HEC MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Models and algorithms for two network design problems

par
Alysson Machado Costa

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Ph.D. en administration

Novembre, 2006

© Alysson Machado Costa

HEC MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Cette thèse intitulée:

Models and algorithms for two network design problems

présentée par

Alysson Machado Costa

a été evaluée par un jury composé des personnes suivantes:

Michèle Breton
président-rapporteur

Jean-François Cordeau
co-directeur de recherche

Gilbert Laporte
co-directeur de recherche

Bernard Gendron
membre du jury

Luís Gouveia
examinateur externe

Robert Clark
représentant du directeur

Résumé

Les problèmes de conception de réseaux constituent une classe importante de problèmes d'optimisation se posant dans une grande variété de domaines. Cette thèse, composée de deux parties, propose de nouveaux modèles et algorithmes pour deux de ces problèmes: le problème de conception de réseaux avec capacités, coûts fixes, et plusieurs produits et le problème de Steiner avec profits.

La première partie est consacrée au problème de conception de réseaux avec capacités, coûts fixes et plusieurs produits. Celui-ci consiste à sélectionner des arêtes et à déterminer des flots dans un graphe de manière à satisfaire une certaine combinaison de demandes. Nous présentons une revue des modèles les plus courants ainsi que des algorithmes basés sur la décomposition de Benders. Nous étudions aussi les interrelations entre trois groupes d'inégalités valides pour ce problème: les inégalités de Benders, les inégalités de séparation d'ensembles et les inégalités métriques. À partir de ces relations, nous proposons une procédure simple pour renforcer les coupes d'admissibilité de Benders au moyen de calculs rapides de plus courts chemins. Nous proposons aussi un nouvel algorithme basé sur la décomposition de Benders pour ce problème, lequel utilise une procédure originale pour générer des coupes additionnelles.

La deuxième partie est consacrée au problème de Steiner avec profits. Ce problème constitue une variante du problème de Steiner dans laquelle il n'y a pas nécessairement de sommets obligatoires, mais où des revenus sont associés à chaque sommet du graphe. Nous présentons un survol et une classification des modèles et des algorithmes qui ont été proposés pour des différentes versions de ce problème. Cette revue inclut la présentation des procédures de réduction de la taille des instances. Nous introduisons une variante du problème qui comporte une contrainte de budget limitant le coût total du réseau, ainsi que des contraintes d'éloignement restreignant le nombre d'arêtes reliant n'importe quel sommet à la racine. Pour cette variante, nous développons et comparons d'abord de nouveaux modèles, puis des méthodes exactes qui permettent de résoudre à l'optimalité des instances de taille moyenne. De plus, nous présentons des procédures heuristiques dont

une procédure gloutonne et une méthode de recherche avec tabous. Finalement, des tests confirment l'efficience des méthodes proposées.

Mots clés: Conception de réseaux; problème de Steiner; coûts fixes; inégalités valides; collecte de prix; contraintes d'éloignement; décomposition de Benders; algorithme de séparation et coupes; heuristiques.

Abstract

Network design problems constitute an important class of optimization problems arising in a wide variety of real-life situations. This two-part thesis proposes new models and algorithms for two network design problems: the multicommodity capacitated fixed-charge network design problem and the Steiner tree problem with profits.

Part I is devoted to the multicommodity capacitated fixed-charge network design problem which consists of selecting edges and flows in a graph in order to supply certain demands while respecting edge capacities. We present an overview of the most common models, as well as a survey of associated algorithms based on Benders decomposition. We also study the interrelations between three important groups of valid inequalities for this problem: Benders, metric and cutset inequalities. Based on these relationships, we propose a simple procedure to strengthen Benders inequalities by means of quick shortest path calculations. We also develop a new Benders decomposition algorithm to solve the problem, which makes use of an original procedure for generating extra cuts.

Part II is dedicated to the Steiner tree problem with profits, a variant of the Steiner problem in which there may be no mandatory vertices, but where revenues are associated with each vertex of the graph. We present an overview and a classification of the models and algorithms that have been proposed for several variants of this problem. The review includes a survey of the preprocessing procedures used to reduce the size of the instances. We introduce a new variant of the problem which includes a budget constraint limiting the total network cost, and hop constraints limiting the maximum number of edges between the root vertex and any other vertex in the graph. For this variant, we first develop and compare new models and exact algorithms capable of solving mid-sized instances to optimality. We then present several heuristic procedures, including a greedy algorithm and a tabu search heuristic. Computational tests confirm the efficiency of the proposed methods.

Keywords: Network design; Steiner tree problem; fixed-charge; valid inequalities; prize collecting; hop constraints; Benders decomposition; branch-and-cut; heuristics.

Resumo

Os problemas de projeto de redes constituem uma classe importante dentre os problemas de otimização, aparecendo em uma vasta gama de situações práticas. Esta tese, dividida em duas partes, propõe novos modelos e algoritmos para dois problemas de projeto de redes: o problema de projeto de redes capacitadas, multi-produto e com custo fixo e o problema de Steiner com ganhos.

A primeira parte da tese é dedicada ao problema de projeto de redes capacitadas, multi-produto e com custo fixo, que consiste em selecionar arestas e fluxos em um grafo, de maneira a suprir uma certa demanda. É apresentada uma coletânea dos modelos mais comuns, bem como dos métodos de resolução baseados na decomposição de Benders propostos. São estudadas as inter-relações entre três grupos importantes de inequações válidas: as inequações de Benders, inequações de separação de conjuntos (*cutset*), e as inequações métricas. A partir dessas relações, um método simples para fortalecer os cortes de factibilidade de Benders é proposto, baseado em cálculos rápidos de caminho mais curto. Também é proposto um novo algoritmo de decomposição de Benders para o problema, que utiliza uma idéia original para geração de cortes adicionais.

A segunda parte da tese lida com o problema de Steiner com ganhos, uma variação do problema de Steiner na qual não necessariamente existem vértices obrigatórios, mas onde existe um ganho associado a cada vértice do grafo. São apresentadas uma revisão e uma classificação dos modelos e algoritmos que foram propostos para esta família de problemas, incluindo os métodos de pre-processamento usados para reduzir o tamanho das instâncias. É definida uma nova variante do problema que inclui uma restrição de orçamento que limita o custo da solução, bem como restrições de separação, limitando o número de arestas entre um vértice qualquer da solução e o vértice raiz. Para esta variante, primeiro são desenvolvidos e comparados novos modelos e algoritmos exatos que são capazes de obter soluções ótimas para instâncias de médio porte. São apresentados, então, vários métodos heurísticos, incluindo um método guloso e um algoritmo de busca tabu. Testes computacionais sugerem a eficiência dos métodos propostos.

Palavras-chave: Projeto de redes; problema de Steiner; custos fixos; inequações válidas; coleta de prêmios, restrições de separação; decomposição de Benders; algoritmo de separação e cortes; heurísticas.

Contents

Resumé **i**

Abstract **iii**

Resumo **iv**

List of tables **x**

List of figures **xii**

List of abbreviations **xiii**

Acknowledgements **xiv**

0 Introduction **1**

0.1 Part I: Multicommodity capacitated fixed-charge network design 2

0.2 Part II: Steiner tree problems with profits 3

I Multicommodity fixed-charge network design **5**

1 A survey on Benders decomposition applied to fixed-charge network design **6**

1.1 Introduction 6

1.2 Benders decomposition 8

1.3 Literature review 12

1.3.1 Uncapacitated network design problems (UNDP) 13
1.3.2 Capacitated network design problems (CNDP) 20
1.4 Conclusions 28

2 Benders, metric and cutset inequalities for multicommodity capacitated network design 32

2.1 Introduction 33
2.2 Benders and metric inequalities 36
2.3 Cutset inequalities 40
2.4 Computational results 44
2.4.1 Cutset inequalities 45
2.4.2 Benders inequalities 47
2.5 Conclusions 52

3 Benders decomposition applied to multicommodity capacitated fixed-charge network design 53

3.1 Introduction 53
3.2 Benders decomposition 56
3.3 Algorithm refinements 57
3.3.1 Relaxed master problem 58
3.3.2 Cutset inequalities 58
3.3.3 Generating Pareto-optimal cuts 58
3.3.4 Additional bounds on the flow variables 59
3.3.5 Strengthening the feasibility cuts and the cutset inequalities 60
3.3.6 Reducing the number of *integer* iterations 60
3.4 Computational results 65
3.5 Conclusions 68

II Steiner tree problems with profits 69

4 A survey of Steiner tree problems with profits 70

4.1 Introduction 70
4.2 Applications 71
4.3 Classification of the problems 72
4.3.1 Node-weighted and prize-collecting Steiner tree problems 72
4.3.2 The quota STPP 77
4.3.3 The budget STPP 77
4.3.4 The fractional STPP 78
4.4 Reduction tests 79
4.4.1 Description of the reduction tests 79
4.4.2 Validity for the quota and budget STPP 81
4.4.3 Validity for the fractional STPP 83
4.4.4 New tests for the budget STPP 83
4.4.5 Summary of the reduction tests 84
4.5 A comparison with the TSP with profits 85
4.6 Conclusions 85

5 Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints 89

5.1 Introduction 89
5.2 Formulations 92
5.2.1 Dantzig-Fulkerson-Johnson formulations 93
5.2.2 Miller-Tucker-Zemlin formulation 95
5.2.3 Garcia-Gouveia Hop formulation 97
5.3 Branch-and-cut Algorithms 104
5.3.1 Branch-and-cut for formulation DDFJ 105
5.3.2 Branch-and-cut for formulation GG-Hop 107
5.3.3 Summary of the algorithms and branch-and-cut template 108
5.4 Computational Experiments 108
5.4.1 Test instances 108
5.4.2 Complete tests 110

5.5 Conclusions 114

6 Fast heuristics for the Steiner tree problem with revenues, budget and hop constraints 126

6.1 Introduction 126

6.2 Mathematical formulation 128

6.3 Greedy algorithm 129

6.3.1 Finding feasible paths 129

6.3.2 Evaluating the paths 131

6.4 Destroy-and-repair algorithm 133

6.5 Tabu search algorithm 134

6.5.1 Add move 136

6.5.2 Remove move 136

6.5.3 Destroy move 136

6.6 Computational experiments 139

6.7 Conclusions 141

7 Conclusions 152

7.1 Summary of contributions 152

7.2 Future research directions 155

Bibliography 157

List of Tables

1.1 Summary of the main Benders decomposition applications 30
1.2 Summary of the main Benders decomposition applications (continued) . . . 31

2.1 Effect of strengthening on cutset inequalities (group R) 46
2.2 Effect of strengthening on cutset inequalities (group C+) 47
2.3 Results for Benders inequalities (group R 10 vertices) 48
2.4 Results for Benders inequalities (group R 20 vertices) 49
2.5 Results for Benders inequalities (group C) 49
2.6 Results for Benders inequalities (group C - time limit exceeded) 50
2.7 Results for Benders inequalities (group C+) 50
2.8 Effect of using Benders inequalities associated with extreme rays 51

3.1 Results for the Benders algorithm with and without extra-cuts 67
3.2 Results for the Benders algorithm with and without extra-cuts (continued) . 68

4.1 Summary of the reduction tests. 84
4.2 Summary and comparison between the methods for the STPP and for the TSPP. 87
4.3 Summary and comparison between the methods for the STPP and for the TSPP (continued). 88

5.1 Summary of the algorithms 108
5.2 Optimal solutions for instance *Steinb8* for the 12 different scenarios 110
5.3 Summary of results 113
5.4 Results - Instances MStein, $h = 3$ 115
5.5 Results - Instances MStein, $h = 3$ (continued) 116
5.6 Results - Instances MStein, $h = 6$ 117

5.7 Results - Instances MStein, $h = 6$ (continued) . . . 118
5.8 Results - Instances MStein, $h = 9$ . . . 119
5.9 Results - Instances MStein, $h = 9$ (continued) . . . 120
5.10 Results - Instances MStein, $h = 12$ . . . 121
5.11 Results - Instances MStein, $h = 12$ (continued) . . . 122
5.12 Results - Instances steinc, $h = 5$ . . . 123
5.13 Results - Instances steinc, $h = 15$ . . . 124
5.14 Results - Instances steinc, $h = 25$ . . . 125

6.1 Summary of results for the mid-sized instances . . . 139
6.2 Summary of results for the larger instances . . . 140
6.3 Results for instances MSteinb, $h = 3$ . . . 142
6.4 Results for instances MSteinb, $h = 6$ . . . 143
6.5 Results for instances MSteinb, $h = 9$ . . . 144
6.6 Results for instances MSteinb, $h = 12$ . . . 145
6.7 Results for instances SteinC, $h = 5$ . . . 146
6.8 Results for instances SteinC, $h = 5$ (continued) . . . 147
6.9 Results for instances SteinC, $h = 15$ . . . 148
6.10 Results for instances SteinC, $h = 15$ (continued) . . . 149
6.11 Results for instances SteinC, $h = 25$ . . . 150
6.12 Results for instances SteinC, $h = 25$ (continued) . . . 151

List of Figures

1.1 Step-increasing *cost* × *capacity* function. 25

2.1 Network for Example 1 39
2.2 Network for Example 2 - four vertices, three arcs and two commodities. . . 42
2.3 Subnetworks for each commodity and ray associated with the cutset inequality for Example 2 43
2.4 Subnetworks for each commodity and ray associated with the cutset inequality for Example 3 44

4.1 Invalidity of test 1 for the budget and quota STPP. 82

5.1 Solutions for instance *Steinb8* in the different scenarios. 111
5.2 Solutions for instance *Steinb8* in the different scenarios (continued). 112

6.1 Example of different optimal subpaths between two vertices depending on the complete path. 131
6.2 Edges incident to leaf vertices. 134
6.3 Breaking points and possible paths to be removed 137

List of abbreviations

CNDP	Capacitated Network Design Problem.
DDFJ	Directed Dantzig-Fulkerson-Johnson.
DFJ	Dantzig-Fulkerson-Johnson.
FNDP	Fixed-Charge Network Design Problem.
GG	Garcia-Gouveia.
LAN	Local Access Networks.
MCFNDP	Multicommodity Capacitated Fixed-Charge Network Design Problem.
MSTP	Minimum Spanning Tree Problem.
MTZ	Miller-Tucker-Zemlin.
NWSTP	Node-Weighted Steiner Tree Problem.
OP	Orienteering Problem.
PCSTP	Prize-Collecting Steiner Tree Problem.
PCTSP	Prize-Collecting Travelling Salesman Problem.
PTP	Profitable Tour Problem.
STP	Steiner Tree Problem.
STPP	Steiner Tree Problem with Profits.
STPR	Steiner Tree Problem with Revenues.
STPRBH	Steiner Tree Problem with Revenues Budget and Hop Constraints.
TSP	Travelling Salesman Problem.
TSPP	Travelling Salesman Problem with Profits.
UDFJ	Undirected Dantzig-Fulkerson-Johnson.
UNDP	Uncapacitated Network Design Problem.

Acknowledgements

I will always feel fortunate and honored to have worked with such advisors as Professors Jean-François Cordeau and Gilbert Laporte. I sincerely thank them for their great availability and trust. I was also privileged to receive scientific advice from Professor Bernard Gendron and to benefit from the contribution of Professor Jean-Louis Goffin to my thesis committee. I am also grateful to Professor Luís Gouveia whose comments have helped improved this thesis.

I thank CAPES and the Brazilian people for the financial support which allowed me to come to Montreal. The financial help provided by the Ph.D. program of HEC Montréal and by the Canada Research Chair in Distribution Management is greatly appreciated. I am also grateful to the technical and administrative staff of the Center for Research on Transportation and of the Ph.D. program at HEC Montréal for providing a perfect work environment during these four years.

Finally, quoting Aristotle, "without friends no one would choose to live, though he had all other goods", I thank all my friends, especially Raquel and my family, for enjoying the good moments with me and for always being supportive when difficulties arose.

Chapter 0

Introduction

Network design problems constitute an important class of optimization problems, arising in diverse forms in a wide variety of fields such as electrical engineering, transportation planning, telecommunications, and computer science. The popularity of network design models is certainly related to the fact that these can appropriately represent a large number of important real-life situations, like the planning of electricity distribution systems (Ramirez-Rosado and Dominguez-Navarro, 2006), the planning of logistics distribution networks (Cordeau et al., 2006), the installation of telecommunications facilities (Gendron et al., 1999), and the planning of computer networks (Altiparmak et al., 2003).

Consider a graph $G = (V, E)$, where V is the vertex set and E is the edge set. Network design problems consist of selecting edges and vertices in such a way that a cost function is optimized and a set of constraints involving the edges and the vertices are respected. Different practical problems can be represented by adequately choosing the optimization criterion and the set of constraints.

Two important groups of network design problems are *Multicommodity Fixed-Charge Network Design Problems* and *Spanning Tree Problems*. In the first case, the edges are selected to allow the flow of commodities between designated origins and destinations. There is a fixed cost to be paid for each selected edge, and an edge-related variable unit cost for the commodities flowing through the edges. The goal is to obtain the minimum cost network that will allow the flow of all commodities. In many situations, one has a maximum flow associated to each edge, giving origin to the *Multicommodity Capacitated Fixed-Charge Network Design Problem* (MCFNDP). In the second case, the aim is to suitably select edges in E in order to span some vertices in V. Again, each edge is associated with a cost that

must be incurred if the edge is selected. Many subproblems may arise from spanning tree problems. For instance, if all vertices in V must be spanned, this results in the minimum spanning tree problem, while if not all vertices must be spanned (some optional vertices being spanned only if their inclusion reduces the solution cost) one obtains the Steiner tree problem. Finally, if besides the costs associated with the edges there are revenues associated with the vertices, one has the *Steiner Tree Problem with Profits* (STPP). In this two-part thesis, we are interested in both the MCFNDP and the STPP.

0.1 Part I: Multicommodity capacitated fixed-charge network design

Part I is devoted to the MCFNDP. In Chapter 1, we present a classification of different variants of this problem. Several methods such as tabu search and Lagrangean relaxation have been proposed to solve such problems. We are particulary interested in the Benders decomposition approach (Benders, 1962). This method relies on a reformulation of the model yielding an optimal solution via an iterative cut generation scheme based on the solution of linear programs. In Chapter 1 we survey the application of Benders decomposition to fixed-charge network design problems, where we include uncapacitated and single-commodity models for the sake of completeness.

When solving these problems, it is common to attempt to reinforce the models through the inclusion of valid inequalities. Three important families of valid inequalities are the Benders (also known as Benders feasibility cuts), the cutset and the metric inequalities. In Chapter 2, we study the interrelations between these three groups of inequalities. We show that Benders inequalities associated with extreme rays are metric inequalities. We also show how to strengthen Benders inequalities associated with non-extreme rays to obtain metric inequalities. We show that cutset inequalities are Benders inequalities, but not necessarily metric inequalities. We give a necessary and sufficient condition for a cutset inequality to be a metric inequality. Computational experiments on classical MCFNDP instances show the effectiveness of strengthening Benders and cutset inequalities to obtain metric inequalities.

In Chapter 3, we extend the proposed Benders algorithm framework developed to test the strengthening procedure in order to develop a complete solution method for the MCFNDP. Several extensions over the standard Benders algorithm are implemented, including the addition of Pareto-optimal cuts, the delayed generation of additional bounds

on certain variables, and the strengthening of the Benders feasibility cuts. In particular, we develop a procedure for increasing the number of generated cuts during the resolution of the relaxed problem. The idea of the procedure is to try and reduce the number of iterations needed to solve the integer problem, since these are much more computationally expensive. This procedure complements the ideas of McDaniel and Devine (1977) who were the first to propose the resolution of the relaxed problem in order to generate initial cuts for the integer problem.

0.2 Part II: Steiner tree problems with profits

Part II is devoted to the STPP, a variant of the Steiner tree problem in which there are not only costs associated with edges, but also revenues associated with the vertices. The idea of the STPP is that one is not obliged to span a vertex but, if a vertex is spanned a revenue is earned. This model represents a situation faced, for instance, by telecommunication facilities (Lucena and Resende, 2004), where the operator of the facility must decide which consumers it will serve by considering both the cost of constructing the network as well as the revenues derived from the spanned vertices. In Chapter 4, we present a survey of the STPP. We focus on the different variants of the STPP, presenting an overview of models and algorithms. We note that most recent algorithms make use of preprocessing reduction tests which try to eliminate vertices and edges that can be deleted without the risk of eliminating all optimal solutions. We present a summary of the most important of these tests. Moreover, we compare the state-of-art algorithms for the STPP with those for the *Travelling Salesman Problem with Profits* (TSPP) (a variant of the classical *Travelling Salesman Problem*, in which each vertex has an associated profit). From this comparison, we note that the Budget STPP, an important variant of the STPP in which there is a limit on the total network cost, has been almost ignored by previous researchers. We show that most of the reduction tests existing in the literature cannot be applied to this problem. We also develop two new reduction tests for this variant.

In Chapter 5, we introduce the notion of hop constraints, which limit the number of edges (hops) between a root vertex and any other vertex in the tree. These constraints can be useful to increase the reliability of the network or to limit its transmission delays. We propose and compare several models and algorithms for the *Steiner Tree Problem with Revenues, Budget and Hop Constraints* (STPRBH). We choose the term *revenues* instead of the more standard term *profits* in order to emphasize the fact that in the problems under

study the goal is to maximize the collected revenue, and not the difference between revenues and costs, since the costs are dealt with by the constraints of the model. In Chapter 5, we prove that the three-index position variables model theoretically dominates another model based on potential variables and Miller-Tucker-Zemlin constraints. Moreover, the computational results show that the best algorithm for the STPRBH clearly depends on the hop limit. They also show that the STPRBH is a difficult problem and that even mid-sized instances can be hard to solve.

In Chapter 6, heuristic procedures are developed for the STPRBH. First, a fast greedy algorithm based on a hop-constrained shortest path procedure is described. This method yields good approximations within very short computation times. This initial solution is improved by means of a *destroy-and-repair* algorithm or by tabu search. The destroy-and-repair algorithm works by carefully changing the costs of some arcs in the current solution and repeatedly rerunning the greedy algorithm. The tabu search heuristic yields very good approximations by sequentially visiting neighbors of the current solution and by using classical mechanisms such as tabu tenures and diversification in order to avoid cycling. Comparative computational tests are performed.

Chapter 7 summarizes the main contributions of the thesis and points to future research directions.

Part I

Multicommodity fixed-charge network design

Chapter 1

A survey on Benders decomposition applied to fixed-charge network design

Abstract

Network Design problems concern the selection of arcs in a graph in order to satisfy, at minimum cost, some flow requirements, usually expressed in the form of origin-destination pair demands. Benders decomposition methods, based on the idea of partition and delayed constraint generation, have been successfully applied to many of these problems. This chapter presents a review of these applications.

Keywords: Benders decomposition; network design; fixed-charge.

1.1 Introduction

Network design problems are central to a large number of contexts including transportation, telecommunications and power systems. The idea is to establish a network of links (roads, optical fibers, electric lines, etc.) that enables the flow of commodities (people, data packets, electricity, etc.) in order to satisfy some demand characteristics. We are particulary interested in *Fixed-Charge Network Design Problems*, where, in order to use a link, one must pay a fixed cost representing, for example, the cost of constructing a road, or installing an electric line, etc.

A large number of practical applications may be represented by fixed-charge network

design models. One important area is the *Service Network Design Problem* which arise, for example, in airline and trucking companies. The idea is to maximize the profit by setting routes and schedules, given some resource constraints. For example, airline companies must determine the covered routes and the frequency of the flights considering aircraft and crew availability (Lederer and Nambimadom, 1998). Similarly, express package delivery companies (e.g. UPS, Fedex) must establish routes, assign aircraft to them and decide about the flow of packages (Armacost et al., 2002).

Various applications can also be found in telecommunications. Examples are the design of a local access network with one or two technologies (Randazzo and Luna, 2001; Randazzo et al., 2001), the project of a terminal layout (Gavish, 1983; Gouveia, 1995a), and the interconnection of existing networks (Myung et al., 1995).

In power systems, fixed-charge network design is used to plan the energy transmission from the generation plants to the consumer centers (Romero and Monticelli, 1994a; Binato et al., 2001) and the energy distribution inside these centers (Gascon et al., 1993; Carneiro et al., 1996). In the latter case, network models can also be used in an operational context to obtain the configuration that minimizes daily loss costs (Cavelluci and Lyra Filho, 1997).

In all these problems, a proper design can yield better operation levels and cost reductions. The total amount of these reductions is obviously related to each specific problem. However, the economical importance of most of the cited problems and the key role played by the network design in the systems operation suggest that the savings can be significant. For example, Standard and Poor's (2000) estimates the revenues of package delivery industry as 52 billions US$ only in the United States.

Probably because of the economical importance of the associated problems, several solution methodologies for network design problems are available. These range from pure heuristic methods to optimal implicit enumeration. Amongst the most successful solution approaches we find Benders Decomposition (Benders, 1962). The basic idea behind this method is to decompose the problem into two simpler parts: the first part, called *master problem*, solves a relaxed version of the problem and obtains values for a subset of the variables. The second part, called *auxiliary problem* (or *subproblem*), obtains the values for the remaining variables while keeping the first ones fixed, and uses these to generate cuts for the master problem. The master and auxiliary problems are solved iteratively until no more cuts can be generated. The conjunction of the variables found in the last master and subproblem iteration is the solution to the original formulation.

The structure of fixed-charge network design problems presents a natural decomposition

scheme for the Benders approach: the variables representing the opening of the links are solved in the master problem while the ones representing the actual flow of commodities are kept in the subproblem. Therefore, at each iteration the master solution gives a tentative network for which the subproblem finds the optimal flow of commodities.

Due to the need to solve the master and the auxiliary problems several times, the decomposition approach is only reasonable if these problems can be solved efficiently. This is the case for network design problems, where most of the times, it is much easier to solve the decomposed problems than the original one. Moreover, especially for the subproblem, it is sometimes possible to proceed with further decomposition (by commodity, by period, etc.), resulting in even more efficient solution methods. All this makes of Benders decomposition a competitive methodology for this family of problems.

