	432	364.50	15.63	432	0	2.88
	3	3	761	653.54	14.12	761	0	14.27
	3	3	721	651.11	9.69	721	0	3.05
	3	3	743	666.05	10.36	743	0	3.56
	3	3	653	584.52	10.49	653	0	3.42
	3	3	627	559.60	10.75	629	0.32	2.08
	4	1	282	189.40	32.84	282	0	9.19
	4	1	271	192.68	28.90	271	0	11.56
	4	1	272	195.88	27.99	272	0	6.72
	4	1	251	175.90	29.92	251	0	5.27
	4	1	230	160.41	30.26	230	0	6.83
	4	2	458	378.81	17.29	458	0	11.88
	4	2	457	385.36	15.68	457	0	11.63
	4	2	462	391.76	15.20	465	0.65	6.44
	4	2	417	351.81	15.63	417	0	6.02
	4	2	371	320.82	13.53	371	0	5.55
	4	3	654	599.42	8.35	654	0	19.78
	4	3	636	588.34	7.49	643	1.1	7.73
	4	3	663	601.62	9.26	680	2.56	13.41
	4	3	576	531.50	7.73	579	0.52	3.39
	4	3	549	507.24	7.61	557	1.46	6.83
	5	1	269	180.47	32.91	273	1.49	20.27
	5	1	258	179.89	30.28	258	0	15.84
	5	1	272	186.79	31.33	272	0	15.56
	5	1	238	167.33	29.69	243	2.1	14.28
	5	1	216	152.97	29.18	216	0	11.30
	5	2	424	360.94	14.87	424	0	20.56
	5	2	423	359.78	14.95	427	0.95	21.59
	5	2	423	373.59	11.68	423	0	18.14
	5	2	377	334.67	11.23	377	0	12.14
	5	2	346	305.94	11.58	348	0.58	10.30
	5	3	622	574.65	7.61	626	0.64	19.81
	5	3	580	565.57	2.49	580	0	7.92
	5	3	625	577.98	7.52	637	1.92	15.88
	5	3	540	510.83	5.40	540	0	6.58
	5	3	519	481.27	7.27	520	0.19	10.95
Arithmetic Mean			446.20	379.06	17.72	447.98	0.36	9.39
Geometric Mean			417.23	341.13	15.20	418.71	0.00	7.48

Table A.9: LP Relaxation. Integer optimal solution and heuristic performances on TE test set.

Instances Name	L	K	IP*	LP*	Gap(%)	heuristic*	Gap(%)	heuristic CPU Time
pdh	3	1	1102316	697172.33	36.75	1102264	0.00	2.73
	3	2	1806322	1394344.66	22.81	1806322	0.00	2.36
	3	3	2313089	2106296.01	8.94	2313089	0.00	0.91
	4	1	928502	566412.26	39.00	928502	0.00	9.52
	4	2	1424803	1132824.52	20.49	1424803	0.00	8.45
	4	3	1802877	1732852.33	3.88	1802877	0.00	0.84
	5	1	906692	542005.34	40.22	906692	0.00	23.89
	5	2	1290545	1084010.69	16.00	1290545	0.00	9.58
	5	3	1747757	1693321.75	3.11	1747757	0.00	0.30
Arithmetic Mean			1480322.55	1216582.21	21.25	1480285.34	0.00	6.51
Geometric Mean			1413083.84	1095094.15	15.33	1413033.08	0.00	3.11
di-yuan	3	1	37240000	22733835.26	38.95	37240000	0.00	3.02
	3	2	58530000	45467670.51	22.32	58530000	0.00	3.06
	3	3	77020000	68201505.77	11.45	77020000	0.00	1.25
	4	1	26540000	17267214.92	34.94	26540000	0.00	2.42
	4	2	44320000	34534429.84	22.08	44320000	0.00	2.64
	4	3	57190000	53447021.28	6.54	57190000	0.00	0.80
	5	1	23510000	15736039.82	33.07	23510000	0.00	3.61
	5	2	37790000	31472079.65	16.72	37790000	0.00	2.83
	5	3	53270000	51885000	2.60	53270000	0.00	0.27
Arithmetic Mean			46156666.67	37860533.01	20.96	46156666.67	0.00	2.21
Geometric Mean			43294833.84	33810204.47	16.09	43294833.84	0.00	1.76
dfn-gwin	3	1	86800	71944.93	17.11	86800	0.00	0.50
	3	2	162160	144383.66	10.96	163560	0.86	0.70
	4	1	81120	61482.05	24.21	81120	0.00	0.55
	4	2	139360	126808.08	9.01	139360	0.00	0.58
	5	1	81120	59260	26.95	82240	1.38	0.64
	5	2	135120	121958.33	9.74	135440	0.24	0.59
Arithmetic Mean			114280	97639.51	16.33	114753.33	0.41	0.59
Geometric Mean			109714.20	91459.98	14.85	110166.31	0.00	0.59
polska	3	1	294200	279850	4.88	294200	0.00	0.16
	4	1	239600	195133.33	18.56	239600	0.00	0.34
	4	2	391700	391700	0.00	391700	0.00	0.14
	5	1	225100	167250	25.70	225100	0.00	0.75
	5	2	361500	338475	6.37	361500	0.00	0.47
Arithmetic Mean			302420	274481.66	11.10	302420	0.00	0.37
Geometric Mean			295333.94	260988.04	0.00	295333.94	0.00	0.31
nobel-us	3	1	16463000	16327000	0.83	16463000	0.00	0.30
	4	1	12322000	11430500	7.24	12322000	0.00	1.03
	5	1	10430000	7570000	27.42	10430000	0.00	2.97
	5	2	16649000	15897000	4.52	16649000	0.00	1.25
Arithmetic Mean			13966000	12806125	10.00	13966000	0.00	1.39
Geometric Mean			13699845.24	12241810.96	5.22	13699845.24	0.00	1.03

Table A.10: LP Relaxation. Integer optimal solution and heuristic performances on SNDlib test set.