In this chapter, we review the use of Benders decomposition in fixed-charge network design. In the next section, we present the Benders decomposition approach in further detail. Then, in Section 1.3 we present the different fixed-charge network problems and a survey of the application of Benders decomposition algorithms to these models. Section 1.4 closes the chapter with a summary and conclusions.

1.2 Benders decomposition

Benders Decomposition (Benders, 1962) is a classical solution approach for combinatorial optimization problems, based on the ideas of partition and delayed constraint generation. Examples of successful applications of this methodology to mixed-integer problems are abundant. We cite, for example, the seminal paper of Geoffrion and Graves (1974) on multicommodity distribution network design and the extensions presented by Cordeau, Pasin and Solomon (2006) on the same problem. Other applications include the locomotive and car assignment problem (Florian et al., 1976; Cordeau et al., 2000,2001) the large scale water ressource management problem (Cai et al., 2001) and the two stage stochastic linear problem (Zhao, 2001).

The method partitions the model to be solved into two simpler problems, named master and auxiliary problems. The master problem is a relaxed version of the original problem, containing only a subset of the original variables and the associated constraints. The auxiliary problem is the original problem with the variables obtained in the master problem fixed.

In the case of the fixed-charge network design problem, the master deals with the

integer variables that define the network while the subproblem works with the continuous variables representing the actual flow of commodities for the tentative network obtained in the master problem. Consider, for example, the general formulation presented below, which encompasses most fixed-charge network design models proposed in the literature:

General Fixed-Charge Network Design Formulation

$$\text{Minimize } cx + dy \tag{1.1}$$

$$\text{subject to} \quad Ax + By \geq b \tag{1.2}$$

$$Dy \geq e \tag{1.3}$$

$$x \geq 0, y \geq 0 \text{ and integer.} \tag{1.4}$$

Vectors x and y are the continuous and integer variables, respectively, while c and d are the row vectors of the associated costs. Matrices A, B and D and vectors b and e have the appropriate dimensions. This problem can be expressed as:

$$\min_{\overline{y} \in Y} \{ d\overline{y} + \min_{x \geq 0} \{cx : Ax \geq b - B\overline{y}\} \}, \tag{1.5}$$

where $Y = \{y \,|\, Dy \geq e,\ y \geq 0 \text{ and integer}\}$. The inner minimization is a linear program. Associating dual variables u to constraints $Ax \geq b - B\overline{y}$, we can write the dual version of this problem as

$$\max_{u \geq 0} \{u(b - B\overline{y}) : uA \leq c\}. \tag{1.6}$$

This is the Benders decomposition subproblem. Using duality theory, the primal and dual formulations can be interchanged. Therefore, (1.5) can be rewritten as:

$$\min_{\overline{y} \in Y} \{ d\overline{y} + \max_{u \geq 0} \{u(b - B\overline{y}) : uA \leq c\} \}. \tag{1.7}$$

Note that the feasible space of the subproblem (inner maximization) is independent of the choice made for variables y. Let $F = \{u \,|\, u \geq 0\ ;\ uA \leq c\}$ represent this feasible space. We assume that F is not empty for it would correspond to a primal problem either infeasible or unbounded. F is therefore composed of extreme points u^p (for $p = 1, \ldots, P$) and extreme rays r^q (for $q = 1, \ldots, Q$).

The solution of the subproblem can be either bounded or unbounded. In the first case, the solution is one of the extreme points u^p ($p = 1, \ldots, P$). In the latter situation, there is

a direction r^q for which $r^q(b - B\overline{y}) > 0$. The unbounded situation results in an unfeasible primal problem and must be avoided. We must therefore eliminate the values of $\overline{y}$ that would yield an unbounded inner dual problem. This is done by explicitly considering the restrictions:

$$r^q(b - B\overline{y}) \leq 0, \qquad q = 1, \ldots, Q. \tag{1.8}$$

With this restrictions in the external formulation, the maximum value of the inner problem is the value of one of the extreme points of F. Problem (1.7) becomes:

$$\min_{\overline{y} \in Y} \{ d\overline{y} + \max\{u^p(b - B\overline{y}) : p = 1, \ldots, P\}\} \tag{1.9}$$

$$\text{subject to} \quad r^q(b - B\overline{y}) \leq 0, \qquad q = 1, \ldots, Q,$$

or, with the use of an auxiliary continuous variable z:

$$\text{Minimize} \quad dy + z \tag{1.10}$$

$$\text{subject to} \quad z \geq u^p(b - B\overline{y}), \qquad p = 1, \ldots, P \tag{1.11}$$

$$r^q(b - B\overline{y}) \leq 0, \qquad q = 1, \ldots, Q \tag{1.12}$$

$$y \in Y,\ z \geq 0. \tag{1.13}$$

Formulation (1.10)–(1.13) is called the Benders reformulation and its drawback is that the number of extreme points and extreme rays is usually extremely large. To overcome this limitation, Benders proposed to delay the generation of constraints (1.11) and (1.12). Initially, only constraints (1.13) are considered, yielding the first master problem:

$$\text{Minimize} \quad dy + z \tag{1.14}$$

$$\text{subject to} \quad y \in Y,\ z \geq 0. \tag{1.15}$$

This problem is a relaxed version of (1.10)–(1.13) and therefore the objective value $dy + z$ is a lower bound to the original problem. Once this problem is solved, a tentative configuration of variables y is used in the subproblem (1.6). This subproblem is solved and the result is either unbounded, in which case a constraint of type (1.12) is inserted in the master, or it is an extreme point. In the latter situation, the conjunction of the solution of the master with the primal solution of the subproblem provides a complete solution

(upper bound) to the original problem, while the dual solution of the subproblem is used to generate a constraint of type (1.11), which is also inserted in the master.

The master and the auxiliary problems are solved iteratively, until the upper and lower bounds are sufficiently close. Because of this iterative constraint generation scheme, Benders Decomposition can be regarded as sort of a "dual version" of Dantzig-Wolfe column-generation: the application of Dantzig-Wolfe column generation to the dual of the original problem.

Several extensions have been proposed for the Benders decomposition method. One of the most important ones is presented by Geoffrion (1972) who suggests a "Generalized Benders Decomposition" approach. Geoffrion makes use of non-linear duality theory and extends the Benders method to the case where the subproblem is a convex optimization problem. This development enables the application of Benders decomposition to a whole new set of problems, particulary those where the joint problem is generally non-convex but can be made convex by fixing one set of variables. Values for these variables are therefore obtained in the master problem and fixed in the subproblem which then becomes a convex optimization problem for which many optimization methods are available.

Magnanti and Wong (1981) have studied the influence of cuts in a Benders Decomposition algorithm. They showed that the use of stronger cuts, when available, may have a great impact in the convergence of the algorithm by reducing the number of iterations. The idea is simple. Many times the solution of the subproblem (1.6) is not unique. Indeed, for network design problems this is usually true, since the subproblems are variations of *Shortest Route*, *Transhipment* and other network optimization problems, known for their degeneracy. Therefore, one can try to judiciously chose the dual variables in order to obtain better cuts. To determine what is a better cut the authors use the notion of dominance: for the presented reformulation of problem (1.1)-(1.4), a cut generated from the extreme point u^1 dominates a cut generated from the extreme point u^2 if

$$u^1(b - B\overline{y}) \geq u^2(b - B\overline{y})$$

for all $\overline{y} \in Y$ with strict inequality for at least one point. A cut that is dominated by no other cut is Pareto-optimal. Let Y^{LP} be the polyhedron defined by the linear relaxation of Y, and $ri(Y^{LP})$ be the relative interior of Y^{LP}. The following problem yields a Pareto-optimal cut for the case of the general problem (1.1)-(1.4):

$$\text{Maximize } u(b - By^o) \tag{1.16}$$

$$\text{subject to} \qquad uA \leq c \tag{1.17}$$

$$v(\overline{y}) = u(b - B\overline{y}) \tag{1.18}$$

$$u \geq 0. \tag{1.19}$$

where $y^o \in ri(Y^L P)$ and $v(\overline{y})$ is the optimal value of the subproblem (1.6) when variables y are fixed in $\overline{y}$. The objective function maximizes the strength of the cut for y^o while the constraints define the feasible space as the optimal solutions of the original subproblem.

The authors also analyze the impact of different formulations on the performance of a Benders Decomposition algorithm. The conclusion is that two different formulations for a problem can yield very different performances. In fact, tighter formulations yield stronger Benders cuts, to the extent that for the Convex Hull formulation, only one cut is needed. However, adding constraints to a formulation not only strengthens the Benders cuts but also complicates the solution of the linear subproblems, yielding a trade-off between the quality of the Benders cuts and the computational effort needed to solve the subproblems.

In a later work, Magnanti, Mireault and Wong (1986) have extended the study to the context of the *Uncapacitated Network Design Problem*. They showed the generality of Benders Decomposition by proving that some known cuts (Boyce et al., 1973; Hoang, 1973; Gallo, 1983; Magnanti and Wong, 1984) are actually Benders cuts for particular choices of the integer variables (the tentative network configuration). Moreover, they showed that in this particular case, Pareto-optimal cuts can be generated at the price of solving k *Minimum Cost Flow Problems* (one for each commodity). The authors also advocate the use of Benders Decomposition in conjunction with other approaches, based on their own experience with the use of a dual-ascent method, as will be seen in the next section.

1.3 Literature review

Network Design problems concern the selection of arcs in a graph in order to satisfy, at minimum cost, some flow requirements usually expressed in the form of origin-destination pair demands (for surveys of these problems see Magnanti and Wong, 1984 ; Minoux, 1989 and Balakrishnan et al., 1997). The *Fixed-Charge Network Design Problem* (FNDP) has the particularity that each arc has an associated fixed cost which must be paid if the arc is part of the solution.

In this section we present a review of the Benders decomposition algorithms proposed for the FNDP. We organize our presentation according to the different problems. First, we

characterize the models as uncapacitated or capacitated, and we then enumerate the most common problems within each category.

The idea is to present different versions of fixed-charge network design problems, giving for each one, a mathematical formulation and general comments on the applications of Benders decomposition methods for that problem found in the literature.

Concerning the mathematical developments, we note that there are two main types of formulations that are used in the literature: node-arc and arc-path models. For ease of presentation, we concentrate on node-arc models and we develop the arc-path model only for the last and more general case of Section 1.3.2.

In all developments we denote by $G(N, A, K)$ an undirected graph where N is the set of vertices, A is the set of edges, and K is the set of commodities to be transported (people or goods, data packets, electric power, etc.). Variables x_{ij}^k represent the actual flow of commodity k going from vertex i to vertex j, and variables y_{ij} are integer variables associated with the utilization of the link (uncapacitated case) or to its capacity (capacitated case).

Note that, although the graph in undirected, it makes sense to define both variables x_{ij}^k and x_{ji}^k, because the flow itself is directed. Thus, we consider that an edge in A is doubly represented by the arcs (i, j) and (j, i). However, whenever we refer to the integer variables (or their coefficients) we assume that only the edge (i, j) with $i < j$ is considered.

1.3.1 Uncapacitated network design problems (UNDP)

We first consider *Uncapacitated Network Design Problems*. In these problems, there is no limit on the flow that can circulate through the selected links. Many real-life applications may be well represented by such models (Gavish, 1983; Balakrishnan et al., 1994; Randazzo et al., 2001).

The origin-destination pairs UNDP

We start with the *Origin-Destination Pairs Uncapacitated Network Design Problem*. In this case, to each commodity $k = 1, \ldots, |K|$ is associated a demand d_k, an origin vertex $O(k)$ and a destination vertex $D(k)$. In some cases, the fixed costs fully represent the real cost of the network, i.e., there is no volume cost associated with the flow of a commodity on one link (see, e.g., Sridhar and Park, 2000, and references therein). However, in order to appropriately model some problems, one must sometimes include volume-based costs, as in the work of Randazzo, Luna and Mahey (2001).

Using the notation introduced at the beginning of the section, we present below one

formulation for this class of problems.

(F_1) ***- origin-destination pairs UNDP formulation***

$$\text{Minimize} \sum_{(i,j)\in A} \left(\sum_{k\in K} c_{ij}^k x_{ij}^k + f_{ij} y_{ij} \right) \tag{1.20}$$

subject to

$$\sum_{j|(i,j)\in A} x_{ij}^k - \sum_{j|(j,i)\in A} x_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \notin \{O(k), D(k)\}, \\ -d_k, & i = D(k), \end{cases} \quad k \in K, \tag{1.21}$$

$$x_{ij}^k \leq d_k y_{ij}, \qquad (i,j) \in A, \quad k \in K, \tag{1.22}$$

$$x_{ij}^k \geq 0, \qquad (i,j) \in A, \quad k \in K, \tag{1.23}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{1.24}$$

Variables y_{ij} are associated with the construction of link $(i,j) \in A$: $y_{ij} = 1$ if (i,j) belongs to the final solution, otherwise $y_{ij} = 0$. The objective function (1.20) is the sum of variable and fixed costs. In this function, c_{ij}^k is the linear cost associated with the flow of one unit of commodity k through link (i,j) and f_{ij} is the fixed cost associated with the selection of link (i,j) in the final solution. Some non-linear versions of this function can be considered by "Generalized Benders Decomposition" (see, for example, the work of Hoang (1982) later on this section or the design of electrical networks in Section 1.3.2). However, for simplicity, we concentrate on linear objective functions.

The set of restrictions (1.21) ensures that the flow of commodity k leaves the origin vertex $O(k)$ and arrives at the destination vertex $D(k)$, and are also responsible for flow conservation. Constraints (1.22) limit the flow of commodities to selected arcs, constraints (1.23) define the non-negativity of variables x_{ij}^k, and constraints (1.24) impose the integrality of variables y_{ij}. Constraints (1.22) can be substituted by aggregate constraints, in the form:

$$\sum_{k\in K} x_{ij}^k \leq (\sum_{k\in K} d_k) y_{ij}, \qquad (i,j) \in A. \tag{1.25}$$

This aggregation can substantially reduce the number of constraints. However, as discussed in the last section, it has been shown that the Benders cuts obtained with disag-

gregated (tighter) formulations are more efficient than the ones obtained with aggregated ones (Magnanti et al., 1986).

Application of Benders decomposition. The basic uncapacitated case was solved via Benders decomposition in an early work of Magnanti, Mireault and Wong (1986). There, the master problem proposes a tentative network by setting the integer variables and the subproblem finds the continuous variables flow distribution. The authors use a set of instances of up to 30 vertices and 130 arcs to prove two points: i) that the use of auxiliary techniques can improve the performance of the Benders decomposition (in this case, a preprocessing based on a dual-ascent procedure in order to eliminate variables), and ii) that the intelligent choice of cuts, specifically the Pareto-optimal cuts (see Section 1.2), can strongly affect the performance of the algorithm.

The work of Magnanti *et al.* was extended ten years later by Gutierrez, Kouvelis and Kurawarwala (1996), who proposed a robust approach able to consider the uncertainty in the transportation costs c_{ij}^k and in the demands d_k. The data uncertainty is described though a set of scenarios, each one with different values for c_{ij}^k. Note that, since the authors deal with the uncapacitated case, the changes in c_{ij}^k may reflect both changes in the cost itself or in the demand (with the appropriate scaling). The solution of the model is done via a multi-master Benders algorithm, where an individual master problem is associated with each possible scenario. Each time a master problem is solved, there is a cross generation of Benders cuts, i.e., the subproblem generates a cut for each of the master problems. The authors use a set of instances based on the one presented by Magnanti *et al.* (1986) with 11 scenarios for each original problem and conclude that the cross-generation of Benders cuts accelerates the convergence of the algorithm.

An interesting version of the uncapacitated network design problem with one facility has been proposed by Hoang (1982). The author develops a formulation of this problem with non-linear costs. The objective is to select among several projects (each corresponding to a set of investments in the network) respecting some budget constraints. In this case, no explicit upper bounds on the arc capacities like (1.22) exist: they are indirectly accounted for by the convexity in the cost functions. Generalized Benders Decomposition is used, together with a Lagrangean heuristic to solve the master, and a convex cost multicommodity algorithm (Nguyen, 1974) is applied to the solution of the subproblem. Two instances are successfully solved, the largest one having 155 vertices, 376 arcs and 720 commodities.

The single-origin UNDP (local access network design)

An important special case of the problem presented in Section 1.3.1 is where all demand points must be connected to a single origin. This is the situation, for example, in the design of *Local Access Networks* (LAN) (Gouveia, 1997; Randazzo and Luna, 2001). When the objective function is well represented only by the edge-related variable costs, the LAN Design can be formulated as the easy *Single-Source Transshipment Problem*. On the other hand, if there are only fixed-charge costs, it becomes an NP-Hard *Steiner Problem*, or a *Minimum Spanning Tree Problem* (if all the vertices must be reached). In the latter case, an important situation is where the number of links connecting directly or indirectly to the source is limited. This problem is known as the *Capacitated Minimum Spanning Tree Problem* (Gavish, 1983; Gouveia, 1995a) (we classify this problem as uncapacitated since the flow on the arc itself is not subject to any upper bound).

We present a mathematical formulation for the general case where each link has a fixed and variable associated cost. In (F_2) we consider each demand as a different commodity, with different transportation costs c_{ij}^k that are dependent both on the link and on the commodity. Vertex O is the origin for all the commodities.

(F_2) - Single-Origin UNDP Formulation

$$\text{minimize} \sum_{(i,j)\in A} \left(\sum_{k\in K} c_{ij}^k x_{ij}^k + f_{ij} y_{ij} \right) \tag{1.26}$$

subject to

$$\sum_{j|(i,j)\in A} x_{ij}^k - \sum_{j|(j,i)\in A} x_{ji}^k = \begin{cases} d_k, & i = O, \\ 0, & i \notin \{O, D(k)\}, \quad k \in K, \\ -d_k, & i = D(k), \end{cases} \tag{1.27}$$

$$x_{ij}^k \leq d_k y_{ij}, \qquad (i,j) \in A, \quad k \in K, \tag{1.28}$$

$$x_{ij}^k \geq 0, \qquad (i,j) \in A, \quad k \in K, \tag{1.29}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{1.30}$$

Besides the fact that all commodities are now originating at one single origin vertex, (F_2) is identical to (F_1). However, the presence of a single origin vertex yields a simplified formulation for the case where the variable costs are independent on the commodity. (F_2') presents this simplified single-origin, single-commodity formulation.

(F_2') - Single-Origin, Single-Commodity UNDP Formulation

$$\text{Minimize} \sum_{(i,j)\in A} (c_{ij}x_{ij} + f_{ij}y_{ij}) \tag{1.31}$$

subject to

$$\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} = \begin{cases} D, & i = O, \\ -d_i, & i \neq O, \end{cases} \tag{1.32}$$

$$x_{ij} \leq Dy_{ij}, \qquad (i,j) \in A, \tag{1.33}$$

$$x_{ij} \geq 0, \qquad (i,j) \in A, \tag{1.34}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{1.35}$$

The parameter d_i represents the demand at each vertex and D is the sum of the demands in all vertices except for the origin. The objective function (1.31) of formulation (F_2') still minimizes the sum of fixed and variable costs. However, there is no need for a differentiation among commodities. The only requirement is that the correct amount of flow leaves the vertex origin and reaches each one of the demand vertices, which is ensured by constraints (1.32). Constraints (1.33) still forbid the flow in closed links.

As pointed out by Magnanti and Wong (1981), selecting the proper formulation is an important factor that affects the computational performance of Benders decomposition. In this case, we note that since constraints (1.33) of formulation (F_2') make use of the sum of all the demands D, they are an equivalent of the aggregated constraints presented in (1.25). Therefore, we should expect this formulation to be weaker than formulation (F_2), which uses the disaggregated constraints as expressed in (1.28).

Application of Benders decomposition. Randazzo and Luna (2001) present a comparison of a Lagrangean relaxation-based branch-and-bound, a branch-and-cut and a Benders decomposition methodologies for the LAN design problem presented in this section. Prior to the Benders decomposition, the authors use a linear relaxation and a shortest-path algorithm to obtain a feasible solution, which is used to generate initial values for the dual variables. Redundant strengthening constraints are used in the master, to increase the probability of generating a feasible solution. The subproblem is decomposed into k easy network flow problems, one for each commodity. Benders decomposition outperformed the two other methods on three small sets of instances. Perhaps the most important conclusion

obtained by the authors is the fact that Benders decomposition, although slower than the branch-and-cut on six of the 30 instances, was the only algorithm able to solve to optimality all instances within the limit time (24h), leading to the idea that it is a more robust method.

In an earlier article, Gavish (1982) has presented a Benders decomposition algorithm for the *Capacitated Minimal Spanning Tree Problem*. The algorithm is rather standard: the master problem fixes the integer variables yielding two situations i) the selected edges form a spanning tree: then, either the solution is optimal (if the number of links connecting to the source respects the limit) or the links connected to the source can be used to generate a cut and solve the master problem again, ii) the graph is not connected, in which case a cut can be generated and the master problem is solved again. The algorithm was tested on networks containing up to 12 vertices. The number of generated cuts was very large, resulting in a poor performance of the algorithm. It is important to note, however, that the implementation by Gavish was very standard. Several improvements like those presented in Chapter 3 could probably change the original conclusions.

The two-technologies single-origin UNDP (two technologies local access network design problem)

A third common situation in uncapacitated network design problems is the case where two technologies, each having their own advantages, are available (Balakrishnan et al., 1994; Randazzo et al., 2001; Gouveia and Telhada, 2001). One technology may have a larger fixed cost but a smaller variable cost and vice-versa (as happens with optical fibers and copper in telecommunications networks).

The formulation is an extension of formulation $(F2)$, with the duplication of variables x_{ij}^k and y_{ij}. Therefore, in $(F3)$, y_{ijt} is the binary variable associated with the construction of the link (i,j) with technology t. Analogously, x_{ijt}^k is the flow of commodity k through that link.

(F_3) - Two-Technologies Single-Origin UNDP Formulation

$$\text{Minimize} \sum_{(i,j)\in A} \sum_{t=1}^{2} \left(\sum_{k\in K} c_{ijt}^k x_{ijt}^k + f_{ijt} y_{ijt} \right) \tag{1.36}$$

subject to

$$\sum_{t=1}^{2}\left(\sum_{j|(i,j)\in A} x_{ijt}^{k} - \sum_{j|(j,i)\in A} x_{jit}^{k}\right) = \begin{cases} d_k, & i = O, \\ 0, & i \notin \{O, D(k)\}, \\ -d_k, & i = D(k), \end{cases} \quad k \in K, \tag{1.37}$$

$$x_{ijt}^{k} \leq d_k y_{ijt}, \qquad (i,j) \in A, \quad k \in K, \quad t = 1,2, \tag{1.38}$$

$$\sum_{t=1}^{2} y_{ijt} \leq 1, \qquad (i,j) \in A, \tag{1.39}$$

$$x_{ijt}^{k} \geq 0, \qquad (i,j) \in A, \quad k \in K, \quad t = 1,2, \tag{1.40}$$

$$y_{ijt} \in \{0,1\}, \qquad (i,j) \in A, \quad t = 1,2. \tag{1.41}$$

In (F_3), the goal (1.36) is to minimize the sum of variable and fixed costs for both technologies. This must be done while respecting the demand requirements (1.37) and the fact that, for both technologies, there can only be flow in a link if the link is open (1.38). Each link uses a single technology, as expressed by constraints (1.39).

We observe that (F_3) is the most complete of the four formulations. In fact, (F_1), (F_2) and (F_2') can all be regarded as special cases of (F_3). Moreover, (F_3) easily adapts to the case where more technologies are available (although the case with two technologies is the most common in practice).

Straightforward extensions of (F_3) include models that require a cost for connecting links with different technologies. These models are useful for the common situation where extra equipment is needed to perform the connection. Other models may be used to limit the number of such connections or restrict the vertices where it may take place.

Additional restrictions may force some 'primary vertices' to be served by the 'primary' technology (Gouveia, 1997). Carneiro, França and Silveira (1996) present an interesting application of this model for the problem of designing power distribution secondary networks. In this case, all the demand must be supplied by the 'secondary' technology. However, it is still advantageous to use the primary technology (in this case, the higher voltages) until very close to the demand points for it has much lower variable costs. The conversion from the first technology to the second requires the installation of additional equipment in the vertices of the network (electric transformers) and is limited to the capacity of these transformers.

Application of Benders decomposition. Randazzo et al. (2001) use Benders decomposition to solve this problem. The algorithm is very similar to the one presented by

Randazzo and Luna (2001), but without the use of linear relaxation. The first feasible solution is obtained by means of a shortest-path algorithm considering only the variable costs. This initial solution is used to generate initial lower and upper bounds. Additional constraints are used in the master in order to guarantee an arborescent structure during its solution. The auxiliary subproblems are still easy network flow problems decomposable by commodity. The tests were conducted on a limited set of instances of up to 41 vertices (all instances could be solved to optimality with a linear relaxation model) for which Benders decomposition was usually faster (especially for the bigger instances) than CPLEX 3.0.

1.3.2 Capacitated network design problems (CNDP)

We now consider the *Capacitated Network Design Problems*. As before, flow in a link is only allowed if a fixed cost is paid. However, the amount of this flow is limited to a certain capacity. Examples of applications of CNDP frequently occur in the design of telecommunications networks (Gendron et al., 1999; Riis and Andersen, 2002). In this section, we analyze some of the most common models. These models are presented in the next subsections and classified according to the structure of the cost-capacity relationship.

The single-facility CNDP

The first CNDP to be considered is the *Single-Facility CNDP*. In this case, there is only one type of technology, which can be installed multiple times on each link. Each unit of the facility installed enables a given amount of flow and has a given cost. Examples of this problem can be found in Magnanti and Mirchandani (1993) and Barahona (1996). In (F_4) we present a mathematical formulation for the case with origin-destination pair demands.

In (F_4), there are two important modifications with respect to (F_1). First, one single constraint of type (1.44) is generated for each arc (instead of k constraints per arc). These constraints now link the x_{ij}^k variables together, as in constraints (1.25). Moreover, in the right-hand side of (1.44), d_k has been changed for C, the amount of capacity given by each unit of the *purchased* facility. The second modification is that constraints (1.45) indicate that y_{ij} is integer. These two modifications enable the selection of an integer number of unities of the facility to be installed in each individual link, therefore, giving the link the total capacity of $y_{ij} \times C$.

(F_4) - Capacitated Single-Facility Network Design Problem

$$\text{Minimize} \sum_{(i,j)\in A} \left(\sum_{k\in K} c_{ij}^k x_{ij}^k + f_{ij} y_{ij} \right) \tag{1.42}$$

subject to

$$\sum_{j|(i,j)\in A} x_{ij}^k - \sum_{j|(j,i)\in A} x_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \notin \{O(k), D(k)\}, \\ -d_k, & i = D(k), \end{cases} \quad k \in K, \tag{1.43}$$

$$\sum_{k\in K} x_{ij}^k \leq C y_{ij}, \qquad (i,j) \in A, \tag{1.44}$$

$$x_{ij}^k \geq 0, \qquad (i,j) \in A, \quad k \in K, \tag{1.45}$$

$$y_{ij} \in \mathbb{Z}, \qquad (i,j) \in A. \tag{1.46}$$

Modifications in order to make (F_4) a *single-origin* model analogous to the one presented in Section 1.3.1 are straightforward and equivalent to those required in the uncapacitated case.

Application of Benders decomposition. Sridhar and Park (2000) work with a version of this problem that does not have variable costs. Besides the arc capacity constraints, their model also includes vertex capacity constraints, in the form:

$$\sum_{j|(i,j)\in A} \sum_{k\in K} x_{ij}^k + \sum_{k|D(k)=i} d_k \leq \kappa_i \qquad i, \tag{1.47}$$

indicating that the vertex capacity κ_i must be greater or equal to all the flow passing through, originating or ending at vertex i. The capacity κ_i of each vertex is a parameter determined by the desired network performance. The model is solved using an embedded Benders enumeration algorithm, initiated with a lower bound given by a cutting plane algorithm based on two polyhedral cuts and a feasible solution given by a heuristic algorithm (Sridhar et al., 2000). The method is applied to instances of up to 20 vertices and several demand levels, for which the Benders-and-cut approach outperformed a conventional branch-and-bound.

Ouorou, Luna and Mahey (2001) deal with a similar single-facility capacitated problem. In this case, however, the demands are *elastic*, i.e., one of the goals of the optimization problem is to establish the optimal demand levels (this can be done considering that the

demands are dependent on the asked prices, as is often the case in telecommunications systems). Another important difference is the fact that the authors deal with a *capacity expansion* problem, i.e., there is already sufficient capacity installed and the question is wether it may be advantageous to expand the capacity in order to serve consumers who are willing to pay higher tariffs in exchange for better service levels. A generalized Benders decomposition scheme is used, yielding a mixed-integer master problem that deals with the capacity expansions and a convex network flow auxiliary subproblem that solves the demand and flow levels. The algorithm chosen for solving the subproblems is an adaptation of the proximal decomposition method (Mahey et al., 1998) to the case of elastic demands which alternates between shortest path calculations and primal-dual updates efficiently distributed among arcs and commodities. The authors solve instances based on real-life data with up to 19 vertices, 68 arcs and 342 commodities and the results suggest that Benders decomposition is an effective strategy.

The two-facility CNDP

The *Two-Facility CNDP* is the analogous capacitated model for the uncapacitated situation presented in Section 1.3.1. The extension here is that besides choosing the technology for each link, we must also choose the capacity to be installed. This capacity must come in multiples of the unitary facility capacity for the chosen technology. Therefore, given two technologies with two different unitary fixed costs f_{ij1} and f_{ij2} and two different unitary capacities C_1 and C_2, formulation (F_5) models the problem of finding the cheapest network that supplies the demand requirements in the form of origin-destination pair demands.

Additional variables z_{ijt} are necessary to limit to one the number of technologies in a link. Variable z_{ijt} equals 1 if the link (i, j) has technology t installed and equals 0 otherwise. The coefficient U_t is the maximum number of units of capacity t that can be installed on a link.

The objective function (1.48) minimizes the sum of variable and fixed costs for both technologies. The demand requirements are supplied by one of the two technologies as stated by (1.49). Constraints (1.50) enforce the capacity limits and constraints (1.51) in conjunction with (1.52) ensure that only one type of technology may be installed in each individual link. The definition of x^k_{ijt} as positive, y_{ijt} as integer and z_{ijt} as binary variables is effected by constraints (1.53),(1.54) and (1.55), respectively.

(F_5) - Two-Facility CNDP Formulation

$$\text{Minimize} \sum_{(i,j)\in A} \sum_{t=1}^{2} \left(\sum_{k\in K} c_{ijt}^k x_{ijt}^k + f_{ijt} y_{ijt} \right) \tag{1.48}$$

subject to

$$\sum_{t=1}^{2} \left(\sum_{j|(i,j)\in A} x_{ijt}^k - \sum_{j|(j,i)\in A} x_{jit}^k \right) = \begin{cases} d_k, & i = O(k), \\ 0, & i \notin \{O(k), D(k)\}, \\ -d_k, & i = D(k), \end{cases} \quad k \in K, \tag{1.49}$$

$$\sum_{k\in K} x_{ijt}^k \leq C_t y_{ijt}, \qquad (i,j) \in A, \quad t = 1,2, \tag{1.50}$$

$$y_{ijt} \leq U_t z_{ijt}, \qquad (i,j) \in A, \quad t = 1,2, \tag{1.51}$$

$$\sum_{t=1}^{2} z_{ijt} \leq 1, \qquad (i,j) \in A, \tag{1.52}$$

$$x_{ijt}^k \geq 0, \qquad (i,j) \in A, \quad k \in K, \quad t = 1,2, \tag{1.53}$$

$$y_{ijt} \in \mathbb{Z}, \qquad (i,j) \in A, \quad t = 1,2, \tag{1.54}$$

$$z_{ijt} \in \{0,1\}, \qquad (i,j) \in A, \quad t = 1,2, \tag{1.55}$$

Magnanti, Mirchandani and Vachani (1995) report a very similar case in the design of private communication networks. In this case, both technologies may coexist in a link, having their capacities added up. Therefore, there is no need for the additional z_{ijt} variables nor for constraints (1.51) and (1.52). The technology index t may be also dropped from variables x_{ij}^k. Constraints (1.50) are therefore replaced by:

$$\sum_{k\in K} x_{ij}^k \leq C_1 y_{ij1} + C_2 y_{ij2}, \qquad (i,j) \in A. \tag{1.56}$$

As in the last subsection, the conversion of (F_5) into a *single-origin* model is straightforward.

We are not aware of any authors that have used Benders decomposition to solve the problem presented in this section.

The step-increasing cost CNDP

The last and more general case is the *Step-Increasing Cost CNDP* (Stoer and Dahl, 1994; Gabrel et al., 1999; Mahey et al., 2001). Here, the cost of *purchasing* capacity for a link is given by a step-increasing cost-capacity function. Formulation (F_6) models this last

case. This formulation creates additional binary variables to cope with the cost-capacity function discontinuities. In (F_6), C_t is the allowed capacity of a link if that link is in the t^{th} level of the cost function (refer to Figure 1.1) and T is the total number of these levels.

(F_6) - The Step-Increasing Cost Network Design Problem (Node-Arc Formulation)

$$\text{Minimize} \sum_{(i,j)\in A} \left(\sum_{k\in K} c_{ij}^k x_{ij}^k + \sum_{t=1}^{T} f_{ijt} y_{ijt} \right) \tag{1.57}$$

subject to

$$\sum_{j|(i,j)\in A} x_{ij}^k - \sum_{j|(j,i)\in A} x_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \notin \{O(k), D(k)\}, \\ -d_k, & i = D(k), \end{cases} \quad k \in K, \tag{1.58}$$

$$\sum_{k\in K} x_{ij}^k \leq \sum_t C_t y_{ijt}, \qquad (i,j) \in A, \tag{1.59}$$

$$\sum_{t=1}^{T} y_{ijt} \leq 1, \qquad (i,j) \in A, \tag{1.60}$$

$$x_{ij}^k \geq 0, \qquad (i,j) \in A, \quad k \in K, \tag{1.61}$$

$$y_{ijt} \in \{0,1\}, \qquad (i,j) \in A, \quad t = 1, \ldots, T. \tag{1.62}$$

Again, the objective function is to minimize the sum of variable and fixed costs, as expressed by (1.57). Constraints (1.58) guarantee that the demand requirements will be satisfied while constraints (1.59) limits the flow on each link according to the purchased capacity. Constraints (1.60) states that only one variable y_{ijt} may be selected, for these variables indicate the purchased capacity level, as shown in Figure 1.1.

It is usual to express the node-arc formulation (F_6) as an equivalent *arc-path* formulation. In this case, let $\mathbb{P}(k)$ be the set of possible paths between vertices $O(k)$ and $D(k)$. The coefficient $a_{kp(i,j)}$ equals 1 if the commodity k flowing on path p uses arc (i,j), and $a_{kp(i,j)} = 0$, otherwise. Also let n_{kp} be the flow of commodity k on path p. (F_6') presents an arc-path formulation for this last case.

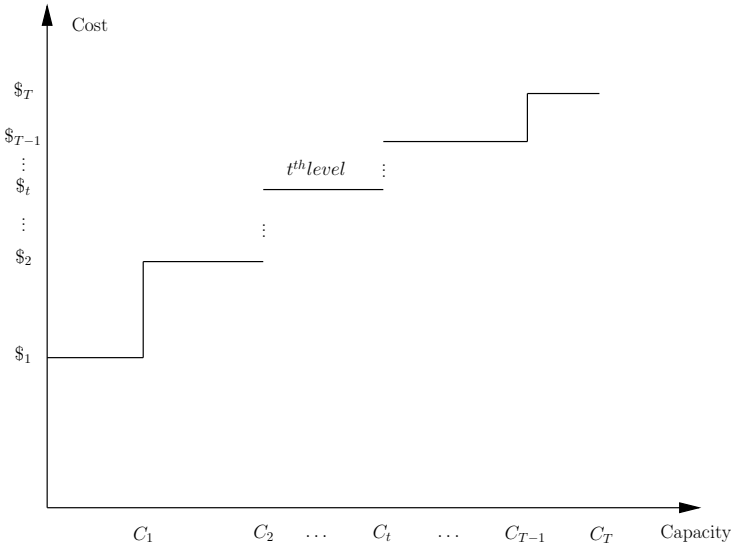


Figure 1.1: Step-increasing *cost* × *capacity* function.

(F_6') - The Step-Increasing Cost Network Design Problem (Arc-Path Formulation)

$$\text{Minimize} \sum_{(i,j)\in A} \left(\sum_{k\in K} c_{ij}^k x_{ij}^k + \sum_{t=1}^{T} f_{ijt} y_{ijt} \right) \tag{1.63}$$

subject to

$$x_{ij}^k = \sum_p a_{kp(i,j)} n_{kp}, \qquad (i,j) \in A, \quad k \in K, \tag{1.64}$$

$$\sum_p n_{pk} = d_k, \qquad k \in K, \tag{1.65}$$

$$\sum_{k\in K} x_{ij}^k \le \sum_t C_t y_{ijt}, \qquad (i,j) \in A, \tag{1.66}$$

$$\sum_{t=1}^{T} y_{ijt} \le 1, \qquad (i,j) \in A, \tag{1.67}$$

$$n_{kp} \ge 0, \qquad k \in K, \quad p \in \mathbb{P}, \tag{1.68}$$

$$x_{ij}^k \ge 0, \qquad (i,j) \in A, \quad k \in K, \tag{1.69}$$

$$y_{ijt} \in \{0,1\}, \qquad (i,j) \in A, \quad t = 1, \ldots, T. \tag{1.70}$$

Here, x_{ij}^k still represents the flow on each arc. Constraints (1.64) define the flow of a commodity on an arc as the sum of the flows of that commodity on all paths that use that arc. Equalities (1.65) are the demand constraints and inequalities (1.66) are the capacity constraints. Constraints (1.67) still state that only one variable y_{ijt} may be selected.

Arc-path formulations can be used to generate a natural reduced version of the Network Design problem by simply considering a limited number of paths for each origin-destination pair demand. We have exemplified for (F_6) but all the former formulations also have an equivalent arc-path model.

Application of Benders decomposition. Gabrel, Knippel and Minoux (1999) deal with this problem by using a set of metric inequalities to define the feasible region. The idea is to define a vector $\lambda = (\lambda_1, \ldots, \lambda_a, \ldots, \lambda_{|A|}) \in \mathbb{R}_+^{|A|}$. Then, $\Theta(\lambda)$ denotes the quantity $\sum_{k=1}^{K} d_k \times l_k(\lambda)$, where $l_k(\lambda)$ is the length of a shortest chain joining $O(k)$ and $D(k)$, when each arc in A is given the length $\lambda_a \geq 0$. Therefore, let $y = (y_1, \ldots, y_a, \ldots y_{|A|})$ be the selected capacities on each link. This solution is feasible if and only if (Onaga and Kakusho, 1971) for all $\lambda \in \mathbb{R}_+^{|A|}$, we have:

$$\sum_{a=1}^{|A|} \lambda_a y_a \geq \Theta(\lambda). \tag{1.71}$$

The authors then use these inequalities within a constraint generation approach where the master problem consists of maximizing the objective function (1.57) while the sub-problem generates the violated metric inequalities (1.71). They also extend the approach with the generation of multiple violated constraints by the use of bipartition inequalities (i.e., metric inequalities corresponding to bipartitions of the vertex set $|N|$).

Mahey, Benchakroun and Boyer (2001) have modified the problem of Gabrel et al. (1999) to introduce the issue of quality of service. Their version of the problem neglects the variable costs and uses an average delay function that can be regarded as a measure of the level of service or of the reliability of the network. The authors introduce variables $n_{(i,j)c}$, the flow of commodities in a link of capacity C. Therefore, the average delay on arc (i, j) given by Kleinrock's law (see e.g., Bertsekas and Gallager, 1987) is proportional to the non-linear expression $(n_{(i,j)c})/(C - n_{(i,j)c})$ which is introduced in the objective function. The term is convex for a fixed capacity but not jointly convex in $(C, n_{(i,j)c})$. Therefore, the authors use a Generalized Benders Decomposition approach that obtains a solution for the variables y_{ijt} in the master problem. The capacities are fixed in the auxiliary problem that reduces to a convex cost multicommodity flow problem, which is solved via a prox-

imal decomposition method that works as a separable augmented Lagrangean algorithm (Mahey et al., 1998). The auxiliary problem feeds the master problem with feasibility and optimality cuts and the algorithm iterates until no more cuts can be generated and the solution is proved to be optimal. The authors have also tested the effect of additional cuts (e.g., connexity cuts and spanning tree cuts) and concluded that they could strongly reduce the number of necessary iterations until convergence. The authors have tested instances derived from real private communications networks and containing up to 25 vertices, 62 arcs and 500 commodities.

An important class of problems that can be modeled as step-increasing cost network design problems is the planning of electric power transmission and distribution networks. In the power systems transmission (distribution) design problem, the goal is to establish a minimum cost set of transmission (distribution) lines in order to enable the flow of one commodity - electrical power - from the generation plants (substations) to the consumer centers.

Power Systems Transmission and Distribution Network Design Problems are generally non-convex. However, once the integer variables are fixed, the remaining model becomes a convex transportation problem. Generalized Benders Decomposition proposed by Geoffrion (1972) is therefore a natural solution approach, as argued in Section 1.2. Benchakroun, Ferland and Cléroux (1992) use decomposition to solve the *Multi-Period Power System Distribution Network Design Problem*. It is basically the *Step-Increasing Cost Network Design Problem* with the addition of several specific constraints related to the power flow characteristics, vertex capacities, radiality and voltage drop limits. A model with linear constraints is proposed and solved via a Generalized Benders decomposition. Gascon, Benchakroun and Ferland (1993) have improved the method by developing a Lagrangean relaxation based heuristic procedure to solve the master Problem. Tests were conducted on examples of up three periods, three sources (substations), ten load locations and 31 admissible arcs which proved the quality of the method.

For the *Transmission Network Design Problem*, the application of the method goes back to Pereira, Pinto, Cunha and Oliveira (1985). The authors propose two models for the problem (one simplified transportation model and a more sophisticated one including power flow equations) and solve both of them via Benders decomposition. When solving the second model the cuts generated for the simplified one are kept. The idea of keeping the cuts obtained for a simplified model in the resolution of a more complex formulation was

extended by Romero and Monticelli (1994a; 1994b) who proposed a three-phase hierarchical heuristic approach that gradually increases the complexity of the model. For each level of complexity, a solution is obtained via Benders decomposition and the cuts are propagated in the next phase. Validations tests were conducted on a real-life network of 45 vertices. A similar algorithm with only two phases is proposed by Oliveira, Costa and Binato (1995) and used to solve a real network of 79 bus (vertices) and 155 circuits (arcs).

Tsamasphyrou, Renaud and Carpentier (2000) introduced stochasticity in this problem by considering several availability (on the lines and generation units) and demand scenarios. The authors carried tests on an IEEE network of 24 vertices and 40 lines, considering cases of up to 2000 scenarios. Benders decomposition was considered for cases ranging from the situation where a single cut was generated at each iteration to the situation where, at each iteration, a cut was generated *for each* scenario. The authors concluded that the best approach lied somewhere between these two extremes, and divided the scenarios into a number (algorithm parameter) of sets, being one cut generated for each set of scenarios. This strategy divided the computation times by a factor ranging from 1.8 (1000 scenarios) to 3.8 (2000 scenarios)

Binato, Pereira and Granville (2001) solve the *Transmission Network Design* with additional constraints to deal with the power transmission stability problem. The authors make use of a disjunctive parameter in order to cope with the non-selected candidate arcs. Benders decomposition had been proposed before for this problem (Granville and Pereira, 1985) but numerical problems caused by the large values of the disjunctive parameter had limited the validity of the results. Binato et al. (2001) have proposed a new scheme where the disjunctive parameter is increased along the Benders iterations. Additional cuts are added at each iteration based on relaxations of the auxiliary problem and on Gomory's lifting procedure. A single real network of 46 vertices and 237 candidate arcs was considered and the authors showed that the proposed modifications helped reduce the computational time significantly.

1.4 Conclusions

Numerous practical applications can be formulated as network design problems. In these problems, the idea is to obtain a least cost network in order to satisfy some flow constraints, commonly expressed in the form of origin-destination demands. We have presented a review on Benders decomposition methods applied to Network Design. For-

mulations for these problems usually contain one set of integer variables associated with the selection of the arcs in the network, and one set of continuous variables associated with commodity flows. This structure offers a natural framework for the decomposition approach which consists of isolating the integer variables in the master problem and the flow variables in the auxiliary subproblem. Moreover, the relative ease of solving the auxiliary subproblem in network design formulations make of Benders decomposition one of the most appropriate approaches. Indeed, in most of the surveyed articles (see Table 1.1 for a summary), validations tests have indicated that Benders decomposition is an efficient method for solving network design problems, and may outperform traditional techniques such as Branch-and-Bound or Lagrangean relaxation. Efficient solution methodologies have also been obtained by combining Benders decomposition with other techniques, as proposed by Magnanti et al. (1986). A rich variety of these successful hybrid approaches is available in the literature.

In spite of this success, Benders decomposition has been mostly ignored for many years, not only for network design problems but for some of the other applications mentioned in Section 1.2. We believe that this tendency is slowly changing, given the increasing number of researchers using this technique, as shown in this chapter.

Reference	Type of Problem	Additional Constraints	Features of the Method	Size of the Problems Solved
Gavish (1982)	Single Origin UNDP	Constraints on the capacities of the edges connecting the origin vertex.	Cuts are generated if the master solution is not connected or if the flows obtained by the auxiliary subproblem violate the capacities.	$\|N\| \leq 12$.
Hoang (1982)	Origin-Destination pairs UNDP	Budgetary Constraints, nonlinear costs, no explicit upper bounds on arc flow.	Generalized Benders decomposition with master problem solved with a Lagrangean relaxation heuristic and subproblem via a convex cost multicommodity algorithm (Nguyen, 1974).	$\|N\| \leq 155$, $\|A\| \leq 376$.
Magnanti et al. (1986)	Origin-Destination pairs UNDP	None.	Preprocessing with dual-ascent procedure to eliminate variables. Use of Pareto-optimal cuts.	$\|N\| \leq 30$, $\|A\| \leq 130$.
Pereira et al. (1985)	Step-Increasing Cost CNDP	Power flow equations, decision also on the generation levels.	Standard algorithm with cuts generated for a simplified model kept in the resolution of a more complex formulation.	$\|N\| \leq 46$, $\|A\| \leq 73$.
Benchakroun et al. (1992), Gascon et al. (1993)	Step-Increasing Cost CNDP	Multi-period situation. Vertex-capacity, power flow and voltage drop limits constraints.	Generalized Benders decomposition with Lagrangean relaxation heuristic to solve the master.	$\|N\| \leq 13$ (3 sources and 10 loads) $\|A\| \leq 31$.
Romero and Monticelli (1994a; 1994b), Oliveira et al. (1995)	Step-Increasing Cost CNDP	Power flow equations, decision also on the generation levels.	Hierarchical heuristic approach with Benders decomposition used to solve each phase of a multi-step algorithm.	$\|N\| \leq 79$, $\|A\| \leq 155$.
Gutierrez et al. (1996)	Origin-Destination pairs UNDP	None. (Robust approach with multi-scenario objective function).	One master problem for each scenario. Cross generation of Benders cuts.	$\|N\| \leq 33$, $\|A\| \leq 100$ (11 scenarios).
Gabrel et al. (1999)	Step-Increasing Cost CNDP	None.	Generation of multiple violated constraints with bipartition inequalities.	$\|N\| \leq 20$, $\|A\| \leq 36$.

Table 1.1: Summary of the main Benders decomposition applications

Reference	Type of Problem	Additional Constraints	Features of the Method	Size of the Problems Solved
Tsamasphyrou et al. (2000)	Step-Increasing Cost CNDP	Power flow equations, stochasticity in the demands and availability of lines and generation unities.	Scenarios divided into a number (algorithm parameter) of sets and one cut generated for each set (instead of the extreme situations of one single cut for all scenarios or one cut for each scenario).	$\|N\| = 24$, $\|A\| = 40$. (up to 2000 scenarios).
Sridhar and Park (2000)	Single Facility CNDP	Vertex-capacity constraints.	Embedded Benders decomposition initiated with a lower bound given by a cutting plane algorithm and a feasible solution given by a heuristic method.	$\|N\| \leq 20$ (for various demand levels)
Randazzo and Luna (2001)	Single Origin UNDP	Redundant strengthening constraints to promote feasible solution in the Benders iterations.	Dual variables in Benders decomposition initialized with feasible solution obtained with a linear relaxation and shortest-path algorithm. Auxiliary subproblem decomposed by commodity.	$\|N\| \leq 100$, $\|A\| \leq 200$
Randazzo et al. (2001)	Two-Technologies Single Origin UNDP	Redundant strengthening constraints to promote feasible solution in the Benders iterations.	Dual variables in Benders decomposition initialized with feasible solution obtained with a shortest-path algorithm. Auxiliary subproblem decomposed by commodity.	$\|N\| \leq 41$, $\|A\| \leq 417$
Binato et al. (2001)	Step-Increasing Cost CNDP	Constraints to ensure the power system stability based on a disjunctive parameter.	Iterative increasing of disjunctive parameter solves earlier Benders numerical problem. In each iteration, several cuts are generated based on relaxed versions of the subproblem and on Gomory's lifting procedure.	$\|N\| = 66$, $\|A\| = 237$
Mahey et al. (2001)	Step-Increasing Cost CNDP	None. (Objective function with non-linear term representing the quality of service).	Generalized Benders decomposition with additional cuts. Convex optimization subproblem solved with the aid of a proximal decomposition method.	$\|N\| \leq 25$, $\|A\| \leq 62$
Ouorou et al. (2001)	Single Facility CNDP	Elastic demands; fixed-open links (expansion problem).	Generalized Benders decomposition. Convex optimization subproblem solved with the aid of a proximal decomposition method.	$\|N\| \leq 19$, $\|A\| \leq 68$

Table 1.2: Summary of the main Benders decomposition applications (continued)

Chapter 2

Benders, metric and cutset inequalities for multicommodity capacitated network design

Abstract

Multicommodity capacitated network design problems are hard combinatorial optimization problems for which several solution strategies have been developed. Several of these strategies involve the relaxation of some problem constraints and the strengthening of the model through the addition of valid inequalities. In this paper, we compare three sets of inequalities that have been widely used in this context: Benders, metric and cutset inequalities. We show that Benders inequalities associated with extreme rays are metric inequalities. We also show how to strengthen Benders inequalities associated with non-extreme rays to obtain metric inequalities. We show that cutset inequalities are Benders inequalities, but not necessarily metric inequalities. We give a necessary and sufficient condition for a cutset inequality to be a metric inequality. Finally, computational experiments show the effectiveness of strengthening Benders and cutset inequalities to obtain metric inequalities.
Keywords: Multicommodity capacitated network design, Benders decomposition, metric inequalities, cutset inequalities.

2.1 Introduction

Multicommodity capacitated network design problems are among those that have received the most attention (Atamtürk, 2002a; Atamtürk, 2002b; Gabrel et al., 1999; Günlük, 1999; Stoer and Dahl, 1994). For the sake of convenience, we will start by defining a typical network design problem on a directed network $G = (V, A, K)$ with vertex set V, arc set A, and commodity set K, each commodity k being represented by a triplet $(O(k), D(k), d_k)$, where $O(k)$ is the origin node, $D(k)$ is the destination vertex and $d_k > 0$ is the demand to be routed between $O(k)$ and $D(k)$. For each commodity k, we assume that there exists at least one path from $O(k)$ to $D(k)$ and we define the connected subnetwork $G^k = (V^k, A^k)$ containing only vertices and arcs that belong to some path from $O(k)$ to $D(k)$. Given an objective function (often non-convex), the problem is to determine a minimum cost solution satisfying the demands for all commodities, often subject to additional constraints, such as budgetary limits and topological restrictions.