Instances			Number of cuts generated				Number of nodes visited					
Name	**L**	**K**	bc-all	bc-int	bc-5	bc-5-heur	bc-all	bc-int	bc-5	bc-5-heur	layered	layered-r
TC-5	3	1	121	64	107	56	15	21	15	125	1	1
	3	1	290	233	137	172	102	538	362	673	22	55
	3	1	54	50	51	30	13	17	14	35	1	1
	3	1	180	93	143	73	37	100	44	209	1	24
	3	1	309	141	146	85	47	150	108	478	1	30
	3	2	57	57	57	52	1	1	1	125	1	1
	3	2	55	55	55	70	3	3	3	82	1	1
	3	2	37	37	37	25	3	3	3	3	1	1
	3	2	73	37	73	46	4	10	4	47	1	1
	3	2	74	65	74	64	6	24	6	64	1	1
	3	3	160	98	104	83	38	151	168	387	20	66
	3	3	261	163	154	113	51	468	108	1651	16	45
	3	3	41	36	41	26	3	4	3	5	1	1
	3	3	115	79	72	82	21	80	58	241	1	9
	3	3	168	66	90	76	43	31	46	302	5	10
	4	1	238	112	162	57	34	94	79	152	7	20
	4	1	1318	303	288	271	393	3087	2722	6426	607	821
	4	1	156	58	95	32	37	40	61	57	1	1
	4	1	269	223	148	101	45	446	148	395	68	43
	4	1	840	392	223	182	166	1281	347	1024	131	137
	4	2	87	94	87	89	7	60	7	372	1	4
	4	2	287	109	204	178	20	117	28	1798	13	15
	4	2	57	58	57	37	8	24	8	41	1	2
	4	2	237	127	149	126	38	225	117	462	11	21
	4	2	338	232	173	115	50	537	91	548	28	32
	4	3	133	65	76	52	55	68	62	267	2	51
	4	3	1362	182	285	241	817	2724	1982	11283	743	451
	4	3	63	125	63	36	8	88	8	21	1	2
	4	3	133	84	106	99	35	127	112	694	8	29
	4	3	243	75	293	156	113	146	497	803	51	86
	5	1	85	95	83	104	14	63	15	219	1	24
	5	1	2789	221	518	357	1263	1535	2236	5490	508	1423
	5	1	88	72	73	42	29	80	28	132	1	4
	5	1	414	185	130	102	188	418	215	398	33	120
	5	1	1041	324	331	161	177	799	475	1079	73	104
	5	2	45	45	45	50	1	1	1	28	1	1
	5	2	65	65	65	69	1	1	1	109	1	1
	5	2	46	46	46	37	1	1	1	10	1	1
	5	2	48	48	48	42	1	1	1	12	1	1
	5	2	38	38	38	35	1	1	1	2	1	1
	5	3	56	89	49	46	58	134	67	423	11	85
	5	3	2246	323	234	179	1707	3804	3233	3712	2830	2728
	5	3	25	23	25	32	5	6	5	6	1	2
	5	3	121	154	79	64	76	217	120	124	21	42
	5	3	571	174	255	117	352	530	428	777	91	210
Arithmetic Mean			342.98	120.33	128.20	94.71	135.27	405.69	311.98	917.58	118.27	149.09
Geometric Mean			155.97	94.80	100.56	75.74	23.86	60.59	37.00	186.67	6.11	12.80

Table A.11: Number of cuts generated and of nodes visited for the TC test set.

Instances			Number of cuts generated				Number of nodes visited					
Name	**L**	**K**	bc-all	bc-int	bc-5	bc-5-heur	bc-all	bc-int	bc-5	bc-5-heur	layered	layered-r
TC-10	3	1	1290	228	398	178	492	1633	1082	3871	239	297
	3	1	3302	539	1059	333	925	8460	3624	11000	1369	1590
	3	1	3072	397	711	233	701	2674	1476	5711	612	498
	3	1	744	222	304	181	251	805	584	3201	156	284
	3	1	2925	425	751	222	455	2813	895	1656	540	354
	3	2	869	313	353	244	232	1996	907	4628	299	256
	3	2	2212	773	573	421	796	4788	1599	21801	1889	1158
	3	2	1276	404	277	234	260	2070	960	5359	1314	272
	3	2	539	196	242	148	97	461	184	1184	84	62
	3	2	910	246	806	259	153	1747	360	676	467	185
	3	3	258	187	211	155	39	159	65	742	31	17
	3	3	448	164	218	169	66	354	168	1361	71	36
	3	3	357	274	223	151	61	467	84	3444	116	61
	3	3	1703	336	365	357	596	5608	3865	15591	2350	1266
	3	3	538	236	257	146	72	883	124	1981	103	128
	4	1	7523	1122	896	483	1421	10519	4713	22545	2016	1676
	4	1	28104	1213	1685	666	4718	41053	19641	40389	3421	3606
	4	1	23199	678	1563	511	3268	25712	11301	37470	6906	2479
	4	1	5958	463	696	239	964	2027	1663	6651	408	716
	4	1	15790	656	1730	549	2024	10357	4505	5952	1981	1629
	4	2	2364	680	396	513	347	1939	1757	16378	819	215
	4	2	2435	960	1008	342	161	3602	474	8933	358	175
	4	2	1784	776	909	494	104	2613	1272	28527	488	348
	4	2	1067	318	413	350	158	633	662	9173	321	167
	4	2	4526	1048	2052	593	421	8432	3117	19174	3490	960
	4	3	1737	283	401	324	451	1982	1180	14693	492	299
	4	3	1374	184	242	361	206	684	154	9423	378	149
	4	3	3814	578	486	529	1092	4627	3579	38154	2009	255
	4	3	5846	567	744	325	775	2989	1138	18579	793	338
	4	3	6039	541	868	351	680	9380	2132	37832	1383	677
	5	1	25517	519	1353	379	4309	11005	11309	16727	2014	2439
	5	1	33330	1706	2333	744	1999	41307	25171	43063	5037	2467
	5	1	29744	928	1926	535	2045	47127	19753	48822	3505	1952
	5	1	10980	721	1339	369	1267	3502	3706	7845	689	962
	5	1	30308	1427	2431	607	2054	12473	11179	19505	2186	2527
	5	2	401	351	304	344	19	395	44	748	148	28
	5	2	2325	540	484	489	216	1814	639	4576	265	181
	5	2	1687	516	720	578	102	1146	406	6661	606	73
	5	2	4153	790	708	479	318	2779	821	4664	1257	329
	5	2	9112	1224	561	1092	932	9087	2922	40691	2219	866
	5	3	4275	422	421	222	1251	5016	4597	3406	641	275
	5	3	1180	292	215	231	496	2302	998	5855	1097	355
	5	3	392	180	256	121	33	305	42	3266	33	16
	5	3	8324	397	788	398	1998	2864	3126	6928	590	225
	5	3	4014	531	446	377	827	4556	1565	19946	669	502
Arithmetic Mean			6616.56	567.80	780.49	378.36	885.60	6825.44	3545.40	13972.93	1241.31	741.11
Geometric Mean			2854.49	473.10	602.97	335.64	429.28	2897.46	1246.09	8010.60	643.91	356.19

Table A.12: Number of cuts generated and of nodes visited for the TC test set.

Instances			Number of cuts generated				Number of nodes visited					
Name	L	K	bc-all	bc-int	bc-5	bc-5-heur	bc-all	bc-int	bc-5	bc-5-heur	layered	layered-r
TE-5	3	1	772	305	339	125	71	515	244	787	64	48
	3	1	456	211	250	89	49	300	115	283	25	26
	3	1	679	253	345	203	98	918	359	1750	82	85
	3	1	632	278	228	149	67	681	323	778	12	36
	3	1	634	241	263	153	67	756	237	854	45	56
	3	2	192	189	192	104	14	177	14	835	6	15
	3	2	128	188	114	101	16	306	16	260	1	1
	3	2	732	420	316	242	76	1259	97	2773	140	57
	3	2	557	114	235	138	69	285	194	1210	89	55
	3	2	740	596	332	242	56	1091	156	532	246	68
	3	3	105	105	105	76	1	1	1	465	1	1
	3	3	89	89	89	42	1	1	1	88	1	1
	3	3	103	103	103	64	3	3	3	101	1	1
	3	3	85	85	85	65	1	1	1	157	1	1
	3	3	114	114	114	71	1	1	1	171	1	1
	4	1	1569	324	422	246	273	2456	757	4157	377	281
	4	1	814	241	362	100	121	502	409	244	71	53
	4	1	2268	411	379	286	594	4750	2287	10316	1267	1029
	4	1	1298	325	585	296	239	1790	873	1975	472	247
	4	1	3016	564	853	262	292	4237	1533	3560	218	221
	4	2	784	254	334	254	120	1793	558	2964	130	101
	4	2	175	139	125	139	35	275	75	427	24	28
	4	2	213	187	152	136	13	305	29	764	8	12
	4	2	288	203	205	122	24	879	62	571	6	16
	4	2	1367	429	442	372	106	2591	261	8346	101	83
	4	3	268	114	148	173	78	435	202	1342	62	45
	4	3	149	86	119	104	52	134	75	352	59	35
	4	3	1098	309	268	196	428	1989	587	11543	667	275
	4	3	190	93	123	105	41	236	75	725	5	34
	4	3	846	356	456	321	122	2402	563	8750	226	166
	5	1	1143	386	315	277	394	2930	1537	2176	952	565
	5	1	691	195	385	165	187	637	828	745	60	214
	5	1	3134	475	456	330	1097	5020	4085	11834	1678	1313
	5	1	1624	246	251	254	386	1336	1283	1109	533	300
	5	1	3237	590	537	244	477	3475	1259	2177	389	274
	5	2	48	48	48	42	1	1	1	10	1	1
	5	2	29	29	29	30	1	1	1	7	1	1
	5	2	70	70	70	57	1	1	1	56	1	1
	5	2	29	29	29	34	1	1	1	1	1	1
	5	2	45	45	45	54	1	1	1	2	1	1
	5	3	164	143	146	148	75	636	401	1497	103	89
	5	3	145	139	87	110	100	577	167	410	77	128
	5	3	2018	199	218	146	1259	3797	2141	9602	499	1639
	5	3	433	138	92	88	300	458	329	268	156	148
	5	3	900	590	387	233	210	2523	1280	2220	454	237
Arithmetic Mean			757.13	236.62	248.40	159.73	169.29	1165.84	520.51	2204.31	206.98	177.56
Geometric Mean			383.61	183.57	190.15	132.80	41.15	210.13	89.04	606.29	34.19	33.68

Table A.13: Number of cuts generated and of nodes visited for the TE test set.

Instances			Number of cuts generated				Number of nodes visited					
Name	**L**	**K**	bc-all	bc-int	bc-5	bc-5-heur	bc-all	bc-int	bc-5	bc-5-heur	layered	layered-r
TE-10	3	1	6661	917	1769	1667	716	13046	2772	2668	1160	1856
	3	1	6852	920	1511	575	485	6011	1486	1578	1114	1112
	3	1	6477	753	1500	744	482	8764	1484	1268	976	404
	3	1	10390	1030	1371	461	924	10510	2999	5515	2138	1804
	3	1	5039	891	1505	378	786	8660	3120	11407	1641	1190
	3	2	11485	2023	2855	1636	1332	92849	7762	13696	9708	2385
	3	2	9756	1716	2253	1035	1417	45647	6088	8839	3530	1926
	3	2	5638	1409	1545	653	641	11165	1883	1271	1562	561
	3	2	13433	1351	1798	706	2016	70434	12605	52042	6843	2878
	3	2	2751	830	1494	378	287	5165	2850	16610	1715	474
	3	3	12292	1941	2309	1537	1748	95626	4210	6906	13709	2088
	3	3	5846	1086	1278	624	1144	33730	2886	9831	3563	1822
	3	3	5114	1676	1399	519	834	17392	1541	2959	2146	1034
	3	3	7522	1243	1275	615	1192	44588	7234	31767	3970	1446
	3	3	5689	713	539	534	1290	14871	5167	28738	2165	1584
	4	1	30957	1973	3501	2288	1326	88712	32481	45962	6602	5588
	4	1	31535	1961	3293	1452	1505	83761	23709	22477	6784	5848
	4	1	32739	1785	3029	1357	1483	35260	7741	4033	2483	1872
	4	1	28310	1723	2464	802	2352	71500	20722	103389	5404	5713
	4	1	22040	943	1669	524	2029	21717	6690	30711	2000	3925
	4	2	33125	4385	5203	1993	1100	101135	62178	38088	6008	4217
	4	2	32862	3216	4244	1845	1314	155102	26122	65747	5582	5624
	4	2	29559	2676	2974	1542	1793	85121	11835	18058	3614	2103
	4	2	32188	4224	2312	1932	1800	103568	22544	433473	5230	4967
	4	2	11189	1092	2008	800	864	20603	2872	54131	2442	1620
	4	3	26417	1873	2283	943	2830	53887	27825	46390	5014	3281
	4	3	13921	1259	1177	705	1752	43213	5246	34237	4365	1879
	4	3	18849	1028	758	1085	2875	73053	21201	8403	8201	3343
	4	3	7100	743	742	933	1062	22296	8927	163699	2220	1296
	4	3	8038	772	1096	885	1718	35773	8053	120197	4094	2987
	5	1	31786	1908	3560	2263	1102	118660	55963	108718	2352	1903
	5	1	31862	2373	3204	1151	1216	45218	36241	22003	2202	1781
	5	1	32879	1823	3134	1364	1149	31816	23704	28575	2071	1747
	5	1	33463	2202	2086	632	1497	51944	34249	36459	2479	1966
	5	1	29847	1148	2218	571	2309	19159	12382	25503	1655	2790
	5	2	31835	3963	3781	1833	1041	157377	47453	106954	1932	2108
	5	2	32243	4083	2915	1682	1259	139511	25763	71831	1967	1747
	5	2	16610	1863	1994	615	820	35098	3299	15201	1440	1168
	5	2	9002	3116	1484	1583	888	19298	3635	41534	2278	2090
	5	2	7951	2297	1065	1508	643	21656	2980	33075	2344	656
	5	3	25002	1424	1684	772	3311	45603	20242	39088	2262	1956
	5	3	740	442	237	309	74	1103	182	4741	572	57
	5	3	7626	687	1037	280	1246	10987	5241	2472	1912	1036
	5	3	3231	333	426	326	443	3200	1701	8455	878	292
	5	3	4103	398	395	286	855	8287	3985	15374	788	1222
Arithmetic Mean			17154.53	1693.60	2008.31	1029.40	1310.00	48490.58	13983.40	43201.62	3402.56	2207.69
Geometric Mean			12651.14	1419.50	1690.40	872.34	1112.66	30359.69	7559.88	19712.40	2674.17	1693.34

Table A.14: Number of cuts generated and of nodes visited for the TE test set.