In this context, many solution algorithms make use of the *multicommodity network flow subproblem*: given a capacity vector $\omega = (\omega_{ij}) \geq 0$, where $(i, j) \in A$, we wish to identify a multicommodity flow vector $x = (x_{ij}^k) \geq 0$, where $k \in K$ and $(i, j) \in A^k$, satisfying

$$\sum_{j \in V_i^k(+)} x_{ij}^k - \sum_{j \in V_i^k(-)} x_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \neq O(k), D(k), \quad i \in V^k, \quad k \in K, \\ -d_k, & i = D(k), \end{cases} \tag{2.1}$$

$$\sum_{k \in K} x_{ij}^k \leq \omega_{ij}, \qquad (i, j) \in A, \tag{2.2}$$

where $V_i^k(+) = \{j \in V^k | (i, j) \in A^k\}$ and $V_i^k(-) = \{j \in V^k | (j, i) \in A^k\}$. In this formulation, the value ω_{ij} indicates whether or not a connection is established on arc (i, j), and what is the capacity available to carry the demands through this arc. We say that ω is a *feasible* capacity vector if and only if there exists a multicommodity flow $x \geq 0$ satisfying (2.1) and (2.2).

For each commodity k, one of the flow conservation equations (2.1) is redundant. Hence, if we introduce dual variables $\pi = (\pi_i^k)$, where $i \in V^k$ and $k \in K$, associated with these equations, we can assume $\pi_{O(k)}^k = 0$ for each $k \in K$. Introducing also dual variables $\alpha = (\alpha_{ij}) \geq 0$, for $(i, j) \in A$, associated with the capacity constraints (2.2), we have, by strong duality in linear programming, that ω is feasible if and only if the following dual

problem, $D(\omega)$, is bounded:

$$\text{Maximize } Z(\omega) = \sum_{k \in K} d_k \pi^k_{D(k)} - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij} \tag{2.3}$$

subject to

$$\pi^k_j - \pi^k_i - \alpha_{ij} \leq 0, \qquad (i,j) \in A^k,\ k \in K. \tag{2.4}$$

Note that the feasible domain for $D(\omega)$, P_D, is a full-dimensional polyhedral cone. Any solution in P_D is called a *ray* and we know, by Minkowski's theorem (1896), that every such ray can be expressed as a non negative linear combination of the extreme rays of P_D (a ray is *extreme* if one cannot express it as the sum of other rays). We denote by $E(P_D)$ the set of extreme rays of P_D.

We study the relationships between four well-known classes of inequalities characterizing the feasibility of a capacity vector ω:

- *Benders inequalities.* The dual problem $D(\omega)$ is bounded if and only if $Z(\omega) \leq 0$ for all feasible dual solutions, i.e., if and only if

$$\sum_{k \in K} d_k \pi^k_{D(k)} - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij} \leq 0, \qquad (\pi, \alpha) \in P_D. \tag{2.5}$$

  We call these the Benders inequalities, since they are used as cuts in the famous Benders decomposition algorithm (Benders, 1962). When $\alpha = 0$, we obtain a so-called *trivial* Benders inequality.

- *Benders inequalities associated with extreme rays.* Since any ray can be expressed as a non negative linear combination of the extreme rays of P_D, we can restrict Benders inequalities to the set of extreme rays and yet obtain a characterization of the feasibility of ω:

$$\sum_{k \in K} d_k \pi^k_{D(k)} - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij} \leq 0, \qquad (\pi, \alpha) \in E(P_D). \tag{2.6}$$

- *Metric inequalities.* The dual problem $D(\omega)$ is bounded if and only if, for any $\alpha \geq 0$, we have

$$\sum_{(i,j) \in A} \omega_{ij} \alpha_{ij} \geq \max\{\sum_{k \in K} d_k \pi^k_{D(k)} | \pi^k_j - \pi^k_i \leq \alpha_{ij},\ (i,j) \in A^k,\ k \in K\}. \tag{2.7}$$

  The problem on the right-hand side of this inequality is the dual of a shortest path problem, with arc distances equal to $\alpha \geq 0$. Assume we solve this problem and obtain

the shortest path lengths $\pi(\alpha) = (\pi_i^k(\alpha))$, for $i \in V^k$ and $k \in K$ between $O(k)$ and i, for each vertex $i \in V^k$ and commodity $k \in K$. We then have the following metric inequalities:

$$\sum_{k \in K} d_k \pi^k_{D(k)}(\alpha) - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij} \leq 0, \qquad \alpha \geq 0. \tag{2.8}$$

Note that $(\pi(\alpha), \alpha) \in P_D$; hence, metric inequalities are a special case of Benders inequalities.

- *Cutset inequalities.* Let $S \subset V$ be a non-empty subset of V, and let $\overline{S} = V \setminus S$ be its complement; $(S, \overline{S})$ is a cutset, i.e., the set of arcs that connect a vertex in S to a vertex in $\overline{S}$. Also, let $K(S, \overline{S}) \subseteq K$ be the set of commodities having their origin in S and their destination in $\overline{S}$ and $d_{(S,\overline{S})} = \sum_{k \in K(S,\overline{S})} d^k$. We then define the following cutset inequalities:

$$\sum_{(i,j) \in (S,\overline{S})} \omega_{ij} \geq d_{(S,\overline{S})}, \qquad S \subset V,\ S \neq \emptyset. \tag{2.9}$$

These inequalities simply mean that there should exist sufficient capacity on the arcs of any cutset to satisfy the demands that must be routed through that cutset. They are often used in branch-and-cut algorithms for multicommodity capacitated network design problems (Barahona, 1996; Chouman et al., 2003).

Clearly, cutset inequalities are necessary for a capacity vector ω to be feasible, but it is well known that they are not in general sufficient (they are in the single-commodity case, due to the max flow-min cut theorem). As we have just seen, Benders inequalities, Benders inequalities associated with extreme rays, and metric inequalities are all necessary and sufficient to characterize a feasible capacity vector ω. In this sense, the three classes of inequalities are equivalent. However, when it comes to using these inequalities in a solution algorithm, the question of determining which class to use is not necessarily trivial, because even if one class of inequalities is a subclass of another, it may be easier to generate inequalities in the larger class. For example, in a Benders decomposition algorithm, generating only Benders cuts associated with extreme rays is not always better than using other Benders inequalities.

To compare inequalities, we use the standard *dominance* criterion. Given a polyhedron $P = \{u | Au \leq b\}$ and two valid inequalities $\lambda^1 u \leq \lambda^1_0$ and $\lambda^2 u \leq \lambda^2_0$ for P, we say that the first one dominates the second if $\lambda^2 u \leq \lambda^1 u$, $\forall u \in P$ and $\lambda^1_0 \leq \lambda^2_0$. Furthermore, if there exists $u \in P$ such that $\lambda^2 u < \lambda^1 u$, the dominance is strict. We prove the following properties:

1. Every Benders inequality associated with an extreme ray is a metric inequality.
2. For a given $\alpha \geq 0$, the corresponding metric inequality dominates all Benders inequalities associated with that particular α; as a consequence, any Benders inequality can be *strenghtened* to a metric inequality by solving the shortest path problem in (2.7).
3. Cutset inequalities are a subclass of Benders inequalities, but are not necessarily metric inequalities.
4. A necessary and sufficient condition can be given for a cutset inequality to be a metric inequality.

Our computational experiments, performed on instances of the multicommodity capacitated fixed-charge network design problem, illustrate the following results:

1. On many instances, a large proportion of cutset inequalities can be strengthened to metric inequalities.
2. For a standard Benders decomposition algorithm, restricting the method to Benders inequalities associated with extreme rays is not necessarily more efficient than generating cuts associated with any ray, extreme or not.
3. For the same algorithm, however, strengthening Benders cuts to generate only metric inequalities is computationally more efficient than using Benders inequalities generated by a linear programming solver.

The remainder of this chapter is organized as follows. In Section 2.2, we present our results concerning metric inequalities, comparing them to Benders inequalities. Section 2.3 is dedicated to our developments on cutset inequalities. Section 2.4 presents our computational results, while Section 2.5 contains some conclusions.

2.2 Benders and metric inequalities

In this section, we show two results that allow us to compare Benders and metric inequalities, and we illustrate these results on a simple example. We already know that Benders inequalities associated with extreme rays constitute a subclass of Benders inequalities. Our first result states that Benders inequalities associated with extreme rays are also a subclass of metric inequalities. To prove this result, we use the following simple lemma.

Lemma 2.1 *If (π, α) is a ray of P_D, then $\pi_h^k \leq \pi_h^k(\alpha)$, $\big(h \in V^k, k \in K\big)$.*

Proof: Let $k \in K$ and P_h^k a shortest path (with respect to α) from $O(k)$ to $h \in V^k$. Then, $\pi_h^k = \sum_{(i,j)\in P_h^k}(\pi_j^k - \pi_i^k) \leq \sum_{(i,j)\in P_h^k} \alpha_{ij} = \pi_h^k(\alpha)$.

Proposition 2.1 *Any non-trivial Benders inequality associated with an extreme ray is a metric inequality.*

Proof: Let (π, α) be an extreme ray of P_D associated with a non-trivial Benders inequality. Assume that this inequality is not a metric inequality.This implies that there exists some commodity $l \in K$ such that $\pi_{D(l)}^l \neq \pi_{D(l)}^l(\alpha)$. By Lemma 2.1, we must have $\pi_{D(l)}^l < \pi_{D(l)}^l(\alpha)$. This implies that, on every path P^l from $O(l)$ to $D(l)$, there exists an arc (i,j) such that $\pi_j^l - \pi_i^l - \alpha_{ij} < 0$, called a *non-binding arc*; otherwise, there would be a path Q^l such that $\pi_j^l - \pi_i^l - \alpha_{ij} = 0$ on every arc $(i,j) \in Q^l$, and $\pi_{D(l)}^l = \sum_{(i,j)\in Q^l}(\pi_j^l - \pi_i^l) = \sum_{(i,j)\in Q^l} \alpha_{ij} \geq \pi_{D(l)}^l(\alpha)$, a contradiction.

Using this property, we now construct a set $T^l \subset N^l$ as follows: each path P^l is visited from $O(l)$ to $D(l)$, starting from $O(l)$, until the first non-binding arc (i,j) is encountered; then add to T^l (initialized to $\emptyset$) all vertices on P^l from j to $D(l)$. Clearly, $O(l) \notin T^l$ and $D(l) \in T^l$, i.e., T^l defines a cutset. Moreover, it is not possible to have an arc (i,j) such that $i \in T^l$ and $j \notin T^l$. Indeed, $i \in T^l$ implies that there is a path from $O(l)$ to i containing a non-binding arc and $j \in N^l$ implies that there is a path from j to $D(l)$. Connecting these two paths through arc (i,j) yields a path from $O(l)$ to $D(l)$ containing a non-binding arc preceding vertex j. Hence, if $i \in T^l$, then $j \in T^l$.

Let $\delta = \min_{i\notin T^l, j\in T^l}\{-\pi_j^l + \pi_i^l + \alpha_{ij}\} > 0$, and define $(\overline{\pi}, \alpha)$ and $(\widetilde{\pi}, 0)$ as follows: $\overline{\pi} = \pi$, except for $\overline{\pi}_i^l = \pi_i^l + \delta$, $i \in T^l$; $\widetilde{\pi} = 0$, except for $\widetilde{\pi}_i^l = -\delta$, $i \in T^l$. We then have:

1. $(\overline{\pi}, \alpha)$ is a ray, since for any $(i,j) \in A^l$, there are three possible cases:
   (a) $i \in T^l$ and $j \in T^l$: $\overline{\pi}_j^l - \overline{\pi}_i^l - \alpha_{ij} = (\pi_j^l + \delta) - (\pi_i^l + \delta) - \alpha_{ij} = \pi_j^l - \pi_i^l - \alpha_{ij} \leq 0$;
   (b) $i \notin T^l$ and $j \notin T^l$: $\overline{\pi}_j^l - \overline{\pi}_i^l - \alpha_{ij} = \pi_j^l - \pi_i^l - \alpha_{ij} \leq 0$;
   (c) $i \notin T^l$ and $j \in T^l$: $\overline{\pi}_j^l - \overline{\pi}_i^l - \alpha_{ij} = (\pi_j^l + \delta) - \pi_i^l - \alpha_{ij} \leq \pi_j^l + (-\pi_j^l + \pi_i^l + \alpha_{ij}) - \pi_i^l - \alpha_{ij} = 0$.
2. $(\widetilde{\pi}, 0)$ is a ray, since for any $(i,j) \in A^l$, there are three possible cases:
   (a) $i \in T^l$ and $j \in T^l$: $\widetilde{\pi}_j^l - \widetilde{\pi}_i^l = -\delta - (-\delta) = 0$;
   (b) $i \notin T^l$ and $j \notin T^l$: $\widetilde{\pi}_j^l - \widetilde{\pi}_i^l = 0 - 0 = 0$;
   (c) $i \notin T^l$ and $j \in T^l$: $\widetilde{\pi}_j^l - \widetilde{\pi}_i^l = -\delta - 0 < 0$.

3. $(\pi, \alpha) = (\overline{\pi}, \alpha) + (\widetilde{\pi}, 0)$, which contradicts the hypothesis that (π, α) is an extreme ray.

□

Even though Benders inequalities associated with extreme rays subsume the two other classes of inequalities, some solution algorithms may generate Benders inequalities that are not necessarily associated with extreme rays. If (π, α) is a ray (not necessarily extreme) associated with a Benders inequality, one can solve a shortest path problem with respect to α and derive a metric inequality associated with the ray $(\pi(\alpha), \alpha)$. Our next result states that we then obtain a metric inequality that always dominates the original Benders inequality.

Proposition 2.2 *Any Benders inequality associated with ray (π, α) is dominated by the metric inequality associated with ray $(\pi(\alpha), \alpha)$. The dominance is strict if and only if there exists $l \in K$ such that $\pi^l_{D(l)} < \pi^l_{D(l)}(\alpha)$.*

Proof: Using the definition of dominance given in the introduction, it suffices to show that

$$\sum_{k \in K} d_k \pi^k_{D(k)} - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij} \leq \sum_{k \in K} d_k \pi^k_{D(k)}(\alpha) - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij}.$$

This is immediate since, by Lemma 2.1, for each $k \in K$, the relations $d^k > 0$ and $\pi^k_{D(k)} \leq \pi^k_{D(k)}(\alpha)$ hold. It also follows immediately that the dominance is strict if and only if there exists $l \in K$ such that $\pi^l_{D(l)} < \pi^l_{D(l)}(\alpha)$.□

Example 1: Figure 2.1 depicts a network with three vertices, two arcs and two commodities, with demands d_1 and d_2, having their origins at vertex 1 and their destinations at vertices 2 and 3, respectively. Once we fix to 0 the dual variables associated with redundant constraints of the multicommodity flow subproblem (i.e., flow conservation equations for commodity 1, vertices 1 and 3, and for commodity 2, vertices 1 and 2) we obtain the following full-dimensional dual polyhedron:

$$P_D = \{(\pi, \alpha) = ((\pi^1_2, \pi^2_3), (\alpha_{12}, \alpha_{13})) | \pi^1_2 - \alpha_{12} \leq 0,\ \pi^2_3 - \alpha_{13} \leq 0,\ \alpha_{12}, \alpha_{13} \geq 0\}.$$

We can easily enumerate the extreme rays of P_D:

$(\pi(1), \alpha(1)) = ((1, 0), (1, 0))$;
$(\pi(2), \alpha(2)) = ((0, 1), (0, 1))$;
$(\pi(3), \alpha(3)) = ((-1, 0), (0, 0))$;

$(\pi(4), \alpha(4)) = ((0, -1), (0, 0))$.

There are two non-trivial Benders inequalities associated with these extreme rays:

$\omega_{12} \geq d_1$ (associated with $(\pi(1), \alpha(1))$);
$\omega_{13} \geq d_2$ (associated with $(\pi(2), \alpha(2))$).

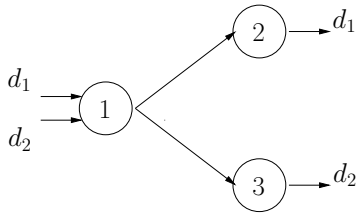


Figure 2.1: Network for Example 1

If $\alpha = (1, 1)$, we obtain the metric inequality $\omega_{12} + \omega_{13} \geq d_1 + d_2$, associated with the ray $((1, 1), (1, 1) = (\pi(1), \alpha(1)) + (\pi(2), \alpha(2))$, i.e., this metric inequality is obtained by aggregating the two non-trivial Benders inequalities associated with extreme rays. In other words, the combination of the two Benders inequalities associated with extreme rays dominates this metric inequality, but neither of them, individually, dominates it. Note that there exist other Benders inequalities associated with $\alpha = (1, 1)$, for instance $\omega_{12}+\omega_{13} \geq d_1$ (associated with the ray $(1, 0), (1, 1)$) and $\omega_{12}+\omega_{13} \geq d_2$ (associated with the ray $(0, 1), (1, 1)$), which are not metric inequalities and are dominated by $\omega_{12}+\omega_{13} \geq d_1+d_2$.

Suppose we perform two iterative solution algorithms. The first one generates, at each iteration, one Benders inequality associated with an extreme ray. The second algorithm generates, at each iteration, one Benders inequality and strengthens it to a metric inequality by solving a shortest path problem with α fixed at its value in the generated Benders inequality. Assume that, at some iteration, both algorithms solve the multicommodity flow subproblem with $\overline{\omega}$ satisfying $\overline{\omega}_{12} < d_1$ and $\overline{\omega}_{13} < d_2$, and that $d_2 - \overline{\omega}_{13} > d_1 - \overline{\omega}_{12}$. Suppose that the first algorithm generates the Benders inequality $\omega_{13} \geq d_2$. The second algorithm first generates the Benders inequality $\omega_{12} + \omega_{13} \geq d_2$ and then strengthens it to the metric inequality $\omega_{12} + \omega_{13} \geq d_1 + d_2$. This inequality is violated by $\overline{\omega}$ by the quantity $(d_2 - \overline{\omega}_{13}) + (d_1 - \overline{\omega}_{12}) > d_2 - \overline{\omega}_{13}$, the amount by which the Benders inequality $\omega_{13} \geq d_2$ is violated.

This example shows that, even if we select the best Benders inequality associated with an extreme ray, we may obtain a better metric inequality associated with a non-extreme

ray. Thus, even though we can restrict ourselves to Benders inequalities associated with extreme rays, it may be interesting from a computational point of view to generate other Benders inequalities, in particular metric inequalities. Our computational results confirm this observation.

A final remark is that one could certainly attempt to generate metric inequalities directly instead of generating Benders inequalities and then strengthening them to obtain metric inequalities. Indeed, Gabrel et al. (1999) have proposed a cutting plane algorithm based on metric inequalities for a multicommodity capacitated network design problem. In their algorithm, a heuristic method is used to generate violated metric inequalities. In general, however, it is harder to derive violated metric inequalities than to obtain violated Benders inequalities, in which case a good quality linear programming solver can be used to solve the multicommodity flow subproblem.

2.3 Cutset inequalities

In this section, we investigate the relationships between cutset inequalities and the other three classes of inequalities defined in the introduction. We first show that cutset inequalities are a subclass of Benders inequalities. To this purpose, we need to introduce, for any $S \subset V$, $S \neq \emptyset$, the ray $(\pi(S), \alpha(S))$ defined as follows:

$$\pi_i^k(S) = \begin{cases} 1, & \text{if } i \in \overline{S}, O(k) \in S \text{ and } D(k) \in \overline{S}, \qquad i \in V^k,\ k \in K, \\ 0, & \text{otherwise,} \end{cases}$$

$$\alpha_{ij}(S) = \begin{cases} 1, & \text{if } i \in S \text{ and } j \in \overline{S}, \qquad (i,j) \in A, \\ 0, & \text{otherwise.} \end{cases}$$

Proposition 2.3 *For any $S \subset V$, $S \neq \emptyset$, the cutset inequality associated with S corresponds to the Benders inequality associated with the ray $(\pi(S), \alpha(S))$.*

Proof: By definition of $(\pi(S), \alpha(S))$, we have $\pi^k_{D(k)}(S) = 1$ for each commodity $k \in K(S, \overline{S})$ and $\alpha_{ij}(S) = 1$ if and only if $(i,j) \in (S, \overline{S})$ for each arc (i,j), which implies:

$$\sum_{k \in K} d_k \pi^k_{D(k)}(S) - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij}(S) = \sum_{k \in K(S,\overline{S})} d_k - \sum_{(i,j) \in (S,\overline{S})} \omega_{ij} = d_{(S,\overline{S})} - \sum_{(i,j) \in (S,\overline{S})} \omega_{ij}.$$

The Benders inequality $\sum_{k \in K} d_k \pi^k_{D(k)}(S) - \sum_{(i,j) \in A} \omega_{ij} \alpha_{ij}(S) \leq 0$ is thus equivalent to the cutset inequality $d_{(S,\overline{S})} - \sum_{(i,j) \in (S,\overline{S})} \omega_{ij} \leq 0$.

In general, a cutset inequality is not necessarily a metric inequality (consequently, it is not necessarily a Benders inequality associated with an extreme ray). The next proposition provides a necessary and sufficient condition for a cutset inequality to be a metric inequality.

Proposition 2.4 *For any $S \subset V$, $S \neq \emptyset$, the cutset inequality associated with S is a metric inequality if and only if for each $k \in K$*
a) if $O(k) \in S$ and $D(k) \in \overline{S}$, there exists a path from $O(k)$ to $D(k)$ that crosses $(S, \overline{S})$ only once;
b) if $O(k) \in \overline{S}$ or $D(k) \in S$, there exists a path from $O(k)$ to $D(k)$ that never crosses $(S, \overline{S})$.

Proof: Assume we solve the shortest path problem associated with the distances $\alpha(S)$ and obtain the shortest path lengths $\pi(\alpha(S))$. By definition of $\alpha(S)$, for each $k \in K$, there are three possible outcomes:

1. $\pi^k_{D(k)}(\alpha(S)) = 0$, which means that there exists a path from $O(k)$ to $D(k)$ that never crosses $(S, \overline{S})$;
2. $\pi^k_{D(k)}(\alpha(S)) = 1$, which means that there exists a path from $O(k)$ to $D(k)$ that crosses $(S, \overline{S})$ only once;
3. $\pi^k_{D(k)}(\alpha(S)) > 1$, which means that there is no path from $O(k)$ to $D(k)$ that never crosses $(S, \overline{S})$ or that crosses $(S, \overline{S})$ only once.

Now, assume that the cutset inequality associated with $S \subset V$, $S \neq \emptyset$, is a metric inequality. For any $k \in K$, we then have $\pi^k_{D(k)}(\alpha(S)) = \pi^k_{D(k)}(S)$, and

- if $O(k) \in S$ and $D(k) \in \overline{S}$, then, by definition of $\pi(S)$, we have $1 = \pi^k_{D(k)}(S) = \pi^k_{D(k)}(\alpha(S))$, which implies that there exists a path from $O(k)$ to $D(k)$ that crosses $(S, \overline{S})$ only once;
- if $O(k) \in \overline{S}$ or $D(k) \in S$, then, by definition of $\pi(S)$, we have $0 = \pi^k_{D(k)}(S) = \pi^k_{D(k)}(\alpha(S))$, which implies that there exists a path from $O(k)$ to $D(k)$ that never crosses $(S, \overline{S})$.

Conversely, assume that, for each $k \in K$, we have either

- $O(k) \in S$ and $D(k) \in \overline{S}$, in which case there exists a path from $O(k)$ to $D(k)$ that crosses $(S, \overline{S})$ only once, which implies that $\pi^k_{D(k)}(\alpha(S)) = 1 = \pi^k_{D(k)}(S)$, by definition of $\pi(S)$;

or

- $O(k) \in \overline{S}$ or $D(k) \in S$, in which case there exists a path from $O(k)$ to $D(k)$ that never crosses $(S, \overline{S})$, which implies $\pi^k_{D(k)}(\alpha(S)) = 0 = \pi^k_{D(k)}(S)$, by definition of $\pi(S)$.

Thus, for each $k \in K$, $\pi^k_{D(k)}(\alpha(S)) = \pi^k_{D(k)}(S)$, and the cutset inequality associated with S is a metric inequality. □

Example 2: Consider the network depicted in Figure 2.2, with four vertices 1, 2, 3 and 4, and three arcs $(1,2)$, $(3,4)$ and $(4,1)$. There are two commodities, 1 and 2, with demands d_1 and d_2, respectively, and $O(1) = 4$, $O(2) = 3$, $D(1) = D(2) = 2$. Consider the cutset $S = \{1,3\}$. The subnetworks $G^k = (V^k, A^k)$ for $k = 1, 2$ are shown in Figure 2.3, along with the ray $(\pi(S), \alpha(S))$ associated with S. For each commodity k, the value inside each vertex i corresponds to $\pi^k_i(S)$, while the value close to each arc (i,j) is $\alpha_{ij}(S)$. This ray is associated with the Benders inequality $\omega_{12} + \omega_{34} \geq d_2$. The length of the shortest path (with respect to $\alpha(S)$) between $O(2)$ and $D(2)$ is equal to 2. Moreover, even though neither $O(1)$ nor $D(1)$ are in set S, the shortest path length between $O(1)$ and $D(1)$ is equal to 1. Therefore, strengthening the cutset inequality yields the metric inequality $\omega_{12} + \omega_{34} \geq d_1 + 2d_2$, which dominates the original cutset inequality.

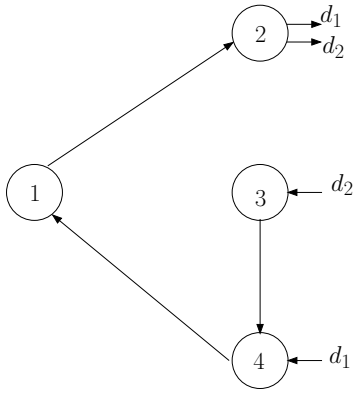


Figure 2.2: Network for Example 2 - four vertices, three arcs and two commodities.