Instances			Number of cuts generated				Number of nodes visited					
Name	**L**	**K**	bc-all	bc-int	bc-5	bc-5-heur	bc-all	bc-int	bc-5	bc-5-heur	layered	layered-r
pdh	3	1	1629	543	546	830	698	4139	2576	12644	2155	1387
	3	2	1945	780	626	782	2464	10656	4003	11273	7237	2030
	3	3	525	232	296	366	134	412	456	1108	368	225
	4	1	8493	1641	1759	1396	3138	11582	10364	15847	5389	2719
	4	2	7883	1510	1767	1572	4771	11403	11831	17612	8376	2911
	4	3	287	147	138	100	75	274	143	270	304	87
	5	1	28885	3508	3638	2358	10441	23236	24606	20351	9086	6267
	5	2	6674	2536	2007	2464	2091	8700	6908	7824	7226	2400
	5	3	112	76	76	61	63	80	77	39	40	63
Arithmetic Mean			6270.33	1219.22	1205.89	1103.22	2652.78	7831.33	6773.78	9663.11	4464.56	2009.89
Geometric Mean			2067.45	661.34	660.97	641.80	891.56	2801.64	2172.74	3475.06	1808.38	906.51
di-yuan	3	1	2615	827	587	870	505	1637	954	4611	2520	579
	3	2	1453	686	480	607	308	1469	1108	4872	2924	324
	3	3	640	364	342	277	71	355	120	996	872	116
	4	1	1920	754	1150	694	185	888	751	1181	575	436
	4	2	1962	716	693	871	459	1155	1015	1597	1562	416
	4	3	168	96	130	97	32	31	41	45	64	31
	5	1	2265	979	1295	771	173	701	592	668	794	160
	5	2	1051	781	585	639	160	543	358	559	1410	126
	5	3	71	64	68	48	5	5	5	2	105	6
Arithmetic Mean			1349.44	585.22	592.22	541.56	210.89	753.78	549.33	1614.56	1202.89	243.78
Geometric Mean			861.03	431.97	431.10	388.76	117.14	333.95	252.69	482.53	705.71	134.91
dfn-gwin	3	1	714	208	235	112	154	407	519	261	261	94
	3	2	1400	226	257	219	385	1006	1044	1516	717	326
	4	1	1417	400	332	219	372	1640	800	955	542	380
	4	2	1052	259	307	192	374	921	934	647	287	158
	5	1	2063	366	430	245	868	1444	1508	845	1062	1206
	5	2	682	279	300	218	337	758	650	499	923	296
Arithmetic Mean			1221.33	289.67	310.17	200.83	415.00	1029.33	909.17	787.17	632.00	410.00
Geometric Mean			1131.31	281.47	304.31	195.06	366.23	937.03	857.24	684.75	552.79	294.83
polska	3	1	78	78	78	50	7	7	7	10	1	1
	4	1	130	124	125	160	15	22	20	147	1	9
	4	2	64	64	64	33	1	1	1	1	1	1
	5	1	174	183	148	234	43	115	83	271	29	61
	5	2	104	102	104	48	14	12	14	15	9	2
Arithmetic Mean			110.00	110.20	103.80	105.00	16.00	31.40	25.00	88.80	8.20	14.80
Geometric Mean			103.27	102.93	99.20	78.42	9.12	11.63	11.02	22.66	3.04	4.06
nobel-us	3	1	155	155	155	53	3	3	3	2	1	1
	4	1	229	219	225	176	12	17	15	80	1	11
	5	1	360	428	344	625	48	160	87	878	177	43
	5	2	153	153	153	85	8	8	8	10	2	2
Arithmetic Mean			224.25	238.75	219.25	234.75	17.75	47.00	28.25	242.50	45.25	14.25
Geometric Mean			210.28	217.13	206.99	149.20	10.84	15.98	13.30	34.43	4.34	5.55

Table A.15: Number of cuts generated and of nodes visited for the SNDlib test set.

Instances			cp	cp-i	
Name	**L**	**K**	integer (MP)	linear (MP)	integer (MP)
TC-5	3	1	21	24	1
	3	1	95	29	16
	3	1	20	21	1
	3	1	24	13	8
	3	1	49	29	6
	3	2	30	21	1
	3	2	30	22	1
	3	2	19	14	1
	3	2	22	16	1
	3	2	45	27	3
	3	3	58	26	12
	3	3	82	22	12
	3	3	13	11	1
	3	3	36	16	10
	3	3	65	19	1
	4	1	24	29	2
	4	1	182	36	29
	4	1	27	18	1
	4	1	26	14	11
	4	1	62	20	15
	4	2	53	25	5
	4	2	103	23	8
	4	2	21	13	4
	4	2	55	11	30
	4	2	99	22	27
	4	3	63	19	1
	4	3	167	30	31
	4	3	18	13	2
	4	3	32	22	9
	4	3	115	23	36
	5	1	22	27	2
	5	1	127	33	56
	5	1	28	15	1
	5	1	27	11	9
	5	1	47	16	41
	5	2	43	24	1
	5	2	86	20	1
	5	2	18	18	1
	5	2	12	10	2
	5	2	16	13	1
	5	3	28	25	3
	5	3	187	22	63
	5	3	17	13	2
	5	3	48	19	9
	5	3	71	30	22
Arithmetic Mean			54.07	20.53	11.11
Geometric Mean			40.97	19.50	4.57

Table A.16: Number of iterations on TC test set.

Instances			cp	cp-i	
Name	**L**	**K**	integer (MP)	linear (MP)	integer (MP)
TC-10	3	1	52	21	9
	3	1	136	22	29
	3	1	92	24	16
	3	1	42	21	4
	3	1	83	24	8
	3	2	98	18	31
	3	2	104	22	52
	3	2	73	24	14
	3	2	41	18	16
	3	2	66	26	15
	3	3	64	23	5
	3	3	89	18	18
	3	3	58	23	4
	3	3	93	22	69
	3	3	85	27	3
	4	1	61	29	16
	4	1	97	30	39
	4	1	179	33	48
	4	1	38	28	3
	4	1	64	36	7
	4	2	141	28	16
	4	2	74	27	14
	4	2	108	26	14
	4	2	64	27	13
	4	2	107	37	16
	4	3	82	23	3
	4	3	73	22	5
	4	3	139	32	63
	4	3	89	29	3
	4	3	121	37	11
	5	1	62	40	25
	5	1	136	29	66
	5	1	168	49	60
	5	1	34	35	3
	5	1	78	37	29
	5	2	78	28	8
	5	2	151	26	28
	5	2	200	32	7
	5	2	140	39	23
	5	2	175	34	91
	5	3	111	27	49
	5	3	136	23	16
	5	3	64	38	1
	5	3	81	19	3
	5	3	138	37	21
Arithmetic Mean			97.00	28.22	22.09
Geometric Mean			88.88	27.43	13.58

Table A.17: Number of iterations on TC test set.

Instances			cp	cp-i	
Name	**L**	**K**	integer (MP)	linear (MP)	integer (MP)
TE-5	3	1	356	36	17
	3	1	145	28	11
	3	1	253	31	38
	3	1	317	35	3
	3	1	322	43	11
	3	2	192	34	1
	3	2	104	24	1
	3	2	192	36	37
	3	2	145	30	9
	3	2	165	38	7
	3	3	127	28	1
	3	3	71	27	1
	3	3	108	31	1
	3	3	75	25	1
	3	3	130	32	1
	4	1	319	41	141
	4	1	124	33	6
	4	1	339	35	99
	4	1	316	33	28
	4	1	346	42	22
	4	2	223	37	100
	4	2	311	33	9
	4	2	240	33	10
	4	2	197	29	10
	4	2	264	32	44
	4	3	145	37	10
	4	3	214	29	9
	4	3	178	46	32
	4	3	131	34	18
	4	3	222	39	25
	5	1	353	44	462
	5	1	152	39	16
	5	1	381	45	110
	5	1	386	35	34
	5	1	456	35	61
	5	2	269	32	1
	5	2	120	26	2
	5	2	376	40	2
	5	2	299	43	1
	5	2	255	31	1
	5	3	184	37	64
	5	3	215	32	23
	5	3	223	40	100
	5	3	155	33	22
	5	3	249	39	47
Arithmetic Mean			229.87	34.71	36.64
Geometric Mean			209.24	34.29	11.18

Table A.18: Number of iterations on TE test set.

Instances			cp	cp-i	
Name	**L**	**K**	integer (MP)	linear (MP)	integer (MP)
TE-10	3	1	147	34	41
	3	1	194	33	28
	3	1	163	33	17
	3	1	140	32	52
	3	1	161	31	36
	3	2	69	32	27
	3	2	85	31	41
	3	2	96	33	24
	3	2	73	37	33
	3	2	91	28	46
	3	3	55	30	41
	3	3	69	28	47
	3	3	77	34	49
	3	3	55	27	67
	3	3	77	30	71
	4	1	127	44	41
	4	1	175	55	44
	4	1	143	38	19
	4	1	132	41	57
	4	1	172	35	34
	4	2	100	47	30
	4	2	116	42	41
	4	2	122	35	43
	4	2	110	40	57
	4	2	125	38	59
	4	3	75	42	57
	4	3	98	36	67
	4	3	117	40	52
	4	3	67	32	102
	4	3	117	33	60
	5	1	146	52	66
	5	1	206	46	44
	5	1	170	45	63
	5	1	168	49	80
	5	1	131	38	35
	5	2	163	55	59
	5	2	156	41	73
	5	2	164	41	65
	5	2	156	43	97
	5	2	190	31	151
	5	3	158	51	101
	5	3	116	34	2
	5	3	151	28	47
	5	3	94	29	10
	5	3	166	37	42
Arithmetic Mean			126.29	37.58	51.51
Geometric Mean			119.25	36.90	43.88

Table A.19: Number of iterations on TE test set.

Instances			cp	cp-i	
Name	**L**	**K**	integer (MP)	linear (MP)	integer (MP)
pdh	3	1	302	16	127
	3	2	166	13	133
	3	3	100	15	19
	4	1	281	16	121
	4	2	171	15	174
	4	3	49	12	4
	5	1	228	15	164
	5	2	269	13	259
	5	3	6	5	2
Arithmetic Mean			174.67	13.33	111.44
Geometric Mean			117.73	12.73	50.91
di-yuan	3	1	261	13	40
	3	2	87	12	18
	3	3	54	11	3
	4	1	103	11	25
	4	2	80	9	39
	4	3	12	7	4
	5	1	66	13	22
	5	2	66	10	34
	5	3	3	3	1
Arithmetic Mean			81.33	9.89	20.67
Geometric Mean			48.85	9.18	12.27
dfn-gwin	3	1	64	22	6
	3	2	111	18	27
	4	1	136	18	28
	4	2	106	15	20
	5	1	49	14	32
	5	2	100	15	64
Arithmetic Mean			94.33	17.00	29.50
Geometric Mean			89.14	16.80	23.89
polska	3	1	54	13	1
	4	1	104	13	6
	4	2	17	8	1
	5	1	87	11	15
	5	2	21	12	1
Arithmetic Mean			56.60	11.40	4.80
Geometric Mean			44.50	11.23	2.46
nobel-us	3	1	63	13	1
	4	1	218	15	6
	5	1	282	9	77
	5	2	26	9	3
Arithmetic Mean			147.25	11.50	21.75
Geometric Mean			100.17	11.21	6.10

Table A.20: Number of iterations on SNDlib test set.