The intuition behind this strengthening is very simple. Observe that in Example 2 the only path between $O(2)$ ($\in S$) and $D(2)$ ($\in \overline{S}$) must use more than one arc from the cutset (in this case, arcs $(3,4)$ and $(1,2)$), because it needs to reenter set S. Therefore, $\pi^2_{D(2)}(\alpha(S))$ actually represents the number of times *an arc crossing the cutset* has to be

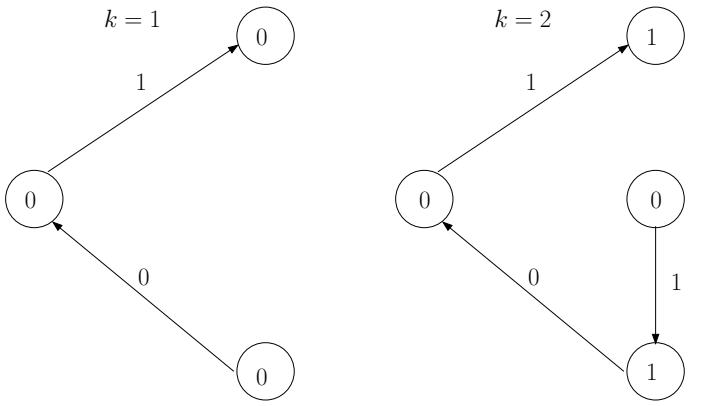


Figure 2.3: Subnetworks for each commodity and ray associated with the cutset inequality for Example 2

used in order for the commodity to leave its origin $O(2)$ and reach its destination $D(2)$. The cutset inequality does not take this information into account (as if a single use of the cutset arcs would always be sufficient) and therefore presents a worse estimate of the required capacity. This strengthening also considers commodities for which neither the origin nor the destination is in set S, but still need to use some capacity across the cutset, as is the case for commodity 1.

As final remark on Example 2, it is interesting to note that the same metric inequality can be obtained as the sum of two cutset inequalities: one associated with $S = \{3\}$, which yields the inequality $\omega_{34} \geq d_2$, and the other associated with $S = \{1, 3, 4\}$, which yields the inequality $\omega_{12} \geq d_1 + d_2$. There are cases, however, in which one cannot obtain the strengthened metric inequality by adding up cutset inequalities, as shown in the next example.

Example 3: Consider the network of Example 2 with the inclusion of arc $(3, 2)$. Again, we consider the cutset defined by $S = \{1, 3\}$. The subnetworks for the two commodities and the ray associated with this cutset inequality are shown in Figure 2.4. The cutset inequality is $\omega_{12} + \omega_{34} + \omega_{32} \geq d_2$. The strengthened metric inequality is $\omega_{12} + \omega_{34} + \omega_{32} \geq d_1 + d_2$, which cannot be obtained as the sum of other cutset inequalities. Note, however, that it is dominated by the cutset inequality associated with $S = \{1, 3, 4\}$, which is $\omega_{12} + \omega_{32} \geq d_1 + d_2$.

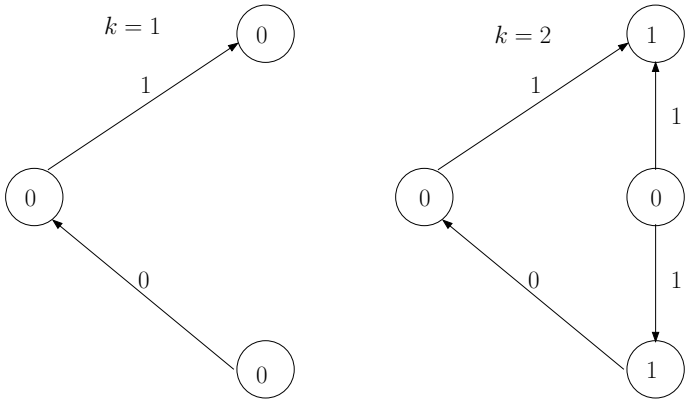


Figure 2.4: Subnetworks for each commodity and ray associated with the cutset inequality for Example 3

The last two examples show that strengthened metric inequalities can be dominated by other cutset inequalities. In spite of this fact, strengthening cutset inequalities is interesting, because solution algorithms usually make use of heuristic methods to generate cutset inequalities. Hence, they do not guarantee that the best cutsets are generated. The proposed strengthening can be useful in that context, as confirmed by our computational experiments, reported in Section 2.4.1.

2.4 Computational results

To perform our computational experiments, we use instances of the *Multicommodity Capacitated Fixed-Charge Network Design Problem* (MCFNDP), for which several exact and approximate solution algorithms have been proposed (Gendron and Crainic, 1994; Gendron and Crainic, 1996; Holmberg and Yuan, 2000; Crainic et al., 2000; Sellman et al., 2002; Ghamlouche et al., 2003; Ghamlouche et al., 2004; Crainic et al., 2004). To each arc $(i, j) \in A$ are associated a binary variable y_{ij} taking value 1 if and only if the arc is chosen, and continuous variables x_{ij}^k, $k \in K$, representing the flow of commodity k on the arc. Parameters f_{ij} and c_{ij}^k are, respectively, the fixed cost for opening arc (i, j) and the variable cost associated with the flow of one unit of commodity k through arc (i, j), while u_{ij} is the capacity of arc (i, j) when it is open. With this notation, one can write the MCFNDP as follows:

$$\text{Minimize} \sum_{(i,j)\in A} \left(f_{ij} y_{ij} + \sum_{k\in K} c_{ij}^k x_{ij}^k \right) \tag{2.10}$$

subject to

$$\sum_{j\in V_i^k(+)} x_{ij}^k - \sum_{j\in V_i^k(-)} x_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \neq O(k), D(k), \quad i \in V^k, \quad k \in K, \\ -d_k, & i = D(k), \end{cases} \tag{2.11}$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij} y_{ij}, \qquad (i,j) \in A, \tag{2.12}$$

$$x_{ij}^k \geq 0, \qquad (i,j) \in A, \quad k \in K, \tag{2.13}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{2.14}$$

Our computational experiments are based on three sets of instances widely used in the literature on the MCFNDP. These instances contain from 10 to 100 vertices, 35 to 687 arcs, and 10 to 400 commodities. They are described in detail by Ghamlouche et al. (2004).

In the next section, we present results obtained on cutset inequalities generated by an enumeration of all cutsets, up to a limited cardinality. We are interested in the proportion of cutset inequalities that are strengthened, as well as in the improvement obtained by the strengthened metric inequalities. In Section 2.4.2, we analyze the results obtained by a Benders decomposition algorithm applied to the MCFNDP. We investigate, in particular, the relative performance of three variants of the algorithm: one that generates Benders inequalities associated with rays that are not necessarily extreme; a second variant that generates only Benders inequalities associated with extreme rays; and a third variant that generates Benders inequalities and strengthens them to obtain only metric inequalities.

2.4.1 Cutset inequalities

In the special case of cutset inequalities, we have considered for each instance all sets S, with $|S| \leq \Psi$, where Ψ is an integer which depends on the number of vertices in the graph (see Tables 2.1 and 2.2). All the associated cutsets have been generated and tested. In Table 2.1 the results are shown for the first group of instances (group R), while Table 2.2 shows the results obtained for the third group (group C+). For the second group (group C), considering sets with $|S| \leq 4$, no cutset inequality has been strengthened. In the tables, for each instance, we show the following information:

- $(|V|, |A|, |K|)$ - Number of vertices, arcs and commodities in the instance;

- Ψ - maximum cardinality of the tested cutsets;
- Nb. Cutsets - number of cutsets tested;
- Nb. Strength - number of cutset inequalities that were strengthened;
- Max. - the maximum strength obtained;
- Avg. - the average strength considering all the strengthened inequalities;

where the *strength* is defined as $\left[\sum_{k\in K} d^k \pi^k_{D(k)}(\alpha) - \sum_{k\in K} d_k \pi^k_{D(k)}\right] / \sum_{k\in K} d^k \pi^k_{D(k)}(\alpha)$.

Inst.	$(\|V\|,\|A\|,\|K\|)$	Ψ	Nb. Cutsets	Nb. Strength	Max. (%)	Avg. (%)
R01.1	(10,35,10)	5	637	32	42.11	32.35
R02.1	(10,35,25)	5	637	76	34.97	19.41
R03.1	(10,35,50)	5	637	63	24.02	10.51
R04.1	(10,60,10)	5	637	0	-	-
R05.1	(10,60,25)	5	637	0	-	-
R06.1	(10,60,50)	5	637	0	-	-
R07.1	(10,82,80)	5	637	0	-	-
R08.1	(10,83,25)	5	637	0	-	-
R09.1	(10,83,50)	5	637	0	-	-
R10.1	(20,120,40)	5	6195	2	14.70	14.18
R11.1	(20,120,100)	5	6195	48	12.26	4.28
R12.1	(20,120,200)	5	6195	67	4.98	2.92
R13.1	(20,220,40)	5	6195	0	-	-
R14.1	(20,220,100)	5	6195	0	-	-
R15.1	(20,220,200)	5	6195	0	-	-
R16.1	(20,314,40)	5	6195	0	-	-
R17.1	(20,318,100)	5	6195	0	-	-
R18.1	(20,315,200)	5	6195	0	-	-

Table 2.1: Effect of strengthening on cutset inequalities (group R)

The results presented in Tables 2.1 and 2.2 indicate that the strengthening is only effective for sparse graphs. This result was expected, since the higher the arc density, the higher the probability that the conditions given in Proposition 2.4 are satisfied. Indeed, consider a subset S of the vertices of a complete graph, and its complement $\overline{S}$. If some commodity has its origin in S and its destination in $\overline{S}$, it is always possible to connect the origin to the destination by using a single arc across the cutset. Similarly, if we consider a commodity with its origin in $\overline{S}$ or its destination in S, we can reach the destination from the origin by an arc that does not cross the cutset. Thus, in the case of a complete graph, strengthening cutset inequalities is useless, since they are already metric inequalities.

Inst.	$(\|V\|,\|A\|,\|K\|)$	Ψ	Nb. Cutsets	Nb. Strength	Max. (%)	Avg. (%)
c25_100_10_F_L_5	(25,100,10)	4	15,275	22	100.00	45.71
c25_100_10_F_T_5	(25,100,10)	4	15,275	41	100.00	88.09
c25_100_10_V_L_5	(25,100,10)	4	15,275	160	100.00	65.77
c25_100_30_F_L_5	(25,100,30)	4	15,275	143	50.00	18.45
c25_100_30_F_T_5	(25,100,30)	4	15,275	868	100.00	25.07
c25_100_30_V_T_5	(25,100,30)	4	15,275	785	100.00	29.80
c100_400_10_F_L_10	(100,400,10)	3	166,750	3	100.00	84.84
c100_400_10_F_T_10	(100,400,10)	3	166,750	6	100.00	60.73
c100_400_10_V_L_10	(100,400,10)	3	166,750	98	100.00	56.00
c100_400_30_F_L_10	(100,400,30)	3	166,750	595	100.00	63.47
c100_400_30_F_T_10	(100,400,30)	3	166,750	304	100.00	75.91
c100_400_30_V_T_10	(100,400,30)	3	166,750	592	100.00	63.18

Table 2.2: Effect of strengthening on cutset inequalities (group C+)

2.4.2 Benders inequalities

We have developed a standard Benders decomposition algorithm which obtains values for the dual variables (α, π) by using CPLEX to solve the multicommodity flow subproblem at each iteration of the algorithm.

The Benders algorithm first relaxes all feasibility and optimality constraints in the master problem. At each iteration, the relaxed master problem provides a lower bound on the optimal solution value of the original problem and a solution $\overline{y}$. These variables define a tentative network, and are used in the dual subproblem.

If the dual subproblem is unbounded, we obtain a ray (not necessarily extreme) that can be used to generate a violated Benders inequality, also called a feasibility cut. If the dual subproblem is bounded, the conjunction of the master problem and subproblem solutions yields a feasible solution to the original problem and provides an upper bound. In this case, the extreme point corresponding to an optimal dual solution can be used to generate a so-called optimality cut. The process iterates until the values of the lower and upper bounds coincide.

We have analyzed the number of iterations and the time it takes for this Benders decomposition algorithm to find a first feasible solution. We compare these two figures with and without strengthening the feasibility cuts to obtain metric inequalities. The number of iterations performed before a first feasible solution is found gives a measure of how much we can strengthen the cuts. The comparison of the computation time between the two cases is useful to determine whether the time spent in strengthening is rewarded by the gain in the strength of the cuts.

In Tables 2.3 to 2.7, the following results are shown:

- With or Without Strength - indicates the use (or not) of the strengthening:

  Nb. Iter. - number of iterations needed to find a first feasible solution;

  Time (s) - time in seconds needed to find a first feasible solution;

- $\neq$ (%) - percentage difference between the equivalent columns with and without strengthening;

- Strength info. - information associated with the strengthening:

  Nb. - number of iterations where the cut was strengthened;

  Max. Strength - maximum strength obtained (%);

  Avg. Strength - average strength obtained (%).

Tables 2.3 and 2.4 present average results for the first group of instances. We see from these tables that strengthening the feasibility cuts not only reduces the number of iterations needed to find a feasible solution for most instances, but also the time needed, indicating that the time spent in solving the shortest path problem is rewarded by the strengthening of the cuts. Note that, for each line in the tables, there is actually a set of instances corresponding to the same network, but having different configurations of fixed and variable costs (Ghamlouche et al., 2004).

	Without Strength		With Strength		$\neq$ (%)		Strength Info.		
Instance	Nb. Iter.	Time (s)	Nb. Iter.	Time (s)	Nb. Iter.	Time	Nb.	Max.	Avg.
R01	45.50	0.10	37.67	0.09	-17.15	-9.14	11.00	28.81	16.06
R02	54.83	0.20	41.33	0.17	-24.90	-18.78	19.67	47.41	19.46
R03	58.50	0.39	44.50	0.31	-23.46	-20.49	29.00	48.79	19.41
R04	31.44	0.08	31.22	0.09	-0.60	12.53	0.33	5.64	16.91
R05	71.78	0.48	66.67	0.46	-6.61	-2.61	17.44	13.87	4.94
R06	94.56	1.49	76.78	1.19	-14.45	-16.23	33.78	20.33	6.82
R07	38.22	0.14	37.11	0.15	-2.89	4.86	1.89	7.63	6.18
R08	79.44	0.81	81.89	0.87	3.70	6.64	11.67	9.86	4.03
R09	117.22	2.79	96.22	2.02	-16.28	-25.87	36.25	25.80	5.86
Mean					-10.10	-6.40	17.25	20.87	9.61

Table 2.3: Results for Benders inequalities (group R 10 vertices)

	Without Strength		With Strength		≠ (%)		Strength Info.		
Instance	Nb. Iter.	Time (s)	Nb. Iter.	Time (s)	Nb. Iter.	Time	Nb.	Max.	Avg.
R10	265.44	10.86	225.89	9.01	-12.66	-13.38	112.56	32.11	8.99
R11	282.33	48.18	204.44	34.04	-24.18	-27.83	136.44	62.92	14.57
R12	209.67	57.20	141.67	36.32	-30.02	-30.98	116.11	57.36	16.22
R13	431.78	97.43	399.89	91.72	-8.12	-6.84	117.89	17.51	3.99
R14	760.56	483.62	597.78	393.95	-17.93	-17.69	339.44	37.95	5.03
R15	542.00	708.65	353.78	483.58	-27.13	-23.70	255.22	62.34	7.83
R16	389.44	159.52	351.22	144.46	-9.60	-4.34	80.56	17.57	3.23
R17	1202.00	2315.23	983.78	1958.08	-18.05	-15.04	519.67	35.36	4.44
R18	978.89	3413.53	654.00	2471.57	-28.03	-26.11	446.22	35.34	4.86
Mean					-19.53	-18.44	236.01	39.83	7.68

Table 2.4: Results for Benders inequalities (group R 20 vertices)

	Without Strength		With Strength		≠ (%)		Strength Info.		
Instance	Nb. Iter.	Time (s)	Nb. Iter.	Time (s)	Nb. Iter.	Time	Nb.	Max.	Avg.
c33	532	111.48	478	103.62	-10.15	-7.59	140	20.13	3.88
c35	454	141.42	393	120.56	-13.44	-17.30	88	16.54	3.40
c36	492	197.67	450	168.32	-8.54	-17.44	85	28.55	3.90
c37	2227	8041.07	1589	6298.42	-28.65	-27.67	1007	44.68	3.83
c38	2909	12738.5	1948	8906.8	-33.04	-43.02	1337	56.38	3.91
c39	2469	11205.6	1655	8391.61	-32.97	-33.53	999	47.60	4.06
c40	2653	12295.4	1750	9123.37	-34.04	-34.77	1105	58.41	3.17
c41	602	325.07	557	306.45	-7.48	-6.08	148	33.80	3.87
c42	588	331.64	553	282.48	-5.95	-17.40	68	15.35	3.17
c43	630	588.32	643	629.94	2.06	6.61	88	38.89	3.70
c44	911	893.24	793	658.8	-12.95	-35.59	107	13.71	3.08
c45	2959	18532.3	2179	14782	-26.36	-25.37	1367	46.06	2.65
c46	3024	18241.6	2206	14690.9	-27.05	-24.17	1300	32.49	2.60
c47	3222	21230	2193	15997.3	-31.94	-32.71	1294	45.43	2.61
c48	2772	15250.2	1991	12312.1	-28.17	-23.86	1127	52.36	2.71
Mean					-19.91	-22.66	684	36.69	3.37

Table 2.5: Results for Benders inequalities (group C)

The results for the second group of instances are presented in Tables 2.5 and 2.6. These instances are harder than those in the first group. Indeed, for about half of these instances (those in Table 2.6), the Benders algorithm could not find a single feasible solution in

the allotted computation time of five hours. Therefore, Table 2.6 does not compare the difference between the total time to find a feasible solution, but the lower bound upon termination. Although it is a different measure, the lower bounds are also relevant since they somehow indicate the strength of the cuts.

	Without Strength		With Strength		≠ (%)	Strength Info.		
Instance	Nb. Iter.	LB	Nb. Iter.	LB	LB	Nb.	Max.	Avg.
c49	849	3677.12	658	3658.39	-0.51	637	28.51	4.65
c50	1077	16381.4	766	16341.7	-0.24	736	32.41	4.65
c51	679	2895.79	549	2930.23	1.19	515	26.01	4.13
c52	760	17175.7	555	17320.2	0.84	529	26.76	4.62
c53	1061	10920.4	560	11977	9.68	553	78.01	11.56
c54	886	20131.3	472	21416.9	6.39	461	49.34	9.25
c55	769	11105.6	436	11956.4	7.66	423	78.92	8.30
c56	741	22470.3	421	24242.4	7.89	410	54.51	8.90
c57	388	2595.49	344	2676.2	3.11	317	24.44	4.59
c58	432	5140.63	347	5233.47	1.81	316	28.91	4.68
c59	501	2512.28	381	2538.63	1.05	352	27.53	3.80
c60	287	5271.1	236	5299.48	0.54	222	21.37	3.64
c61	690	8728.18	338	9245.4	5.93	314	63.75	13.24
c62	587	17684.2	317	19029.1	7.61	306	45.13	11.41
c63	628	8467.78	325	8988.69	6.15	310	49.36	9.33
c64	492	17317.9	264	18521.3	6.95	243	40.97	10.22
Mean					4.13	415.25	42.25	7.31

Table 2.6: Results for Benders inequalities (group C - time limit exceeded)

	Without Strength		With Strength		≠ (%)		Strength Info.		
Instance	Nb. Iter.	Time (s)	Nb. Iter.	Time (s)	Nb. Iter.	Time	Nb.	Max.	Avg.
$c_{100_400_10FL10}$	897	106.42	902	105.5	0.56	-0.86	56	21.88	9.84
$c_{100_400_10FT10}$	2170	491.34	2083	495.85	-4.01	0.92	59	33.49	11.62
$c_{100_400_10VL10}$	1361	312.08	1352	254.95	-0.66	-18.31	88	22.22	9.40
$c_{100_400_30FL10}$	6109	10871.3	5240	9678.83	-14.22	-10.97	1620	40.20	5.19
$c_{100_400_30FT10}$	3374	2361.29	3027	2085	-10.28	-11.70	1070	41.78	7.34
$c_{25_100_10FL5}$	171	1.06	158	1.09	-7.60	2.83	31	36.84	12.72
$c_{25_100_10FT5}$	133	0.71	127	0.73	-4.51	2.82	14	25.68	14.11
$c_{25_100_10VL5}$	115	0.64	95	0.54	-17.39	-15.63	25	25.81	17.07
$c_{25_100_30FL5}$	300	6.21	233	5.09	-22.33	-18.04	105	45.16	13.13
$c_{25_100_30FT5}$	226	3.7	164	2.86	-27.43	-22.70	99	69.80	22.32
$c_{25_100_30VT5}$	233	4.04	182	3.22	-21.89	-20.30	118	51.31	15.21
Mean					-11.80	-10.18	298.63	37.65	12.54

Table 2.7: Results for Benders inequalities (group C+)

For the instances of Table 2.5, a mean reduction of 22% in the computation time needed

to obtain the first feasible solution was obtained by using strengthened cuts. The results are consistent with those of the first group of instances and indicate that the harder the problem, the more effective the strengthening seems to be in reducing the total computation time.

An interesting result is obtained in Table 2.6. In this case, the time limit of five hours was not sufficient to generate a single feasible solution. However, one observes that the lower bounds obtained by the Benders decomposition algorithm using strengthening were slightly better (mean increase of 4.13%) than those obtained without strengthening. This indicates that the behavior is similar to the other two groups of instances. Finally, Table 2.7 shows the results for the last group of instances. Again, these results are similar to those for the other groups.

	Benders inequalities		Benders inequalities (extreme rays)	
Instance	Nb. Iter.	Time (s)	Nb. Iter.	Time (s)
R01	40.33	0.14	**34.67**	**0.13**
R02	44.17	0.24	**40.00**	0.24
R03	43.67	0.39	**37.00**	**0.34**
R04	**32.44**	**0.15**	32.89	0.17
R05	**67.33**	**0.65**	71.22	0.81
R06	72.33	1.32	**64.56**	**1.26**
R07	34.00	**0.19**	**33.56**	0.24
R08	81.89	1.15	**73.56**	1.15
R09	98.44	2.74	**87.44**	**2.70**
R10	225.11	**11.59**	**202.56**	14.58
R11	205.56	**37.95**	**154.11**	38.10
R12	143.22	38.00	**116.89**	**33.45**
R13	396.33	107.65	**340.78**	**89.32**
R14	**581.33**	**441.89**	694.11	782.58
R15	**347.00**	**492.39**	414.22	1760.19
R16	347.11	158.10	**325.33**	**126.93**
R17	**963.44**	**1938.99**	1413.00	5348.33
R18	**638.78**	**2530.32**	820.11	9646.16

Table 2.8: Effect of using Benders inequalities associated with extreme rays

A final study concerns the relative strength of Benders inequalities and Benders inequalities associated with extreme rays. It is common in the literature to use extreme rays when solving problems by means of a Benders decomposition approach, since they are believed to yield stronger cuts. When solving an unbounded subproblem with CPLEX we had the option of obtaining an extreme ray or a ray (not necessarily extreme). We tested both options on the first group of instances and the results are summarized in Table 2.8.

These results somehow contradict what is commonly believed, since no clear answer can be given to the question of which is the best option. Our interpretation is that, although the set of extreme rays is limited (and therefore, common sense would lead us to believe that the associated Benders feasibility cuts are stronger), combining two or more Benders feasibility cuts may sometimes be more effective in cutting the feasible space.

2.5 Conclusions

We have clarified the relationships between three well-known classes of inequalities used in solution algorithms for multicommodity capacitated network design problems: Benders, metric and cutset inequalities. We have shown that Benders inequalities associated with extreme rays are always metric, but that it can be interesting to generate other Benders inequalities, and to strengthen them into metric inequalities. This has been demonstrated for the particular case of cutset inequalities, a subclass of Benders inequalities for which we have provided a necessary and sufficient condition for them to be metric inequalities. Computational results on a Benders decomposition algorithm have shown that, for some instances, the time needed to identify a first feasible solution was reduced by more than 30% when strengthening Benders feasibility cuts to obtain metric inequalities.

Chapter 3

Benders decomposition applied to multicommodity capacitated fixed-charge network design

Abstract

Network design problems concern the selection of arcs in a graph in order to satisfy, at minimum cost, some flow requirements. In this chapter, we deal with the multicommodity capacitated fixed-charge network design problem, for which we propose a Benders decomposition method. The approach hierarchically performs the integer (arc selection) and continuous (flow) decisions, and iterates between them in order to obtain optimality. Several extensions to the standard Benders decomposition algorithm are suggested, in particular, the generation of extra cuts during the solution of the linear relaxation of the problem which allows a significant reduction in the number of mixed-integer problems to be solved.
Keywords: Benders decomposition; network design; fixed-charge.

3.1 Introduction

In this chapter, we deal with the *Multicommodity Capacitated Fixed-Charge Network Design Problem* (MCFNDP), a family of network design problems where multiple commodities with given demand, origin and destination must be considered. Each link has a fixed cost and a maximum capacity as well as edge-related variable costs associated with the flows of commodities. The problem is to select a set of links such that the demand of

each commodity can be routed from its origin to its destination while minimizing the sum of fixed and variable costs. The MCFNDP can be modeled as a mixed-integer program, with binary variables associated with the opening of the links and continuous variables associated with the flow of commodities. A formulation has been presented in Section 2.4. It is an NP-hard problem for which several solution methods have been proposed.