Instances			Cutting plane algorithms		Branch-and-cut algorithms				heuristic
Name	L	K	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	
TC-5	3	1	0.11	0.14	0.13	0.05	0.09	0.06	0.02
	3	1	0.39	0.25	0.63	0.44	0.25	0.20	0.11
	3	1	0.08	0.03	0.06	0.03	0.00	0.00	0.05
	3	1	0.06	0.06	0.20	0.06	0.16	0.05	0.05
	3	1	0.22	0.17	0.69	0.22	0.19	0.09	0.09
	3	2	0.14	0.09	0.05	0.05	0.05	0.03	0.00
	3	2	0.11	0.11	0.05	0.08	0.11	0.06	0.05
	3	2	0.05	0.08	0.03	0.02	0.00	0.02	0.03
	3	2	0.08	0.16	0.06	0.05	0.08	0.05	0.02
	3	2	0.13	0.02	0.08	0.06	0.11	0.09	0.05
	3	3	0.36	0.16	0.28	0.14	0.09	0.08	0.08
	3	3	0.33	0.11	0.41	0.22	0.19	0.22	0.05
	3	3	0.03	0.00	0.06	0.03	0.05	0.03	0.06
	3	3	0.08	0.09	0.19	0.13	0.09	0.17	0.06
	3	3	0.27	0.09	0.33	0.06	0.11	0.08	0.09
	4	1	0.16	0.22	1.91	0.69	1.20	0.41	0.25
	4	1	1.33	0.50	12.92	2.41	2.38	2.30	0.19
	4	1	0.14	0.13	1.34	0.20	0.63	0.14	0.09
	4	1	0.20	0.22	2.42	1.45	0.95	0.52	0.14
	4	1	0.30	0.31	7.64	2.61	1.64	1.31	0.17
	4	2	0.31	0.25	0.45	0.69	0.45	0.73	0.09
	4	2	0.63	0.22	2.27	0.95	1.88	1.48	0.08
	4	2	0.09	0.09	0.28	0.33	0.30	0.27	0.03
	4	2	0.31	0.25	2.08	1.11	1.16	1.02	0.09
	4	2	0.69	0.39	3.08	1.67	1.53	0.78	0.09
	4	3	0.41	0.13	1.42	0.47	0.63	0.42	0.11
	4	3	1.34	0.47	18.45	1.72	2.92	2.52	0.34
	4	3	0.11	0.16	0.41	0.83	0.41	0.16	0.08
	4	3	0.19	0.27	1.22	0.77	1.02	0.81	0.42
	4	3	0.89	0.47	2.53	0.59	2.42	1.27	0.14
	5	1	0.19	0.28	1.38	1.86	1.47	2.41	0.22
	5	1	1.38	0.95	102.59	4.78	14.38	9.70	0.41
	5	1	0.34	0.13	1.98	1.19	1.27	0.36	0.23
	5	1	0.22	0.16	13.53	3.64	2.83	2.02	0.27
	5	1	0.48	0.47	26.86	6.88	6.73	3.58	0.27
	5	2	0.33	0.22	0.73	0.75	0.69	0.45	0.06
	5	2	0.83	0.19	1.59	1.48	1.66	1.69	0.11
	5	2	0.14	0.22	0.59	0.59	0.61	0.58	0.09
	5	2	0.02	0.11	0.63	0.64	0.67	0.81	0.05
	5	2	0.16	0.13	0.98	0.95	1.02	0.38	0.00
	5	3	0.17	0.28	2.50	1.78	1.02	0.95	0.31
	5	3	2.02	0.78	126.86	11.00	7.91	7.11	1.97
	5	3	0.22	0.20	0.34	0.30	0.30	0.41	0.14
	5	3	0.58	0.34	5.23	4.00	2.05	1.48	0.50
	5	3	0.63	0.53	24.97	5.52	7.44	3.55	1.81
Arithmetic Mean			0.38	0.24	8.28	1.41	1.58	1.13	0.21
Geometric Mean			0.24	0.15	1.02	0.50	0.39	0.33	0.08

Table A.21: CPU time (sec.) spent for solving Benders subproblems on TC test set.

Instances			Cutting plane algorithms		Branch-and-cut algorithms				heuristic
Name	**L**	**K**	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	
TC-10	3	1	0.39	0.28	4.66	0.33	0.80	0.27	0.34
	3	1	1.00	0.38	10.83	1.11	2.81	0.69	0.34
	3	1	0.73	0.36	9.75	0.80	1.61	0.38	0.23
	3	1	0.22	0.23	2.16	0.30	0.48	0.27	0.56
	3	1	0.66	0.34	7.39	0.66	1.64	0.33	0.25
	3	2	0.78	0.52	2.66	0.63	0.78	0.52	0.25
	3	2	1.05	0.80	8.91	1.97	1.28	1.06	0.39
	3	2	2.25	0.56	2.88	0.58	0.30	0.41	0.25
	3	2	0.25	0.34	0.91	0.27	0.42	0.17	0.22
	3	2	0.70	0.27	1.64	0.34	1.22	0.34	0.39
	3	3	0.80	0.41	0.67	0.22	0.50	0.17	0.28
	3	3	0.89	0.30	0.89	0.27	0.34	0.31	0.31
	3	3	0.38	0.14	0.89	0.36	0.31	0.25	0.30
	3	3	0.81	0.98	5.48	0.70	0.81	1.23	0.63
	3	3	1.05	0.42	1.09	0.30	0.34	0.30	0.31
	4	1	0.63	0.67	94.91	9.98	8.66	4.36	3.19
	4	1	1.39	1.02	305.36	12.33	16.06	6.20	2.17
	4	1	2.41	1.25	234.48	7.19	15.70	4.41	1.89
	4	1	0.50	0.66	67.25	3.86	6.36	1.97	2.36
	4	1	0.83	0.72	166.55	5.16	15.48	4.19	2.66
	4	2	1.75	0.75	27.31	6.06	3.89	5.94	1.31
	4	2	0.97	0.61	19.61	8.36	9.14	3.22	1.89
	4	2	1.42	0.56	14.42	6.56	7.61	4.91	1.66
	4	2	0.91	0.73	10.64	2.72	3.78	3.31	1.66
	4	2	1.48	0.83	39.13	9.89	18.81	5.86	3.31
	4	3	1.38	0.38	22.53	2.08	3.72	3.50	1.17
	4	3	1.03	0.41	14.91	1.64	2.33	3.86	1.58
	4	3	2.09	1.55	57.39	7.03	5.75	7.17	1.91
	4	3	1.48	0.70	56.84	6.27	7.36	3.31	1.92
	4	3	1.73	0.88	53.34	4.55	8.14	4.31	2.06
	5	1	1.25	1.20	1008.38	14.00	51.89	10.20	5.84
	5	1	2.70	1.75	877.70	55.19	81.66	25.41	8.27
	5	1	3.00	2.28	906.17	33.17	72.59	18.41	16.95
	5	1	0.53	0.75	402.05	20.72	43.38	8.67	6.72
	5	1	1.22	1.45	876.33	40.06	78.92	19.06	6.81
	5	2	1.33	0.59	12.66	10.48	8.59	13.39	3.16
	5	2	2.55	1.17	65.16	21.09	13.72	20.64	6.78
	5	2	3.64	1.03	56.83	17.70	29.20	22.22	9.81
	5	2	2.56	1.39	118.36	29.89	23.58	18.53	4.41
	5	2	2.92	2.42	260.53	42.72	15.66	47.89	4.41
	5	3	2.33	1.61	246.86	18.80	19.53	8.20	2.97
	5	3	3.02	0.84	79.48	12.31	7.41	9.27	4.03
	5	3	1.39	1.09	11.45	5.05	6.91	3.78	3.08
	5	3	1.80	0.55	402.25	14.97	35.63	20.84	2.50
	5	3	3.11	1.27	163.86	20.38	15.64	16.44	7.44
Arithmetic Mean			1.45	0.83	149.63	10.20	14.46	7.47	2.87
Geometric Mean			1.19	0.69	28.32	3.75	5.08	2.87	1.51

Table A.22: CPU time (sec.) spent for solving Benders subproblems on TC test set.

Instances			Cutting plane algorithms		Branch-and-cut algorithms				heuristic
Name	L	K	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	
TE-5	3	1	1.47	0.19	1.50	0.44	0.72	0.16	0.16
	3	1	0.52	0.19	0.81	0.28	0.45	0.09	0.13
	3	1	0.80	0.34	1.41	0.42	0.73	0.33	0.05
	3	1	1.45	0.14	1.39	0.58	0.50	0.30	0.09
	3	1	1.19	0.28	1.30	0.42	0.39	0.28	0.11
	3	2	0.88	0.27	0.27	0.31	0.33	0.19	0.16
	3	2	0.52	0.09	0.25	0.34	0.14	0.22	0.08
	3	2	0.64	0.30	1.53	0.77	0.58	0.63	0.13
	3	2	0.63	0.28	1.28	0.08	0.42	0.19	0.08
	3	2	0.61	0.28	1.47	0.83	0.39	0.52	0.06
	3	3	0.47	0.14	0.13	0.16	0.14	0.17	0.00
	3	3	0.30	0.16	0.14	0.11	0.09	0.05	0.00
	3	3	0.63	0.11	0.13	0.14	0.09	0.08	0.09
	3	3	0.42	0.17	0.13	0.14	0.13	0.11	0.02
	3	3	0.61	0.13	0.16	0.19	0.11	0.11	0.00
	4	1	1.97	1.16	13.19	2.38	3.23	2.11	0.25
	4	1	0.70	0.25	6.55	1.77	3.09	0.69	0.22
	4	1	2.13	0.81	18.59	3.38	3.53	2.95	0.41
	4	1	2.06	0.45	11.13	2.63	4.58	2.14	0.23
	4	1	2.33	0.39	23.20	4.33	7.58	1.98	0.19
	4	2	1.38	0.88	6.27	2.19	2.47	2.39	0.17
	4	2	2.02	0.31	1.45	1.23	0.88	1.00	0.19
	4	2	1.47	0.30	1.72	1.47	1.09	0.98	0.17
	4	2	1.23	0.19	2.31	1.66	1.61	1.00	0.13
	4	2	1.70	0.48	11.81	3.70	3.78	3.66	0.16
	4	3	0.97	0.48	2.84	0.80	0.97	1.61	0.19
	4	3	1.53	0.20	1.75	0.73	0.98	0.97	0.16
	4	3	1.22	0.69	12.09	2.63	2.41	2.22	0.58
	4	3	0.95	0.45	1.77	0.89	1.05	0.80	0.30
	4	3	1.67	0.39	8.58	3.08	4.23	3.75	0.28
	5	1	3.20	4.44	36.48	9.19	7.89	7.02	0.31
	5	1	1.34	0.55	21.39	4.70	9.42	3.70	0.11
	5	1	3.48	1.59	89.94	12.20	11.77	10.50	0.42
	5	1	3.27	0.64	43.77	5.66	6.31	5.61	0.27
	5	1	3.98	0.89	90.08	12.67	13.98	5.80	0.23
	5	2	2.59	0.36	0.61	0.56	0.53	0.55	0.03
	5	2	1.14	0.31	0.30	0.23	0.30	0.38	0.06
	5	2	3.44	0.36	1.42	1.33	1.39	1.27	0.08
	5	2	2.70	0.36	0.39	0.42	0.42	0.47	0.03
	5	2	2.39	0.31	0.91	0.81	0.94	1.13	0.03
	5	3	1.70	0.97	5.88	4.14	5.02	5.08	0.97
	5	3	2.41	0.48	5.61	4.66	2.44	3.44	0.78
	5	3	2.25	1.44	106.02	7.38	7.00	5.17	2.73
	5	3	1.55	0.63	23.59	4.53	3.11	2.84	0.78
	5	3	2.56	1.13	37.02	21.03	14.02	9.94	2.22
Arithmetic Mean			1.61	0.55	13.30	2.84	2.92	2.10	0.31
Geometric Mean			1.33	0.39	2.87	1.19	1.24	0.93	0.09

Table A.23: CPU time (sec.) spent for solving Benders subproblems on TE test set.

Instances			Cutting plane algorithms		Branch-and-cut algorithms				heuristic
Name	L	K	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	
TE-10	3	1	1.03	0.75	13.88	2.09	3.56	3.59	1.23
	3	1	1.45	0.58	14.22	1.73	2.94	1.17	1.06
	3	1	1.25	0.41	12.75	1.69	2.91	1.78	0.91
	3	1	1.38	0.78	26.00	2.30	3.03	0.94	1.67
	3	1	1.17	0.64	13.75	1.92	3.06	0.91	0.56
	3	2	0.55	0.67	26.34	5.55	5.33	3.69	3.06
	3	2	0.47	1.08	24.66	4.19	4.84	2.23	1.33
	3	2	0.69	0.53	13.20	3.09	3.19	1.28	0.73
	3	2	0.45	0.63	38.48	3.92	4.34	2.22	1.30
	3	2	0.59	0.72	6.98	1.61	3.50	1.09	1.22
	3	3	0.56	0.73	29.58	5.33	4.56	3.45	3.00
	3	3	0.59	1.06	16.56	2.34	2.20	1.67	1.00
	3	3	0.66	0.70	14.95	3.92	3.11	0.97	1.22
	3	3	0.50	1.06	22.98	3.77	3.03	1.97	1.31
	3	3	0.70	0.77	20.89	2.02	1.11	1.53	1.02
	4	1	1.67	1.36	178.89	17.48	28.33	19.77	3.89
	4	1	2.17	1.83	224.11	17.38	27.03	11.86	5.31
	4	1	2.20	0.95	217.16	15.47	26.61	10.70	3.25
	4	1	1.67	1.59	250.78	15.17	21.89	8.16	2.36
	4	1	2.17	0.97	192.19	8.28	17.20	5.34	2.61
	4	2	1.47	1.44	175.86	44.02	50.08	16.48	4.78
	4	2	1.81	1.41	177.08	37.00	35.56	16.66	5.67
	4	2	1.83	1.11	192.88	26.02	24.88	10.11	3.25
	4	2	1.48	1.50	229.66	46.16	22.63	25.11	2.81
	4	2	1.58	1.48	97.31	10.98	16.91	8.83	2.20
	4	3	1.11	1.75	233.11	18.31	22.55	11.27	9.36
	4	3	1.38	1.59	126.25	12.67	9.19	8.34	4.16
	4	3	1.86	1.59	191.28	13.14	7.27	10.39	7.14
	4	3	1.06	2.77	77.22	8.77	8.17	13.78	1.25
	4	3	1.58	1.75	111.45	11.13	12.38	12.30	3.17
	5	1	2.28	2.56	619.03	58.77	107.53	68.16	10.36
	5	1	3.66	1.78	709.06	74.36	104.94	34.14	8.25
	5	1	3.11	2.19	689.56	50.95	93.53	35.31	9.41
	5	1	2.92	2.50	772.11	64.22	66.86	18.80	8.39
	5	1	2.33	1.34	815.27	33.67	74.41	16.06	5.44
	5	2	2.84	2.66	578.84	171.73	122.83	74.42	11.83
	5	2	3.06	2.44	557.52	167.64	88.47	60.45	14.44
	5	2	3.05	2.25	337.14	79.09	46.81	24.05	11.64
	5	2	2.77	2.89	296.34	111.63	46.72	71.03	6.38
	5	2	3.45	3.41	243.23	96.80	38.09	67.97	5.02
	5	3	2.77	3.34	855.28	80.80	73.13	45.33	10.59
	5	3	2.25	0.83	27.06	16.33	7.53	13.91	2.83
	5	3	2.92	1.67	285.36	28.03	36.36	9.30	10.77
	5	3	1.80	0.89	119.63	14.34	14.92	13.25	3.27
	5	3	3.20	1.88	197.08	20.81	18.28	13.95	5.47
Arithmetic Mean			1.77	1.49	223.84	31.48	29.37	17.42	4.58
Geometric Mean			1.50	1.29	103.46	13.84	14.51	8.42	3.23

Table A.24: CPU time (sec.) spent for solving Benders subproblems on TE test set.