Algorithms based on Lagrangean relaxations have been proposed by Gendron and Crainic (1994; 1996). The authors use subgradient optimization to approximate the optimal values of the Lagrangean duals in a lower bounding procedure. This procedure is combined in a *composite-procedure* with an upper bounding resource-decomposition heuristic. Holmberg and Yuan (2000) also use Lagrangean relaxation but within a branch-and-bound framework. As before, a subgradient algorithm is used to optimize the Lagrangean dual.

Crainic et al. (2001) have approached the problem with bundle-based relaxation methods. Their work calibrates and compares bundle and subgradient methods applied to the optimization of duals arising from two Lagrangean relaxations. The authors recommend the use of bundle methods since these appear to be more robust (relatively to different relaxations, problem characteristics and choice of initial parameters) and converge faster. They also demonstrate that effective lower bounds can be obtained with these methods for large scale problems, in a fraction of the time needed by a standard simplex method to solve the linear relaxation.

Tabu search heuristics have also been proposed. Crainic et al. (2000) have developed a tabu search method that explores the space of flow variables by combining pivot moves with column generation, creating a procedure that explores the space of continuous flow variables while evaluating the actual mixed integer objective. Ghamlouche et al. (2003) have used a tabu search algorithm with special neighborhoods called *cycle-based neighborhoods*. These neighborhoods define moves that explicitly take into account the impact on the total design cost of potential modifications to the distribution of several commodities simultaneously.

Chouman et al. (2003) have studied the mixed integer formulation of the MCFNDP and proposed a cutting-plane algorithm which incorporates valid inequalities based on cutsets of the network, generated by an algorithm developed by the authors. They also proposed separation heuristics and lifting procedures. The authors have obtained better lower bounds with competitive computational effort when compared to the results obtained by CPLEX.

In another article, Crainic et al. (2004) have proposed a slope scaling / Lagrangean perturbation heuristic with long-term memory. The slope scaling procedure is an iterative

scheme that solves linear approximations of the problem and adjusts the costs in order to represent the total mixed-integer (linear and fixed) costs. The authors apply the slope scaling concurrently with a Lagrangean bounding procedure and add a long term memory that stores information from the past iterations.

In this chapter we propose a new algorithm based on Benders decomposition, a classical optimization method first proposed by Benders (1962) and which has been applied to different mixed-integer problems with success, as presented in Chapter 1. Benders decomposition exploits a natural characteristic of the problem which has two sets of interrelated decisions: the selection of the open arcs and the routing of the flows. This method naturally adapts to this structure by partitioning the problem. At a higher level, called *master problem*, a tentative network is obtained by setting the values of the binary variables. This tentative network is then used in a *subproblem* whose purpose is to obtain values for the flow variables while satisfying the demands constraints.

Many authors have used Benders decomposition to solve uncapacitated versions of network design problems. Magnanti et al. (1986), for example, have solved the uncapacitated equivalent of the MCFNDP. Other versions of uncapacitated network design problems solved with Benders include the single-origin case (see Gavish, 1982 and Randazzo et al., 2001) and the two-technologies single-origin case (see Randazzo and Luna, 2001).

Benders decomposition has also been used to solve different versions of capacitated network design problems. With this approach, Sridhar and Park (2000) have solved the single-facility case in which multiple units of capacity can be installed on each arc. The problem also included capacities on the incoming and outgoing flow on the vertices. They have proposed a Benders enumeration algorithm initiated by a lower bound given by a cutting plane algorithm. This Benders-and-cut algorithm outperformed a conventional branch-and-bound on instances of up to 20 vertices.

Ouorou et al. (2001) have solved a similar version with the additional consideration of the elasticity of the demands (the demands vary according to the charged costs). The authors have dealt with the capacity expansion problem where there is already sufficient capacity installed and the question is whether it would be profitable to expand the service. A generalized Benders algorithm has been used to solve instances based on real-life data with up to 19 vertices, 68 arcs and 342 commodities, and the results suggest that Benders decomposition is an effective strategy.

Finally, Gabrel et al. (1999) have studied a more general version where the fixed cost for purchasing capacity for a link is given by a step-increasing cost-capacity function. The

authors have used a set of metric inequalities to define the feasible region (Onaga and Kakusho, 1971) and have embedded them within a constraint generation approach. They also have extended the approach with the generation of multiple violated constraints at each iteration. This modification enables them to solve instances of up to 20 vertices and 36 arcs.

In this chapter we study the application of a Benders decomposition approach to the MCFNDP. In the next section, we show basic Benders decomposition approach for the MCFNDP and in Section 3.3 we present some algorithmic refinements used to increase the efficiency of the method. In Section 3.4, we show the computational results and in Section 3.5, we end the paper with some concluding remarks and suggestions for future work.

3.2 Benders decomposition

Benders decomposition (Benders, 1962) is a classical solution approach for combinatorial optimization problems, based on the ideas of partition and delayed constraint generation. The method partitions the model to be solved into two simpler formulations, called master problem and subproblem. The master problem is a relaxed version of the original problem, containing only a subset of the original variables and the associated constraints. The subproblem is the original problem with the variables obtained in the master problem fixed. In the case of the MCFNDP, the master problem deals with the integer variables defining the network, while the subproblem works with the continuous variables representing the actual flows of commodities for the tentative network obtained in the master problem. The original MCFNDP formulation presented in Chapter 2 is reproduced here for convenience.

$$\text{Minimize} \sum_{(i,j)\in A} \left(f_{ij} y_{ij} + \sum_{k\in K} c_{ij}^k x_{ij}^k \right) \tag{3.1}$$

subject to

$$\sum_{j\in V_i^k(+)} x_{ij}^k - \sum_{j\in V_i^k(-)} x_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \neq O(k), D(k), \\ -d_k, & i = D(k), \end{cases} \quad i \in V^k, \quad k \in K, \tag{3.2}$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij}y_{ij}, \qquad (i,j)\in A, \tag{3.3}$$

$$x_{ij}^k \geq 0, \qquad (i,j)\in A, \quad k\in K, \tag{3.4}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j)\in A. \tag{3.5}$$

In this chapter, we implement several extensions to the standard Benders algorithm used in Chapter 2, in order to try to improve its efficiency and use it as a tool to solve large instances of problem (3.1)-(3.5). Before introducing these extensions, we first particularize the developments presented in Chapter 2, obtaining the following subproblem, in which the y variables are fixed:

$$\text{Minimize} \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k x_{ij}^k \tag{3.6}$$

subject to

$$\sum_{j\in V_i^k(+)} x_{ij}^k - \sum_{j\in V_i^k(-)} x_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \neq O(k), D(k), \\ -d_k, & i = D(k), \end{cases} \quad i\in V^k, \quad k\in K, \tag{3.7}$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij}\overline{y}_{ij}, \qquad (i,j)\in A, \tag{3.8}$$

$$x_{ij}^k \geq 0, \qquad (i,j)\in A, \quad k\in K. \tag{3.9}$$

The master problem, in turn, can be written in terms of the cuts obtained for every extreme point and extreme ray associated with the dual feasible space of (3.7)-(3.9):

$$\text{Minimize} \sum_{(i,j)\in A} f_{ij}y_{ij} + z \tag{3.10}$$

subject to

$$\sum_{k\in K} d_k\pi_{D(k)}^k - \sum_{(i,j)\in A} w_{ij}\alpha_{ij} \leq 0, \qquad (\pi,\alpha)\in R_D. \tag{3.11}$$

$$z \geq \sum_{k\in K} d_k\pi_{D(k)}^k - \sum_{(i,j)\in A} u_{ij}\overline{y}_{ij}\alpha_{ij}, \qquad (\pi,\alpha)\in E_D. \tag{3.12}$$

where, R_D and E_D are the sets of extreme rays and extreme points associated with the dual feasible space of (3.7)-(3.9), respectively. Inequalities (3.11) are the extreme Benders (feasibility) cuts while inequalities (3.12) are the Benders optimality cuts.

3.3 Algorithm refinements

In this section we describe some refinements to improve the performance of the basic Benders decomposition algorithm.

3.3.1 Relaxed master problem

If we relax the integrality constraints on variables y_{ij}, and proceed with the decomposition approach, we will obtain the solution of the linear relaxation of the original problem. However, all the cuts obtained are valid not only for the relaxed problem but also for the original integer one. Since solving the linear relaxation is much faster than solving the integer problem, it is usually useful to solve the relaxed version in order to obtain an initial set of cuts to be applied in the integer decomposition. This initial set of cuts reduces the number of integer iterations needed for convergence, compensating for the time spent in the relaxed problem and making the whole process more efficient. This idea, first suggested by McDaniel and Devine (1977), was thus used in our implementations.

3.3.2 Cutset inequalities

The cutset inequalities introduced in Section 2.1 are a classical set of valid inequalities for the MCFNDP. These inequalities are usually used with the intent of speeding up the solution procedure. In this work, cutset inequalities with $|S| = 1$ and $|S| = 2$ have been introduced as *a priori* constraints in the master problem.

3.3.3 Generating Pareto-optimal cuts

Magnanti and Wong (1981) have studied the influence of cuts in a Benders decomposition algorithm and showed that the use of stronger cuts may have a great impact on the convergence of the algorithm by reducing the number of iterations. The idea is to make use of the existence of multiple optimal solutions to the dual problem (2.3)-(2.4) to obtain better cuts. To determine what is a better cut the authors use the notion of dominance: for the presented reformulation of MCFNDP, a cut generated from the extreme point $(\overline{\pi}, \overline{\alpha})$ dominates a cut generated from the extreme point $(\underline{\pi}, \underline{\alpha})$ if

$$\sum_{k \in K} d_k \overline{\pi}^k_{D(k)} - \sum_{(i,j) \in A} u_{ij} \overline{y}_{ij} \ \overline{\alpha}_{ij} \geq \sum_{k \in K} d_k \underline{\pi}^k_{D(k)} - \sum_{(i,j) \in A} u_{ij} \overline{y}_{ij} \ \underline{\alpha}_{ij} \tag{3.13}$$

for all values of $\overline{y}_{ij}$ that yield a bounded subproblem and with strict inequality for at least one point. A cut that is not dominated by any other cut is said to be Pareto-optimal. Let Y denote the set of binary vectors yielding a feasible primal subproblem. For $\overline{y} \in Y$, let $v(\overline{y})$ denote the optimal value of this subproblem. In the case of the MCFNDP, Pareto-optimal cuts for a given solution $\overline{y}$ can be obtained by solving the following linear program:

$$\text{Maximize} \sum_{k \in K} d_k \pi^k_{D(k)} - \sum_{(i,j) \in A} u_{ij} y^0_{ij} \alpha_{ij} \tag{3.14}$$

subject to

$$\pi^k_j - \pi^k_i - \alpha_{ij} \leq c^k_{ij}, \qquad (i,j) \in A,\ k \in K, \tag{3.15}$$

$$\sum_{k \in K} d_k \pi^k_{D(k)} - \sum_{(i,j) \in A} u_{ij} \overline{y}_{ij} \alpha_{ij} = v(\overline{y}), \tag{3.16}$$

$$\alpha_{ij} \geq 0, \qquad (i,j) \in A, \tag{3.17}$$

where y^0_{ij} is a solution in the relative interior of Y. We call problem (3.14)-(3.17) the *auxiliary dual subproblem*. The objective function maximizes the strength of the cut with respect to the interior point y^0_{ij} while constraint (3.16) limits the feasible space to the set of optimal solutions to the dual subproblem for $\overline{y}_{ij}$.

Associating a free variable q with constraint (3.16), we can write the auxiliary subproblem in its primal form:

$$\text{Minimize} \sum_{(i,j) \in A} \sum_{k \in K} c^k_{ij} x^k_{ij} + v(\overline{y}) q \tag{3.18}$$

subject to

$$\sum_{j \in N^+_i} x^k_{ij} - \sum_{j \in N^-_i} x^k_{ji} \begin{cases} +d_k q &= d_k, & i = O(k), \\ &= 0, & i \notin \{O(k), D(k)\}, \\ -d_k q &= -d_k, & i = D(k), \end{cases} \quad i \in N, \quad k \in K, \tag{3.19}$$

$$\sum_{k \in K} x^k_{ij} + u_{ij} \overline{y}_{ij} q \leq u_{ij} y^0_{ij}, \qquad (i,j) \in A, \tag{3.20}$$

$$x^k_{ij} \geq 0, \qquad (i,j) \in A, \quad k \in K. \tag{3.21}$$

The primal form (3.18)-(3.21) is a convenient way of expressing the problem in our implementation, since it can be obtained with a few modifications to the primal subproblem.

3.3.4 Additional bounds on the flow variables

Some network design problems may include individual arc capacities for each commodity, which limit the flow of each particular commodity on an open arc :

$$x^k_{ij} \leq b^k_{ij} y_{ij}. \tag{3.22}$$

In most cases, there is no such limit but only the total flow limit expressed in constraints (1.22). Bounds (3.22) can still be constructed by using as b_{ij}^k the minimum value between d_k and u_{ij}. In this case, these bounds are clearly redundant for the original problem but not for the LP relaxation (see Gendron and Crainic, 1994 and Crainic et al., 2001). The use of these additional bounds reduces the integrality gap between the original formulation and its LP relaxation. As we first solve the relaxed version of the problem (see Section 3.3.1), the bounds are important because they enable the generation of additional cuts that reduce the number of iterations needed to solve the integer (and more difficult) problem.

3.3.5 Strengthening the feasibility cuts and the cutset inequalities

We use the results obtained in Chapter 2 in order to strengthen both the feasibility constraints and the cutset inequalities.

3.3.6 Reducing the number of *integer* iterations

The Benders decomposition algorithm presented takes the general form depicted in **Algorithm 3.1**.

Algorithm 3.1 Benders Decomposition

Add Initial cuts;
$LB = 0, UB = \infty$;
solve Benders LP relaxation (**Algorithm 3.2**);
add LP extra cuts;
solve Benders IP (**Algorithm 3.3**).

Solution procedures for the Benders LP relaxation and for the Benders IP problem are depicted in **Algorithms 3.2** and **3.3**. Temporarily ignore the steps in *italic*. Our first computational tests showed that even with the Benders LP relaxation phase, there was still the need of several iterations of the Benders IP solver (**Algorithm 3.3**), in order to find the solution for the original problem. During the LP relaxation, a set of cuts is generated with the hope that these cuts will help solving the IP. However, in the IP phase, the algorithm is conducted to another area of the search space: an area that has not been much explored when solving the LP relaxation. It is important to observe that this is an important drawback, since each single iteration of the IP phase is computationally much more expensive than several LP iterations. Indeed, in our preliminary tests, almost all the time was spent in the IP phase and, in particular, in solving the master problem. In order

to overcome this limitation, several steps have been added to the original algorithm. These steps are emphasized in **Algorithms 3.2** and **3.3**. The basic idea of all these additional steps is to generate extra cuts at each iteration, especially during the LP phase, so that the cuts obtained will more accurately describe the search space before entering the IP phase. Note that these cuts are not introduced in the problem during the relaxed phase, but only in the beginning of the IP phase.

Algorithm 3.2 Benders LP relaxation

```
while LB < UB − ε do
  Solve Master;
  update LB;
  change constraints in SubProblem;
  solve SubProblem.
  if SubProblem is Feasible then
    set UB;
    change constraints in AuxProblem;
    solve AuxProblem;
    add violated Bij constraints (if any: set UB =∞);
    add optimality cut;
    generate extra relaxed feasible cuts (Algorithm 3.4).
  else
    add feasibility cut;
    generate extra relaxed infeasible cuts (Algorithm 3.5);
  end if
end while
```

In the sequel we present the algorithms used for the generation of extra cuts. The basic mechanism in all of them is to slightly modify the Y solution (the definition of open and closed arcs) and re-solve the subproblem in order to obtain a new cut.

Generating extra cuts during the relaxed phase

During the LP phase, each master iteration gives a tentative fractional Y solution. If these values for the y variables yield a feasible subproblem we have a complete solution for the original problem, including values for the flows on the arcs. In this case, as shown by **Algorithm 3.4**, we use the flow variables in order to initialize a Slope Scaling algorithm

Algorithm 3.3 Benders IP

```
while LB < UB − ε do
  while master IP not solved do
    Get next master integer solution;
    update LB;
    change constraints in SubProblem;
    solve SubProblem.
    if SubProblem is feasible then
      update UB;
      change constraints in AuxProblem;
      solve AuxProblem;
      add optimality cut;
      generate extra integer feasible cuts (Algorithm 3.6);
      generate extra cuts.
    else
      generate feasibility cut.
    end if
  end while
  add feasibility and optimality cuts.
end while
```

(see Crainic et al., 2004). Slope Scaling is a heuristic that tries to obtain a good solution to the MCFNDP by solving successive linear problems. At each iteration the linear costs are adjusted to reflect both the linear and fixed costs incurred by the former solution. The result obtained by the Slope Scaling algorithm is used to set the subproblem, which is solved once again in order to obtain an extra cut.

Additional extra cuts are retrieved by solving the subproblem for perturbed versions of the solution given by the Slope Scaling algorithm. The perturbations try to simulate the expected structure of the integer solution. This is done by closing, one by one, the arcs with the largest fixed cost to flow ratios, $(f_{ij}/flow_{ij})$, since it is expected that these are the currently used arcs with the smallest probability of remaining in the final solution. To some extent, with these procedures, we intend to generate feasibility cuts that otherwise would only be generated in the integer phase, when the algorithm would "realize" that the price for using the arc is too high.

Algorithm 3.4 Generate extra relaxed feasible cuts

```
Get flow values;
(ρ) : ρ_ij = f_ij / flow_ij;
solve Slope Scaling (and get Y solution);
change constraints in SubProblem;
solve SubProblem.
if SubProblem is feasible then
    change constraints in AuxProblem;
    solve AuxProblem;
    generate optimality cut.
else
    generate feasibility cut.
end if
while nc < numberOfCutsToAdd do
    ij = nc^th argmax ρ;
    temp = y_ij;
    y_ij = 0;
    change constraints in SubProblem;
    solve SubProblem.
    if SubProblem is feasible then
        change constraints in AuxProblem;
        solve AuxProblem;
        generate optimality cut.
    else
        generate feasibility cut.
    end if
    y_ij = temp;
    nc++.
end while
```

In the case of an infeasible subproblem, the pertubations are made by giving priority to the arcs with a high value of y, as shown by **Algorithm 3.5**. The idea is to increase the value of these arcs (since the high flow versus low fixed cost indicates that they will probably be open in the final solution) in order to generate optimality cuts that, again,

would otherwise only be generated in the integer phase of the Benders decomposition. Whenever the modified Y yields a feasible subproblem, the Slope Scaling algorithm is used to better take into account the fixed costs.

Algorithm 3.5 Generate extra relaxed infeasible cuts

loop
  $ij =$ argmax $y_{ij} \in Y | y_{ij} < 1$;
  $y_{ij} = 1$;
  change constraints in SubProblem;
  solve SubProblem.
  if SubProblem is feasible **then**
    change constraints in AuxProblem;
    solve AuxProblem;
    generate optimality cut;
    $(\rho) : \rho_{ij} = f_{ij}/flow_{ij}$;
    solve Slope Scaling;
    change constraints in SubProblem;
    solve SubProblem.
    if SubProblem is feasible **then**
      change constraints in AuxProblem;
      solve AuxProblem;
      generate optimality cut.
    else
      generate feasibility cut.
    end if
    BREAK.
  else
    generate feasibility cut.
  end if
end loop

Generating extra cuts during the integer phase

During the integer phase it was also important to try to obtain the maximum quantity of information at each iteration.

As depicted in **Algorithm 3.3**, an important strategy was to generate new cuts (by solving the subproblem) for each new better integer solution found during the branch-and-bound procedure used to solve the master problem. This way, instead of generating a single cut at each iteration, we could generate several good cuts. The quality of these cuts comes from the fact that the solutions obtained during the branch-and-bound procedure are good perturbations of the final best solution.

Moreover, extra cuts were also generated by perturbing the Y solution, in a similar way to that conducted during the LP phase, as can be observed in **Algorithm 3.6**.

Algorithm 3.6 Generate extra integer feasible cuts

```
Get flow values;
(ρ) : ρ_ij = f_ij / flow_ij.
while nc < numberOfCutsToAdd do
  ij = nc^th argmax ρ;
  temp = y_ij;
  y_ij = 0;
  change constraints in SubProblem;
  solve SubProblem.
  if SubProblem is feasible then
    change constraints in AuxProblem;
    solve AuxProblem;
    add optimality cut.
  else
    add feasibility cut.
  end if
  y_ij = temp;
  nc++.
end while
```

3.4 Computational results

We have tested the Benders decomposition algorithm with the extensions proposed in Section 3.3. For these tests, a widely used set of instances was selected. The algorithm was coded in C++ and the linear problems were solved with the aid of CPLEX 8.1. We used an Intel Pentium 4 with a 3.00GHz CPU and 512 KB of memory and we allowed a

maximum time of two hours for each instance.

Preliminary tests have indicated that each of the refinements worked in the sense that it made the algorithm more efficient. In particular, the additional bounds on the flow variables (see Section 3.3.4) has proved to be quite beneficial since it could extend the generation of cuts in the relaxed phase of the solution and therefore reduce the number of (computationally expensive) integer iterations.

However, even though the improvements over the standard version of the Benders algorithm were considerable, these are far from sufficient to make the algorithm competitive with the basic CPLEX branch-and-cut algorithm. The main reason for this seems to be the need for a large amount of iterations before the algorithm can converge. More importantly, even after solving the relaxed problem, there was still the need for a considerable number of integer iterations. Our tests suggest that the generation of extra cuts can be useful to reduce the need for so many integer iterations, but only at the cost of a larger time spent within the relaxed phase. The overall method is useful when this extra time compensates for the gain in time due to the reduction of integer iterations. We would expect that this would happen for larger instances. Table 3.1 shows that this was indeed the case for some large instances. However, due to the inefficiency of the Benders algorithm cuts for the MCFNDP, it was not possible to test larger instances than those showed in the tables. We expect that the algorithm would have been better on these instances if it could have been tested. Moreover, we expect the extra-cuts generation procedures to be useful for other problems where the Benders decomposition approach is competitive.

Inst.		Benders No Extra Cuts				Benders With Extra Cuts			
Nb.	(N/A/K)	Time	LP	IP	gap	Time	LP	IP	gap
R01.1	10/35/10	0.36	77	2	0.00	0.43	77	0	0.00
R01.2	10/35/10	0.53	108	2	0.00	0.6	108	1	0.00
R01.3	10/35/10	0.84	149	4	0.00	1.03	149	2	0.00
R01.4	10/35/10	5.01	332	31	0.00	4.84	332	16	0.00
R01.5	10/35/10	5.59	230	26	0.00	2.6	230	3	0.00
R01.6	10/35/10	7.94	336	25	0.00	5.03	336	7	0.00
R02.1	10/35/25	29.63	315	80	0.00	15.01	315	8	0.00
R02.2	10/35/25	28.47	252	65	0.00	13.4	252	6	0.00
R02.3	10/35/25	13.15	252	38	0.00	11.8	252	6	0.00
R02.4	10/35/25	3.15	105	35	0.00	5.44	105	9	0.00
R02.5	10/35/25	2.91	129	15	0.00	6.02	129	4	0.00
R02.6	10/35/25	2.87	130	11	0.00	4.72	130	2	0.00
R03.1	10/35/50	7.62	253	7	0.00	9.54	253	3	0.00
R03.2	10/35/50	12.92	315	18	0.00	20.45	327	5	0.00
R03.3	10/35/50	16.58	363	19	0.00	26.42	349	8	0.00
R03.4	10/35/50	3.74	95	18	0.00	8.78	95	6	0.00
R03.5	10/35/50	5.25	129	13	0.00	11.07	129	4	0.00
R03.6	10/35/50	6.62	141	19	0.00	19.57	141	5	0.00
R04.1	10/60/10	1.61	206	2	0.00	1.77	206	0	0.00
R04.2	10/60/10	1.62	201	2	0.00	2.3	205	1	0.00
R04.3	10/60/10	1.85	217	2	0.00	2.05	217	1	0.00
R04.4	10/60/10	3.13	177	21	0.00	3.22	177	5	0.00
R04.5	10/60/10	4.95	256	13	0.00	4.99	256	5	0.00
R04.6	10/60/10	3.47	221	9	0.00	3.7	221	4	0.00
R04.7	10/60/10	219.63	292	186	0.00	65.99	292	22	0.00
R04.8	10/60/10	94.51	276	82	0.00	52.64	276	20	0.00
R04.9	10/60/10	74.76	266	53	0.00	18.55	266	8	0.00
R05.1	10/60/25	31.37	623	24	0.00	22.59	627	3	0.00
R05.2	10/60/25	47.61	882	19	0.00	33.31	872	6	0.00
R05.3	10/60/25	434.95	1151	77	0.00	105.8	1404	10	0.00
R05.4	10/60/25	39.6	524	38	0.00	21.89	524	6	0.00
R05.5	10/60/25	2742.05	904	186	0.00	526.07	759	23	0.00
R05.6	10/60/25	–	5566	37	9.13	–	4174	20	1.91
R05.7	10/60/25	17.88	358	20	0.00	29.74	357	6	0.00
R05.8	10/60/25	14.69	254	20	0.00	20.07	249	4	0.00
R05.9	10/60/25	13.69	295	12	0.00	21.1	288	4	0.00

Table 3.1: Results for the Benders algorithm with and without extra-cuts

Inst.		Benders No Extra Cuts				Benders With Extra Cuts			
Nb.	(N/A/K)	Time	LP	IP	gap	Time	LP	IP	gap
R06.1	10/60/50	–	9835	48	0.26	2413.49	9826	11	0.00
R06.2	10/60/50	–	13737	4	10.49	–	13700	1	5.71
R06.3	10/60/50	–	15199	0	9.96[a]	–	13576	1	7.77
R06.4	10/60/50	–	2587	149	1.28	–	2616	27	0.43
R06.5	10/60/50	–	6142	14	7.49	–	6001	5	5.07
R06.6	10/60/50	–	7186	5	8.75	–	6967	3	6.59
R06.7	10/60/50	23.04	238	7	0.00	32.1	238	2	0.00
R06.8	10/60/50	15.05	174	6	0.00	22.41	174	2	0.00
R06.9	10/60/50	1127.89	895	114	0.00	306.23	895	24	0.00
R07.1	10/82/10	5.26	420	2	0.00	5.66	420	0	0.00
R07.2	10/82/10	4.94	369	3	0.00	5.32	369	2	0.00
R07.3	10/82/10	6.96	489	3	0.00	7.44	489	1	0.00
R07.4	10/82/10	13.55	309	31	0.00	6.25	309	7	0.00
R07.5	10/82/10	89.63	354	78	0.00	27.35	359	13	0.00
R07.6	10/82/10	894.15	429	146	0.00	101.9	429	16	0.00
R07.7	10/82/10	–	232	336	1.33	1937.8	232	34	0.00
R07.8	10/82/10	–	448	124	5.38	3558.97	448	38	0.00
R07.9	10/82/10	–	290	175	2.35	–	290	35	0.11

[a] On the relaxed model

Table 3.2: Results for the Benders algorithm with and without extra-cuts (continued)

3.5 Conclusions

Although there was good indication that Benders decomposition was a good choice of exact algorithm for the MCFNDP, our results showed the contrary. Several extensions were added to the standard Benders algorithm framework, in particular, an original contribution presenting procedures in order to generate extra cuts at each iteration and therefore reduce the number of total iterations. These procedures are easily integrated within an implementation that uses the ideas of McDaniel and Devine (1977), which have proposed the resolution of the relaxed problem in order to reduce the number of (computationally expensive) integer iterations and therefore reduce the overall solution time.

Part II

Steiner tree problems with profits

Chapter 4

A survey of Steiner tree problems with profits

Abstract

This is a survey of the Steiner tree problem with profits, a variation of the classical Steiner problem where, besides the costs associated with edges, there are also revenues associated with vertices. The relationships between these costs and revenues are taken into consideration when deciding which vertices should be spanned by the solution tree. The survey contains a classification of the problems falling within this category and an overview of the methods developed to solve them. It also lists several graph preprocessing procedures and analyzes their validity for the different variants of the problem. Finally, a brief comparison is made between the profit versions of the Steiner tree problem and of the travelling salesman problem.

Keywords: Prize collecting, network design, Steiner tree problem, quota, budget, reduction tests.

4.1 Introduction

The purpose of this chapter is to present a survey of models and algorithms for the family of *Steiner Tree Problems with Profits* (STPP), a variant of the classical *Steiner Tree Problem* (STP). Both problems are defined in the following. Let $G = (V, E)$ be a graph with vertex set $V = \{1, \ldots, n\}$ and edge set $E = \{e = (i, j) : i, j \in V, i < j\}$, where each edge $e \in E$ has an associated cost c_e. The problem of determining a minimum cost network

spanning all vertices V of G is known as the *Minimum Spanning Tree Problem* (MSTP). The STP is a very similar problem arising when some vertices need not be spanned, but may be used if their inclusion reduces the solution cost (see, for instance, Hwang et al., 1992). Unlike the MSTP, the STP is NP-hard. In the STPP, in addition to the costs associated with the edges, there are also revenues r_i, associated with the vertices i of the graph. The goal is to determine a subtree minimizing cost or maximizing revenue (or profit), subject to constraints. The exact criteria guiding the optimization vary for the different versions of the STPP. In some particular problems, both cost and revenue are combined in the objective function, while in others, limits on either the cost or the revenue will appear as constraints. Therefore, four basic criteria can be used to distinguish between the variants: the cost of edges, the revenue of vertices, the minimum collected revenue (quota), and the maximum allowed cost (budget). Depending on the problem, some of these criteria can be ignored or combined.

Our aim is to review the main applications of the STPP as well as the models and algorithms proposed for its resolution. The remainder of this chapter is organized as follows. Applications are described in Section 4.2, followed, in Section 4.3, by a more detailed description of the different problems and by an overview of the existing methods. An important component of modern algorithms is the application of reduction tests to eliminate some vertices and edges that are absent from at least one optimal solution. In Section 4.4, these preprocessing tests are reviewed and their validity is analyzed for the different variants of the STPP. Then, in Section 4.5, a brief comparison between the STPP and the travelling salesman problem with profits is reported. The chapter ends with some conclusions in Section 4.6.

4.2 Applications

Several network design problems can be appropriately modeled as an STPP. One traditional application is the design of telecommunications local access networks. Here, the goal is to create or expand a local access network to offer service to new customers. Each new consumer represents a potential revenue for the company but there also exists a connection cost associated with the network to be constructed. The problem is clearly modeled in a graph, where customers are represented by vertices and physical links between them are represented by edges. The cost associated with an edge is the cost of laying down the optical fiber, while the revenue of a vertex is derived from customers. Ljubić *et al.* (2005) have

reported a similar application in the planning of heating networks, where customers have an estimated heat demand and the street network provides the underlying graph where pipes can be laid down.

The STPP may also appear as a subproblem of more general problems. Engevall et al. (1998) have solved an STPP as a subproblem within a constraint generation algorithm for the problem of cost allocation in minimum 1-trees (in a graph G, a 1-tree is a spanning tree on $G = (V\backslash\{1\}, E)$ connected to vertex 1 by two edges). Chawla *et al.* (2003) have used an STPP as part of a mechanism for the extended multicasting game in which vertices and edges are selfish agents trying to maximize their own profit. This mechanism obtains a fraction of the market profit or demonstrates that no profitable solution exists. Finally, Friedman and Parkes (2003) have suggested the use of an STPP as part of an online mechanism for the dynamic pricing of WiFi service in an internet cafe. The idea is to recompute the optimal STPP tree at each period, based on the current clients, and to maximize the profit equal to the revenues on the connections, minus the cost of maintaining the links.

4.3 Classification of the problems

Several variants of the STPP can be derived depending on how the costs and revenues are considered in the optimization problem. In the sequel, we present the most important variants.

4.3.1 Node-weighted and prize-collecting Steiner tree problems

In the node-weighted and prize-collecting STP, both costs and revenues are present in the objective function. The idea is not to obtain a non-dominated solution, as would be the case in a multi-objective optimization, but to optimize a linear combination of cost and revenue.

Several mathematical formulations have been developed for this case. The following formulation makes use of a generalization of the classical subtour elimination constraints (SEC) introduced by Dantzig et al. (1954) for the *Travelling Salesman Problem* (TSP). Let x_{ij} be real-valued variables associated with edges $(i, j) \in E$ and y_i be binary variables associated with vertices $i \in V$. Variable y_i is equal to 1 if vertex i belongs to the solution, and 0 otherwise. For $S \subseteq V$, define $E(S)$ as the set of edges with both endpoints in S. Finally, let T be the set of vertices that must be present in a solution. These vertices are

called *mandatory* or *terminal* vertices. With these definitions, the problem can be written as:

$$\text{Minimize} \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{i\in V} r_i(1-y_i) \tag{4.1}$$

subject to

$$\sum_{(i,j)\in E} x_{ij} = \sum_{i\in V} y_i - 1, \tag{4.2}$$

$$\sum_{(i,j)\in E(S)} x_{ij} \leq \sum_{i\in S\setminus\{k\}} y_i, \qquad k \in S \subseteq V,\ |S| \geq 2, \tag{4.3}$$

$$y_i = 1, \qquad i \in T, \tag{4.4}$$

$$0 \leq x_{ij} \leq 1, \qquad (i,j) \in E, \tag{4.5}$$

$$y_i \in \{0,1\}, \qquad i \in V. \tag{4.6}$$

The objective function minimizes the cost of the opened edges and the lost revenue of the unreached vertices. Constraint (4.2) forces the presence of $s-1$ edges, where s is the number of spanned vertices. Constraints (4.3) are the generalized subtour elimination constraints. Note that for subsets S formed only by spanned vertices, constraints (4.3) reduce to the classical SEC. Constraints (4.4) force the presence of the vertices of T in the solution.

Model (4.1)–(4.6) was used by Lucena and Resende (2004) (without constraints (4.4)) and follows from an extended formulation for the Steiner tree problem proposed by Lucena (1991), Goemans (1994) and Margot et al. (1994). It is interesting to observe that the x variables associated with edges need not to be declared as integer. Indeed, once the y variables are equal to 0 or 1, constraints (4.2) and (4.3) define the convex hull of the characteristic vectors of the spanning trees on the subgraph of G formed by the selected vertices (see Margot et al., 1994).

Goemans (1994) has observed that in the presence of mandatory vertices, some redundant generalized subtour elimination constraints can be eliminated. Let $N = V\setminus T$ be the set of non-mandatory vertices. Constraints (4.3) then reduce to

$$\sum_{(i,j)\in E(S)} x_{ij} \leq \sum_{i\in S\cap N} y_i + |S\cap T| - 1, \qquad S \subseteq V, S\cap T \neq \emptyset, |S| \geq 2 \tag{4.7}$$

and

$$\sum_{(i,j)\in E(S)} x_{ij} \leq \sum_{i\in S\setminus\{k\}} y_i, \qquad k \in S \subseteq N, |S| \geq 2. \tag{4.8}$$

The objective function (4.1) can be equivalently written as:

$$\text{Maximize} \sum_{i \in V} r_i y_i - \sum_{(i,j) \in E} c_{ij} x_{ij}, \tag{4.9}$$

i.e., the goal is to maximize the profit, equal to the revenue of the spanned vertices minus the cost of the selected edges. When the objective function (4.1) is used, the problem is sometimes referred to as the Goemans-Williamson minimization problem (Goemans and Williamson, 1995). When the "real profit" of objective function (4.9) is used, the problem is known as the Net-Worth maximization problem. While these are equivalent from an optimization point of view, they are not equivalent for the computation of the worst case performance ratio of approximation algorithms. Later in this section, we present some approximation algorithms for the Goemans-Williamson problem. However, no such algorithm exists for the Net-Worth version. Feigenbaum et al. (2001) have indeed proved that it is NP-hard to derive an approximation to a constant factor for this problem.

Both the Goemans-Williamson and the Net-Worth problems are known under the more general names of *Node-Weighted Steiner Tree Problem* (NWSTP) and *Prize-Collecting Steiner Tree Problem* (PCSTP). The difference between the NWSTP and the PCSTP is that in the second case, the set of mandatory vertices T is empty. In practice, these mandatory vertices often correspond to facilities (like a telecommunications central) that must be present in the network in order to supply the service.

The NWSTP was first proposed by Segev (1987) who developed three simple heuristic solution methods for a special case of the problem in which a single vertex is mandatory. This author also described two different Lagrangean relaxation bounding procedures. Both the heuristics and the bounding procedures were tested in complete graphs of up to 40 vertices.

Bienstock *et al.* (1993) were the first to propose a solution method for the PCSTP. They worked primarily with the *TSP with Profits* (TSPP), a version of the TSP in which a vertex has a revenue but need not be visited (see Section 4.5). They developed a heuristic based on Christofides's heuristic for the TSP (Christofides, 1976). This method was later adapted to the PCSTP (with the Goemans and Williamson objective function), yielding a heuristic with a worst-case performance ratio of 3. Goemans and Williamson (1995) have later improved this worst-case performance with a $(2 - 1/(|V| - 1))$-approximation algorithm. The authors deal with the rooted version of the problem (where one vertex is a root and must be present in the final tree, as first proposed by Segev). Their procedure

is an adaptation of a more general method which can be applied to a large number of graph optimization problems called *Constrained forest problems* (including the shortest path problem and the *Generalized Steiner Tree Problem*, a variant of the STP in which clusters of vertices are defined and the tree must include at least one vertex from each cluster). The method maintains a set of components. Initially, each vertex is a component. Each component has an associated surplus (initially, the vertex revenue) and is declared active if its surplus is positive. The edges have associated deficits, initially equal to their costs. The deficits and surpluses are dynamically reduced until the deficit of an edge connecting two active components reaches zero, in which case the two components are merged together, or until an active component becomes inactive. The process ends when no active components are left (see Johnson et al., 2000). As suggested by the authors, the method can be viewed as a greedy algorithm in which, at each iteration, a minimum reduced cost edge is selected. Through the execution, the reduced costs are updated, giving rise to a primal-dual structure for the algorithm. In a final phase, unnecessary edges are pruned.

The Goemans and Williamson algorithm was recently revisited by Cole *et al.* (2001) who proposed a faster implementation. Their algorithm runs in $O(k(|V| + |E|)\log^2|V|)$ time in comparison to $O(|V|^2 \log|V|)$ for the original algorithm, at the expense of an additive degradation of $|V|^{-k}$ in the approximation factor, for any constant k. Johnson et al. (2000) have worked specifically with the STPP and proposed a new strategy for the pruning phase. The authors have compared the results in graphs of up to 25,600 vertices and their tests have shown a slightly better performance of the new pruning strategy, which has the additional advantage of being conceptually much simpler. Johnson et al. have also proposed modifications to the core of the algorithm which leads to a $(2 - 1/n)$-approximation for the unrooted problem (PCSTP). The authors also extended the methodology to the quota and budget versions of the STPP, as will be seen in Sections 4.3.2 and 4.3.3.

The most recent works on the PCSTP and NWSTP have explored two main streams of research: a) the obtention of good and reasonably fast methods for real-size problems and b) the development of exact algorithms and lower bounding procedures. Regarding the first stream, the tendency has been to use modern heuristic methods. A good example is the work by Canuto et al. (2001) who have developed and tested a multi-start local search method on instances with up to 1000 vertices and 25,000 edges. The different restarts are conducted in a GRASP fashion: the initial solutions are given by the Goemans and Williamson algorithm feeded with perturbed vertex revenues. The final solution of each

restart goes through a path-relinking procedure, with one of the solutions maintained in a *pool* of the best solutions found so far. Finally, in the last phase of the algorithm, the best known solution is used within a variable neighborhood search procedure (Mladenović and Hansen, 1997). Optimal solutions were obtained for most of the tested graphs.

Klau *et al.* (2004) have reached similar results but with less computational time. Their approach uses a memetic algorithm which works with a population of trees. Each tree goes through an exact algorithm to eliminate non-optimal branches. The entire procedure is followed by an exact method. The idea is to use the memetic algorithm to eliminate edges that are probably not present in the optimal solution and then apply the exact method, based on a separation procedure, on the reduced instance.

Cunha *et al.* (2003) formulate the PCSTP as a restricted minimum forest problem. The authors then use a Lagrangean relaxation which dualizes some constraints only when they first become violated. A constraint is dropped from the objective function when the associated multiplier is zero. A heuristic procedure based on the work of Johnson et al. (2000) is used to obtain upper bounds. Results obtained on instances of up to 400 vertices and 1507 edges are rather modest, an outcome that the authors attribute to probable cycling in the procedure.

Besides heuristics, a fair amount of research has been conducted on exact algorithms and lower bounding procedures. Engevall et al. (1998) have proposed a Lagrangean-based lower bounding procedure using a formulation based on a graph transformation which enables the NWSTP to be treated as an MSTP. The resulting bounds were stronger than those obtained by Segev (1987).

Lucena and Resende (2004) have also developed a new lower bounding method. The authors have used formulation (4.1)-(4.6), and proposed a cutting-plane algorithm based on a separation procedure that identifies violated subtour elimination constraints. The separation procedure has been refined to introduce the idea of orthogonal cuts (cuts generated from subtour elimination constraints with no common vertices). A total of 114 instances with up to 1000 vertices and 25,000 edges were solved and the bounds were proved to be optimal in 99 of these.

Recently, Ljubić *et al.* (2005) have improved these results with a branch-and-cut algorithm, applied to a formulation that uses connectivity inequalities. These inequalities are generated efficiently via a separation procedure based on a maximum flow algorithm. The authors have solved instances of up to 2500 vertices and 62,500 edges, including the Lucena and Resende instances, for which similar results were obtained with a decrease of

two orders of magnitude in the computation time.

4.3.2 The quota STPP

In the quota version of the STPP the goal is to minimize the costs while guaranteing a minimum value for the collected revenues:

$$\text{Minimize} \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{4.10}$$

$$\text{subject to (4.2)–(4.6) and}$$

$$\sum_{i\in V} r_i y_i \geq Q, \tag{4.11}$$

where Q is the quota, i.e., the minimum acceptable collected revenue.

The quota version of the STPP has received almost no attention in the literature. To our knowledge, this problem is mentioned in only one article: Johnson et al. (2000) have observed that the quota STPP is a generalization of the k-MSTP, in which one tries to determine a minimum cost spanning tree containing at least k vertices (note that if the revenues of all vertices are set to 1 and the quota is set to k, then the PCSTP becomes a k-MSTP). The authors use this fact to obtain a 2.5-approximation algorithm for the resulting quota STPP, based on an algorithm for the unrooted k-MSTP (Arya and Ramesh, 1998). The resulting procedure is, however, not likely to be efficient in practice.

A more practical algorithm has also been developed. The idea is to multiply the vertex revenues by a factor α and then use the Goemans and Williamson algorithm on the modified instance. The larger the value of α, the more revenues should be collected (since these will easily counterbalance the costs). The value of α is increased until a solution with the minimum quota has been reached. If the total collected revenue exceeds the quota, then one can try to reduce α and still find a total revenue that will satisfy (4.11). The authors use binary search to find the minimum value α yielding an acceptable quota. By varying α, they also construct tradeoff curves between the total cost and total revenue.

4.3.3 The budget STPP

The budget STPP is the counterpart of the quota problem. Here, one wants to maximize the total collected revenues under the restriction that the total cost is limited by a budget:

$$\text{Maximize} \sum_{i \in V} r_i y_i \tag{4.12}$$

subject to (4.2)–(4.6) and

$$\sum_{(i,j) \in E} c_{ij} x_{ij} \leq B, \tag{4.13}$$

where B is the allowed budget.

Almost no research has been devoted to the budget STPP. Johnson et al. (2000) have proved that by repeatedly running a 3-approximation algorithm for the quota problem, one can find a $(5 + \epsilon)$-approximation algorithm for the unrooted budget STPP, where $\epsilon > 0$. Again, this approximation algorithm is not likely to be useful in practice. For a more efficient procedure, the authors propose applying the same strategy as for the quota STPP, i.e., multiplying the costs by a factor α (this time, in order to try to find a solution that satisfies the budget constraint) and applying a binary search procedure on the values of α.

4.3.4 The fractional STPP

A non-linear version of the STPP maximizes the revenue-to-cost ratio. In this version of the problem, the value of a solution associated with a solution tree $T = (V_T, E_T)$, where $V_T \subseteq V$ is the set of selected vertices and $E_T \subseteq E$ is the set of selected edges, is given by

$$v(T) = \frac{\sum_{v \in V_T} r_v}{c_0 + \sum_{e \in E_T} c_e}, \tag{4.14}$$

where c_0 is a fixed cost, and the goal is to determine a tree T maximizing $v(T)$. The fractional STPP enables the consideration of a fixed cost for the construction of the network.

Klau *et al.* (2003) have dealt with a variation of this problem where the underlying graph is already a tree. The goal is, therefore, to identify an optimal subtree containing the root. The authors have developed three algorithms based on a parametric formulation: a binary search, a Newton method, and an algorithm based on Megiddo's parametric search (Megiddo, 1979). Tests were conducted on trees of up to 10,000 vertices and showed that the Newton method had a better performance. One could probably consider adding budget or quota constraints to this problem. However, to our knowledge, not even the basic version of the fractional STPP on graphs has been the subject of any research.

4.4 Reduction tests

When solving the Steiner tree problem and its variants, several authors make use of reduction tests. These tests attempt to find vertices and edges that are absent from at least one optimal solution, or to find sets of vertices and edges that can be treated as a single vertex. Preprocessing procedures often iterate between one or more different tests until no more reductions can be made.

The field of graph preprocessing for Steiner tree problems is rooted in the work of Beasley (1984) and Balakrishnan and Patel (1987). A classical set of tests was established by Duin and Volgenant (1989). With the appearance of the profit versions of the problem, some of these tests were promptly adapted to the new problems (Duin and Volgenant, 1987) and combinations of these started being used as a preprocessing phase in many solution procedures. These preprocessing phases have become an important part of the more recent algorithms. Since they can significantly reduce the size of the graph that must effectively be solved, they enable the resolution of larger instances. Ljubić *et al.* (2005) have studied the efficiency of a few simple reduction tests. In about 100 seconds of computational time, their largest instances of 2500 vertices and 62,500 edges could be slightly reduced in terms of the number vertices, while the number of their edges was more than halved. In one extreme case, the final number of vertices and edges after preprocessing were 861 and 3881, a reduction of 65% and 94%, respectively.

4.4.1 Description of the reduction tests

As part of this survey on the STPP, we present an overview of the most common reduction tests. These tests are adapted from the classical STP to the PCSTP and to the NWSTP. Their validity is not always clear for other variants of the STPP, like the quota or budget versions, and is therefore analyzed in this section. For all tests presented in the sequel, we assume all costs c_{ij} and revenues r_i to be non-negative. Moreover, note that one can eliminate any edge (i, j) with cost $c_{ij} = 0$ by merging vertices i and j into a single vertex.

Test 1 (Degree one test)*: (a) A vertex i that is only adjacent to one vertex j can be eliminated if $r_i < c_{ij}$. (b) If $r_i > c_{ij}$ and $r_j > c_{ij}$, vertices i and j are merged into a single vertex v of revenue $r_v = r_j + r_i - c_{ij}$.*

Test 2 (Degree two test)*: A vertex i that is only adjacent to vertices j and k cannot have degree 1 in the optimal solution if $r_i < min\{c_{ij}, c_{ik}\}$. In this case, the vertex can be*

eliminated and an edge (j,k) *of cost* $c_{jk} = c_{ij} + c_{ik} - r_i$ *is created or, if edge* (j,k) *exists, its cost is redefined as* $min\{c_{jk}, c_{ij} + c_{ik} - r_i\}$.

Test 3 (General degree test)*: A generalization of tests 1 and 2 can be applied to vertices with degree superior to two. Consider a vertex* i *with* $r_i = 0$ *and let* $Ad(i) = \{v_1, v_2, \ldots, v_n\}$ *be the set of vertices adjacent to it. For a subset of vertices* $V' \subseteq V$*, let MST(V')* *be the cost of the minimum spanning tree problem in the complete graph formed by the vertices in V', where the cost of each edge* (v_i, v_j) *is given by the shortest path distance between* v_i *and* v_j *in* G*. We know that if*

$$MST(V') \leq \sum_{v \in V'} c_{iv}, \qquad \forall\, V' \subseteq Ad(i), \quad |V'| \geq 3, \tag{4.15}$$

then the degree of i *in the optimal PCSTP solution must be zero or two. Therefore, vertex* i *can be eliminated and each pair of edges* $(v_i, i), (i, v_j)$ *can be replaced by an edge* (v_i, v_j) *of cost* $c_{v_i v_j} = c_{v_i i} + c_{i v_j}$ *or, if this edge already exists, of cost* $c_{v_i v_j} = min\{c_{v_i v_j}, c_{v_i i} + c_{i v_j}\}$.

Test 4 (Minimum adjacency or V\K reduction test)*: Adjacent vertices* i *and* j *can be merged into a single vertex* v *if* c_{ij} *is the smallest cost of all edges incident to vertex* i *(or* j*), and* c_{ij} *is smaller than each revenue* r_i *and* r_j*. The revenue* r_v *of the merged vertex is equal to* $r_i + r_j - c_{ij}$.

Test 5 (Shortest-path or least-cost test)*: An edge* (i,j) *can be eliminated if there exists a shortest path between* i *and* j *(considering the costs* c_{ij}*) not containing* (i,j).

Test 6 (Nearest vertex test)*: In the NWSTP, let* T *be the set of mandatory vertices and* d_{ij} *be the shortest path distance between vertices* i *and* j*, considering the edge costs. Suppose* $k \in T$ *and* (i,k) *is a minimum-weighted edge incident to* k*, i.e.,* $c_{ik} = min\{c_{jk} | j \in V, j \neq k\}$*. If a vertex* $l \in T \backslash \{k\}$ *exists with*

$$d_{li} + c_{ik} \leq min\{c_{jk} | j \in V, j \neq i, k\},$$

then edge (i,k) *is part of an optimal solution and vertices* k *and* i *can be merged.*

Recently some new tests have been proposed by Uchoa (2006) who has revisited the tests for the classical Steiner problem and, in particular, two tests using what Duin and Volgenant have called *bottleneck distances*. Let $\mathcal{P}(i,j)$ be the set of all paths connecting vertices i and j. The bottleneck distance between two vertices i and j is defined as

$$B(i,j) = \min\{SD(P) | P \in \mathcal{P}(i,j)\}, \tag{4.16}$$

where $SD(P)$ is the *Steiner distance*, i.e., the maximal distance between two terminal vertices in path P. Uchoa proposes a counterpart of the bottleneck distances for problems with revenues. Consider $P \in \mathcal{P}(i,j)$, and let $e(P)$ be the set of edges $(k,\ell) \in E$ such that k and ℓ are consecutive in P, and $v(P)$ be the set of vertices $v \in V$ appearing in P. Also let S_{xy} be the subpath of P between x and y, with $x, y \in v(P)$. Then define the Steiner distance associated with this subpath as

$$SD(S_{xy}) = \sum_{e \in e(S_{xy})} c_e - \sum_{i \in v(S_{xy}) \setminus \{x,y\}} r_i, \tag{4.17}$$

and the Steiner distance associated with the whole path as

$$SD(P) = \max_{x,y \in v(P)} SD(S_{xy}). \tag{4.18}$$

This new definition of the Steiner distance can be used in the bottleneck distance formula (4.16), and the tests originally proposed for the Steiner Problem in the article of Duin and Volgenant (1989) become valid for the PCSTP. In these tests, reproduced below, $B(i,j)^{-e}$ is the bottleneck distance without passing through a given edge e, and is defined as:

$$B(i,j)^{-e} = \min\{SD(P) | P \in \mathcal{P}(i,j); e \notin e(P)\}. \tag{4.19}$$

Test 7 (Special distance): *Let $(i,j) \in E$. If $B(i,j)^{-(i,j)} \leq c_{ij}$, then edge (i,j) can be removed.*

Test 8 (Non-terminal degree 3): *Let i be a vertex with $r_i = 0$, adjacent to vertices v, w and z. If*

$$min\{B(v,w) + B(v,z), B(w,v) + B(w,z), B(z,v) + B(z,w)\} \leq c_{iv} + c_{iw} + c_{iz},$$

then there exists an optimal solution in which the degree of i is 0 or 2. Therefore, i and its three adjacent edges can be replaced by the following three edges: (v,w) with cost $c_{iv} + c_{iw}$, (v,z) with cost $c_{iv} + c_{iz}$, and (w,z) with cost $c_{iw} + c_{iz}$.