Instances			Cutting plane algorithms		Branch-and-cut algorithms				heuristic
Name	**L**	**K**	cp	cp-i	bc-all	bc-int	bc-5	bc-5-heur	
pdh	3	1	3.00	1.38	4.58	0.42	0.50	0.44	0.84
	3	2	1.80	1.59	12.28	0.72	0.69	0.61	0.77
	3	3	1.23	0.56	1.30	0.13	0.42	0.11	0.39
	4	1	3.61	1.83	50.34	2.69	3.06	2.30	3.86
	4	2	2.63	3.03	68.22	2.95	3.91	2.91	4.09
	4	3	0.91	0.30	1.23	0.45	0.44	0.08	0.50
	5	1	3.94	3.25	321.58	12.53	13.36	8.20	12.27
	5	2	5.44	5.23	67.22	11.17	8.86	10.45	7.97
	5	3	0.14	0.13	1.17	0.22	0.36	0.08	0.11
Arithmetic Mean			2.52	1.92	58.66	3.48	3.51	2.80	3.42
Geometric Mean			1.77	1.18	12.57	1.18	1.47	0.78	1.41
di-yuan	3	1	4.23	0.97	8.20	0.66	0.50	0.67	1.06
	3	2	2.09	0.67	4.39	0.83	0.47	0.47	1.11
	3	3	1.16	0.52	1.25	0.22	0.30	0.22	0.42
	4	1	2.38	0.89	7.45	1.03	2.70	0.84	1.48
	4	2	2.34	1.34	11.66	1.23	1.52	1.50	1.66
	4	3	0.39	0.42	0.77	0.14	0.47	0.16	0.34
	5	1	2.06	1.16	13.91	2.67	4.42	2.23	3.14
	5	2	2.23	1.64	7.77	3.52	2.66	2.14	2.03
	5	3	0.11	0.14	0.28	0.13	0.22	0.03	0.09
Arithmetic Mean			1.89	0.86	6.19	1.16	1.47	0.92	1.26
Geometric Mean			1.31	0.71	3.56	0.67	0.89	0.50	0.85
dfn-gwin	3	1	0.20	0.11	0.61	0.06	0.11	0.03	0.19
	3	2	0.42	0.19	1.44	0.17	0.09	0.05	0.17
	4	1	0.69	0.27	2.19	0.64	0.44	0.19	0.31
	4	2	0.47	0.23	2.33	0.41	0.69	0.23	0.36
	5	1	0.38	0.36	8.31	0.92	1.13	0.72	0.58
	5	2	0.67	0.75	3.02	0.86	0.95	0.73	0.50
Arithmetic Mean			0.47	0.32	2.98	0.51	0.57	0.33	0.35
Geometric Mean			0.44	0.27	2.20	0.36	0.38	0.18	0.32
polska	3	1	0.36	0.14	0.03	0.02	0.03	0.05	0.05
	4	1	0.97	0.20	0.17	0.16	0.16	0.22	0.09
	4	2	0.20	0.06	0.09	0.08	0.13	0.02	0.03
	5	1	1.17	0.28	0.58	0.50	0.41	0.75	0.52
	5	2	0.30	0.19	0.33	0.30	0.25	0.08	0.19
Arithmetic Mean			0.60	0.17	0.24	0.21	0.20	0.22	0.18
Geometric Mean			0.48	0.15	0.15	0.13	0.14	0.11	0.11
nobel-us	3	1	0.89	0.14	0.13	0.09	0.17	0.02	0.08
	4	1	5.42	0.67	0.56	0.41	0.50	0.36	0.48
	5	1	8.86	2.61	2.94	2.59	2.08	4.77	2.17
	5	2	0.91	0.50	0.83	0.81	0.83	0.39	0.42
Arithmetic Mean			4.02	0.98	1.12	0.98	0.90	1.39	0.79
Geometric Mean			2.50	0.59	0.65	0.53	0.62	0.34	0.43

Table A.25: CPU time (sec.) spent for solving Benders subproblems on SNDlib test set.

Université catholique de Louvain
Faculté des sciences économiques, sociales et politiques
Louvain School of Management
Center of Excellence for Supply Chain Management

Place des Doyens 1, 1348 Louvain-la-Neuve, Belgique
Tél. : + 32 (0) 10 47 83 83 - fax : + 32 (0) 10 47 83 24
E-mail : quentin.botton@uclouvain.be
http://www.uclouvain.be/lsm.html

Nowadays our worldwide society depends largely on the performance of huge information systems more than often organized in large-scale, complex and costly networks. With time, the material (routers, fiber optic cables ...) worsens and the risk of failure is something that networks managers must control as well as possible in order to guarantee the best service to users. As a consequence, the development of survivable networks becomes a crucial field of research and investigation. To ensure survivability we suggest to find for each demand, at least K edge-disjoint paths (paths that cannot share the same group of edges). It guarantees that, if a maximum of (K-1) edges break down simultaneously, it is always possible to reroute all the demands since one feasible path remains functional.

However, new applications like video, audio or IP telephony... require small transmission delays. As a consequence, the survivability constraints are not sufficient to guarantee a routing with a good quality of service since the primary path where the traffic flows or the secondary paths, if a path fails, may lead to unacceptable delays. We impose a limit on the number of arcs of the routing paths, so that the traffic may be routed on a different path with a quality of service guaranteed in terms of delay.

After he got his degree in business engineering (UCL), Quentin Botton started his professional career as teaching-assistant in the logistic and modelling department of the Louvain School of Management. In 2007-2008, thanks to two grants (Communauté française de Belgique – ADONET) Quentin spent 6 months abroad and collaborated with professor Luis Gouveia from the operational research center of the University of Lisbon (Portugal). His field of interests is composed of combinatorial optimization, survivable network design optimization, polyhedral analysis, Benders decomposition, branch-and-cut algorithms...

629 | 2010

Quentin Botton

Survivable network design with quality of service constraints

UCL