4.4.2 Validity for the quota and budget STPP

The structure of the objective function for the PCSTP and the NWSTP is quite convenient for the development of reduction tests since it combines additive edge costs and vertex revenues. In the cases of the quota and budget STPP, however, the objective function contains only one of these terms (either revenues or costs) and therefore, all tests based on a cost analysis using both the costs and revenues become invalid. This fact is

even more pronounced in the presence of the additional constraints. Indeed, the quota constraint might force the presence of a vertex that was considered unprofitable for the PCSTP or the NWSTP (since the cost to reach it was larger than its revenue), while the budget constraint might avoid the inclusion of a profitable vertex. Therefore, all tests based on a combination of costs and revenues become invalid for the quota and budget STPP. These include tests 1, 2 and 4. In Figure 4.1, the representative case of degree-one test (test 1 in the list) is analyzed through two simple examples.

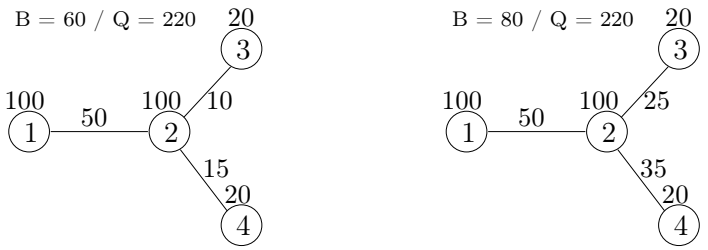


Figure 4.1: Invalidity of test 1 for the budget and quota STPP.

In Figure 4.1, the values on the vertices represent their revenues, while those on the edges represent their costs. Consider the application of test 1(a) to the graph of Figure 4.1a. Vertices 3 and 4 should be merged with vertex 2. However, in the optimal solution of the budget STPP, vertex 2 is present along with vertices 1 and 3, but vertex 4 is not, due to the budget constraint. In the quota STPP, vertex 4 is not present either, since its presence would increase the objective function and the presence of vertices 1, 2 and 3 is sufficient to satisfy the quota constraint. For the case of test 1(b), consider the graph of Figure 4.1b. According to the test, vertices 3 and 4 should be deleted from the graph. However, vertex 3 belongs to the optimal solution of the budget STPP along with vertices 1 and 2 since its addition increases the objective function value and does not violate the budget constraint. For the quota STPP, vertex 3 is also present in the final solution, along with vertices 1 and 2, since it is necessary to satisfy the quota constraint.

Like tests 1, 2 and 4, tests 7 and 8 make use of calculations mixing revenues and edge costs (see the definition of the special distance for one subpath in (4.17)). They are therefore invalid for the quota and budget STPP, since the presence of either budget or quota constraints, or the absence of either costs or revenues in the objective function might invalidate the conclusions. Even for the case of the PCSTP and NWSTP, Uchoa (2006)

reports that the computation of the new bottleneck distances is NP-hard. However, the author uses a heuristic method to calculate these distances and implements a preprocessing package including classical preprocessing tests as well as tests 7 and 8. A comparison of the results with those of Ljubić *et al.* (2005) has revealed that significant extra reductions were obtained with the additional tests, especially for instances containing a large number of mandatory vertices.

4.4.3 Validity for the fractional STPP

Although no author has considered the fractional STPP in general graphs, it is interesting to analyze the validity of the reduction tests for this problem. One can easily conclude that the presence of the quotient in the objective function forbids any local consideration using both the costs and revenues. Therefore, as for the quota and budget STPP, tests 1, 2 and 4 become invalid. Tests 3, 5 and 6 make considerations based on the edge costs, acting only on the denominator of (4.14) and are, therefore, valid. Like tests 1, 2 and 4, tests 7 and 8 are invalid for the fractional STPP since they are based on local considerations using both the costs and the revenues.

4.4.4 New tests for the budget STPP

As a last note on the reduction tests, one can clearly see that the introduction of the quota and budget constraints invalidates many of the tests developed for the NWSTP and for the PCSTP. However, these constraints may also represent an opportunity. Indeed, some new reduction tests specifically adapted to the quota and budget versions of the problem can be derived because of the introduction of the new constraints. The two simple tests presented below for the case of the budget STPP are illustrative examples.

Test 9 (Distance to terminal)*: In the budget STPP with a mandatory vertex set T, a vertex $i \notin T$ can be eliminated if the cost of the shortest path linking i and any vertex in T is larger than B.*

Test 10 (Minimum distance to terminal tree)*: Let T be the set of mandatory vertices and $STP_G(T)$ be the cost of the STP in G. A vertex $i \notin T$ with a single incident edge (i, k) can be eliminated if $STP_G(T) + c_{ik} > B$.*

4.4.5 Summary of the reduction tests

Table 4.1 presents a summary of the reduction tests presented in this section. For each test, it is possible to see its applicability to the different problems, the original reference in which the test is described and the articles where it has been used as part of a preprocessing algorithm.

Test	NWSTP/ PCSTP	Quota STPP	Budget STPP	Fractional STPP	Proposed by	Used in
1	yes	no	no	no	Duin and Volgenant (1987)	Cunha *et al.* (2003), Lucena and Resende (2004) Klau *et al.* (2004), Lujbic *et al.* (2004)
2	yes	no	no	no	Duin and Volgenant (1987)	Cunha *et al.* (2003), Lucena and Resende (2004), Klau *et al.* (2004), Lujbic *et al.* (2004)
3	yes	yes	yes	yes	Duin and Volgenant (1989) [a]	Cunha *et al.* (2003) Lucena and Resende (2004), Klau *et al.* (2004), Lujbic *et al.* (2004)
4	yes	no	no	no	Duin and Volgenant (1987)	Cunha *et al.* (2003), Klau *et al.* (2004), Lujbic *et al.* (2004)
5	yes	yes	yes	yes	Duin and Volgenant (1987)	Cunha *et al.* (2003), Lucena and Resende (2004), Klau *et al.* (2004), Lujbic *et al.* (2004)
6[b]	yes	yes	yes	yes	Duin and Volgenant (1989)	–
7	yes	no	no	no	Uchoa (2006)[c]	–
8	yes	no	no	no	Uchoa (2006)[c]	–
9	no	no	yes[d]	no	this chapter	–
10	no	no	yes[d]	no	this chapter	–

[a] Proposed for the classical STP.
[b] The problem must contain at least two mandatory vertices.
[c] Adapted from Duin and Volgenant (1989).
[d] For the version with mandatory vertices.

Table 4.1: Summary of the reduction tests.

4.5 A comparison with the TSP with profits

The *Travelling Salesman Problem with Profits* (TSPP) is a variation of the classical TSP where revenues are associated with vertices. It has received considerable attention as witnessed by the review articles of Feillet et al. (2006) and of Laporte and Rodríguez-Martín (2006). Several versions of the TSPP have been proposed, as in the case of the STPP. Indeed, the similarities between the two problems are numerous. Concerning the different versions, the TSPP also exists in a variation where both revenues and costs are considered in the objective function (the *Profitable Tour Problem* (PTP)), and there exist variants where limits on either the revenues or the costs are imposed as constraints: these are the *Prize-Collecting Travelling Salesman Problem* (PCTSP) and the *Orienteering Problem* (OP), respectively. Concerning the solution methodologies for the two families of problems, there are also some common features. As seen in Section 4.3, the approximation algorithm of Bienstock et al. (1993) for the PCSTP has been adapted from an algorithm for the PTP. Moreover, the general framework of Goemans and Williamson (1995) has been used to develop approximation algorithms both for the PCSTP and for the PTP.

Due to the closeness of the problems, we found that it could be insightful to present a comparison between the methods proposed both for the STPP and for the TSPP. In Table 4.2, we present a summary of these methods. For the STPP, we use Section 4.3 of this chapter, while for the TSPP, we use the articles surveyed in Feillet et al. (2006) and in Laporte and Rodríguez-Martín (2006).

Table 4.2 shows that the TSPP has received considerably more attention than the STPP. More interesting, and perhaps less justifiable, is the fact that the research efforts for both problems have been distributed differently. While for the TSPP, most research has concentrated on the OP (where the costs are introduced in the constraints of the model), for the STPP almost all research has been devoted to the NWSTP and the PCSTP.

4.6 Conclusions

We have studied the Steiner Tree Problem with Profits, and we have presented an overview of the available algorithms for several variants of this problem. We have also presented an overview of the existing reduction tests to eliminate or agglomerate vertices and edges of the original graph. We have observed that the research efforts to solve the STPP are concentrated on the PCSTP and NWSTP variants. Indeed, a single article mentions the quota and the budget versions of the STPP, and no author seems to have

considered the fractional STPP on general graphs. While this may not come as a surprise for the fractional STPP (since it is hard to associate its objective function with a profit), this is less justifiable for the other problems. In the particular case of the budget STPP, the absence of any study is even more remarkable considering that most of the research conducted on the similar TSPP has been devoted to the OP.

The question that arises is: are the budget variants only important in the context of the TSP and not in the case of the STPP ? There are at least two indications that the answer to this question is negative. First, budget constraints seem to be relevant in practice in several design problems. It is quite natural to conceive that they are also present in the major projets associated with the expansion of telecommunications or heating networks, two important applications of the STPP. Secondly, the PCSTP and NWSTP variants may present optimal solutions with the same value but with very different practical implications. It is easy to construct an example along the lines of Johnson et al. (2000). Consider two different solutions, both with a profit of \$100. The PCSTP and the NWSTP are insensitive to the fact that this solution could have come from a situation where the sum of the costs are \$10 and the sum of the revenues are \$110, or from a network with construction costs of \$10,000 and revenues of \$10,100. The budget STPP as well as the quota STPP are more robust to this kind of situation, since only the revenues (or the costs) are present in the objective function.

Even if the budget STPP also has its own flaws (as seen in the example of Figure 4.1b, where the unprofitable vertex 3 is unnecessarily spanned), this version of the problem seems to deserve more attention than it has received so far in the literature. The existing knowledge on the PCSTP and NWSTP versions can certainly be useful in developing methods that deal with the particularities of this problem. An interesting study from a practical point of view is the analysis of cost *vs* revenue tradeoff curves, as suggested by Johnson et al. (2000).

Concerning the preprocessing phases, Section 4.4 has presented an overview of the most common reduction tests. These tests have been developed for the NWSTP and PCSTP and are in general invalid for the other variants of the problem. Since the preprocessing phase which is based on these reduction tests has become an important part of most algorithms, the need for additional research for the different versions of the STPP is, again, justified. We hope that this survey will encourage such research efforts and prove a helpful source of ideas and references.

Type	Steiner Tree Problem with Profits	Travelling Salesman Problem with Profits
Costs and profits in the objective function	**Node-Weighted STP or Prize-Collecting STP** Segev (1987): Simple heuristics and Lagrangean bounding procedures. Bienstock *et al.* (1993): Heuristic with worth case performance of 3. Goemans and Williamson (1995): $(2 - 1/(\|V\| - 1))$-approximation algorithm. Johnson et al. (2000): $(2 - 1/\|V\|)$-approximation algorithm for the unrooted problem. Cole *et al.* (2001): Faster implementation of the Goemans-Williamson algorithm with a small loss in the approximation. Canuto et al. (2001): Multi-Start with local search and VNS post-optimization phase. Cunha *et al.* (2003): Lagrangean relaxation with dynamically generated cutting planes. Lucena and Resende (2004): Lower bounding procedure based on a cutting planes algorithm. Klau *et al.* (2004): Hybrid Exact-Memetic algorithm. Ljubić *et al.* (2004): Branch-and-cut algorithm based on connectivity inequalities.	**Profitable Tour Problem** Mittenthal and Noon (1992): Insertion/deletion heuristic procedure. Bienstock *et al.* (1993): Heuristic with worth case performance of 2.5. Goemans and Williamson (1995): $(2 - 1/(\|V\| - 1))$-approximation algorithm. Dell'Amico et al. (1998): Branch-and-bound on the problem transformed into an asymmetric TSP.
Costs in the objective function, profits as constraints	**Quota STPP** Johnson et al. (2000): Theoretical 2.5-approximation algorithm and a more efficient heuristic based on the work of Goemans and Williamson (1995).	**Prize-Collecting Travelling Salesman Problem** Fischetti and Toth (1988): Branch-and-bound with additive bounding procedures. Dell'Amico et al. (1998): Heuristic based on Lagrangean relaxation and improvement moves. Awerbuch *et al.* (1998): Heuristic based on an approximation algorithm for the k-MSTP problem.

Table 4.2: Summary and comparison between the methods for the STPP and for the TSPP.

Type	Steiner Tree Problem with Profits	Travelling Salesman Problem with Profits
Profits in the objective function, costs as constraints	**Budget STPP** Johnson et al. (2000): Theoretical 2.5-approximation algorithm and a more efficient heuristic based on the work of Goemans and Williamson (1995).	**Orienteering Travelling Salesman Problem** Tsiligirides (1984): Two approximate algorithms, one based on a former vehicle routing procedure and another based on Monte Carlo techniques. Golden et al. (1987): Heuristic based on the idea of center of gravity. Golden et al. (1988): Heuristic based on the idea of center of gravity improved by a learning measure. Kataoka and Morito (1988): Branch-and-bound algorithm. Keller (1989): Two-phase algorithm consisting of a construction procedure followed by an improvement phase. Laporte and Martello (1990): Branch-and-bound algorithm. Ramesh and Brown (1991): Four-phase algorithm incorporating a tabu list. Ramesh et al. (1992): Lagrangean relaxation based branch-and-bound algorithm. Leifer and Rosenwein (1994): Cutting planes algorithm. Wang et al. (1995): Neural network algorithm. Chao et al. (1996): Construction algorithm followed by an deterministic annealing algorithm. Fischetti et al. (1998): Branch-and-cut algorithm. Gendreau et al. (1998a): Branch-and-cut algorithm. Gendreau et al. (1998b): Tabu search with neighborhoods based on clusters. Tasgetiren and Smith (2000): Genetic algorithm.

Table 4.3: Summary and comparison between the methods for the STPP and for the TSPP (continued).

Chapter 5

Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints

Abstract

The Steiner tree problem with revenues, budget and hop constraints is a variant of the Steiner tree problem where, besides the costs associated with edges, there are also revenues associated with the vertices. Budget and hop constraints impose limits on the total cost of the network and on the number of edges between any vertex and the root, respectively. This chapter introduces and compares several mathematical models for this problem and describes two branch-and-cut algorithms which solve to optimality instances with up to 500 vertices and 625 edges.

Keywords: Prize collecting, network design, Steiner tree problem, budget, branch-and-cut, hop constraints.

5.1 Introduction

Several real-life decision situations can be described as the problem of determining a least cost network spanning all or some of the vertices of a graph. The most widely known cases include the *Steiner Tree Problem* (STP) and the *Minimum Spanning Tree*

Problem (MSTP). These two problems are described as follows. Let $G = (V, E)$ be a graph with vertex set $V = \{1, \ldots, n\}$, where vertex 1 is the *root* vertex, and edge set $E = \{e = (i, j) : i, j \in V, i < j\}$, where each edge $e \in E$ has an associated cost c_e. The set V is partitioned into a set of *terminal* vertices (including the root) and a set of *Steiner* vertices. The STP consists of determining a minimum cost tree spanning all terminal vertices and possibly some Steiner vertices (see, for instance, Courant and Robbins, 1941; Gilbert and Pollak, 1968; and Hwang et al., 1992). The MSTP is a special case of the STP for which all vertices are terminal. Unlike the MSTP which can be solved in polynomial time (see, e.g., Prim, 1957), the STP is NP-hard (Garey et al., 1977).

Steiner Tree Problems with Revenues (STPR) are an important generalization of the classical STP. In the STPR, in addition to the costs associated with the edges, there is also a revenue $r_i \geq 0$ associated with each vertex i. The goal is to determine a cost minimizing or revenue maximizing subtree subject to constraints. This chapter deals with a particular case of the STPR with two main features. We are first interested in the *Steiner Tree Problem with Revenues and Budget* (STPRB), where the goal is to maximize the collected revenue while respecting an upper limit on the total network cost. We also consider hop constraints which limit the number of edges between any vertex in the solution and the root vertex to an upper limit equal to h. We call the resulting problem the *Steiner Tree Problem with Revenues, Budget and Hop Constraints* (STPRBH).

Several authors (see, e.g., Ljubić et al., 2005; Lucena and Resende, 2005) have studied a related version of the STPR where the goal is to maximize the difference between the collected revenues and the edge costs. This problem is often called the *Prize-Collecting Steiner Tree Problem* or the *Steiner Tree Problem with Profits* (STPP). Since, in our case, the goal is to maximize the collected revenue while respecting a limit on the total cost (and not to maximize the difference between the collected revenue and the network cost), we use the term "revenue" in order to avoid confusion. Maximizing revenue and considering the costs only in the constraints instead of maximizing the net profit may be useful, for example, when costs and revenues are measured in incommensurate units (e.g. resource and monetary units).

Our motivation for studying the STPRBH is twofold. Concerning the budget constraint, we have been stimulated both by the lack of studies dealing with these constraints as well as by the practical importance of them. Indeed, we are aware of only two articles that have considered such constrains (Shogan, 1983; Johnson et al., 2000), despite the fact that they are often present in real-life situations, as seen in Chapter 4. Hop con-

straints, in turn, are often used in telecommunications applications. They guarantee that the probability of service failure at a vertex will not exceed a given threshold $1-(1-\pi)^h$, where π is the probability of failure of any link and h is the number of allowed hops. These constraints also restrict the maximum transmission delay in telecommunication networks (Gouveia, 1996). Finally, Voß (1999) mentions a different motivation for considering Steiner trees with hop constraints: when modeling certain lot sizing problems as Steiner tree problems, hop constraints limit the number of periods some goods can be held in stock.

Steiner Tree Problems with Profits have received considerable attention by operations researchers (see Chapter 4). The first studies go back to Segev (1987) who implemented simple greedy heuristics and a Lagrangean lower bounding procedure. Bienstock et al. (1993) have proposed a heuristic with worst case performance ratio of 3, while Goemans and Williamson (1995), Johnson et al. (2000) and Cole et al. (2001) have developed approximation algorithms. More recent research has focused on lower bounding procedures and metaheuristics. Canuto et al. (2001) have proposed a multi-start algorithm with local search and a variable neighborhood search post-optimization phase. Cunha et al. (2003) have developed a Lagrangean relaxation algorithm with dynamically generated cutting planes, while Klau et al. (2004) have proposed a hybrid exact-memetic algorithm where the final population of the evolutionary approach is used to construct a reduced instance that will go through an exact optimization phase. Two recent articles have applied branch-and-cut algorithms to the STPP. Lucena and Resende (2004) have obtained interesting results based on a separation of generalized subtour constraints. These results have later been improved by Ljubić et al. (2005) who have dealt with an alternative formulation and proposed a separation procedure to identify violated connectivity constraints.

Despite the considerable amount of research dedicated to the STPP, very little work has been done on the STPRB. Indeed, we are only aware of the article of Johnson et al. (2000) who proposed an approximation algorithm with limited practical interest. This lack of research is particulary intriguing due to the fact that budgetary constraints are often present in real-life situations. Concerning the hop constraints no author has, to our knowledge, ever considered their inclusion (or the inclusion of any other notion of reliability) in conjunction with the STPP. These constraints have only been considered in more classical contexts such as the MST and STP. Gouveia (1996) and Gouveia and Requejo (2001) have proposed Lagrangean relaxation lower bounding approaches for the MST with hop constraints and were able to solve instances on complete graphs of up to 60 vertices. Gouveia (1999) has

presented several hop-indexed models for the MST and STP with hop constraints, among other problems, while Dahl et al. (2006) have described a general framework for modeling hop constrained MST problems. Finally, Voß (1999) has presented a mathematical formulation and a tabu search for the STP with hop constraints.

We study four mathematical models for the STPRBH and we propose several branch-and-cut algorithms. Our choice of branch-and-cut methods is motivated by the good results obtained by Lucena and Resende (2004) and Ljubić et al. (2005) in the context of the pure STPP. The first algorithm relies on the initial relaxation of connectivity and hop constraints, while the second relaxes a set of linking constraints on disaggregated variables. In both cases, the constraints are dynamically included in the model when found to be violated. Our computational experiments show that the hop constraints make the problem highly difficult and that the choice of the most efficient strategy depends on the maximum number of hops.

The remainder of this chapter is organized as follows. Section 5.2 proposes mathematical models for the STPRBH. In Section 5.3, we present the branch-and-cut algorithms. Section 5.4 reports the computational results and the chapter ends with some conclusions in Section 5.5.

5.2 Formulations

We introduce four formulations for the STPRB. In Section 5.2.1, we adapt the Dantzig-Fulkerson-Johnson (DFJ) subtour elimination constraints and connectivity constraints (Dantzig et al., 1954) to construct an undirected and a directed formulation for the STPRBH, respectively. The resulting formulations contain a limited number of variables but an exponential number of constraints. A different approach is used in Section 5.2.2, where we use a lifted version of the Miller-Tucker-Zemlin (MTZ) constraints (Miller et al., 1960; Desrochers and Laporte, 1991) to eliminate circuits and to limit the number of hops in each path of the solution. Finally, in Section 5.2.3 we model the problem with three-index position variables and we show that this formulation dominates that obtained with the MTZ constraints.

Gouveia (1996) also presents a formulation belonging to a different family and based on multicommodity flow variables for MST problems with hop constraints, which could also be adapted to the STPRBH. A variation of the multicommodity flow formulation has been proposed by Gouveia (1998) and Gouveia and Requejo (2001), inspired by the idea of

variable redefinition. This latter formulation is a combination of an exact solution for the hop-constrained shortest path problem and the original multicommodity flow formulation. In both cases, the authors have proposed the use of Lagrangean relaxation methods to solve the formulations. Computational tests have shown that even their linear relaxations pose a real challenge. For this reason, we do not consider formulations of this type.

5.2.1 Dantzig-Fulkerson-Johnson formulations

Several mathematical formulations have been developed for the STPP and can be adapted to the STPRB. The following model is based on the classical subtour elimination constraints introduced by Dantzig et al. (1954) for the *Travelling Salesman Problem* (TSP). Let x_{ij} and y_i be binary variables associated with edges $(i,j) \in E$ and vertices $i \in V$, respectively. Variable y_i is equal to 1 if vertex i belongs to the solution ($y_1 = 1$), and to 0 otherwise. Likewise, variable x_{ij} is equal to 1 if edge (i,j) belongs to the solution, and to 0 otherwise. For $S \subseteq V$, define $E(S)$ as the set of edges with both end vertices in S. Let also $P = (i_1 = 1, \ldots, i_\ell)$ denote a path originating at the root node and containing ℓ vertices. Finally, define $\mathcal{P}_h$ as the set of paths P in G with length $\ell = h + 2$. Each path $P \in \mathcal{P}_h$ contains $h+1$ edges and thus violates the hop constraint. The STPRBH can then be written as:

Undirected Dantzig-Fulkerson-Johnson (UDFJ) formulation

$$\text{Maximize} \sum_{i \in V} r_i y_i \tag{5.1}$$

subject to

$$\sum_{(i,j) \in E} x_{ij} = \sum_{i \in V} y_i - 1, \tag{5.2}$$

$$\sum_{(i,j) \in E(S)} x_{ij} \leq \sum_{i \in S \setminus \{k\}} y_i, \qquad k \in S \subseteq V,\ |S| \geq 2, \tag{5.3}$$

$$\sum_{(i,j) \in E} c_{ij} x_{ij} \leq b, \tag{5.4}$$

$$\sum_{t=2}^{h+2} x_{i_{t-1}, i_t} \leq h, \qquad P = (i_1 = 1, \ldots, i_{h+2}) \in \mathcal{P}_h, \tag{5.5}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in E, \tag{5.6}$$

$$y_1 = 1, \tag{5.7}$$

$$y_i \in \{0,1\}, i \in V\backslash\{1\}. \tag{5.8}$$

The objective function maximizes the revenue of the spanned vertices. Constraint (5.2) forces the presence of $v-1$ edges connecting the v spanned vertices. Constraints (5.3) are generalized subtour elimination constraints. These are stronger than the classical subtour elimination constraints used in the TSP formulation in which the right-hand side is $|S|-1$. Note that for subsets S formed only by spanned vertices, constraints (5.3) reduce to the classical subtour elimination constraints. Constraint (5.7) forces the presence of the root vertex in the solution. Constraints (5.4) and (5.5) are specific to the STPRBH. Constraint (5.4) is the budget constraint which forces the total network cost not to exceed the budget b, while constraints (5.5) guarantee that there are no more than h hops between the root vertex and any other vertex in the solution. An interesting lifting of constraints (5.5) consists of replacing the right-hand side h for $\sum_{t=2}^{h+1} y_{i_t}$.

A version of model (5.1)–(5.8) with no budget and no hop constraints was used by Lucena and Resende (2004) and follows from an extended formulation for the STP proposed by Lucena (1991), Goemans (1994) and Margot et al. (1994). It is interesting to observe that in the absence of hop constraints, the x variables associated with edges need not be declared as integers. Indeed, when the y variables are equal to 0 or 1, constraints (5.2) and (5.3) define the convex hull of the characteristic vectors of the spanning trees on the subgraph of G induced by the selected vertices (Margot et al., 1994).

Formulation (5.1)–(5.8) is a straightforward model for the STPRBH. An equivalent model can be developed based on constraints which ensure the existence of a path between the root vertex and all other selected vertices. This has been done in the context of the STPP by Ljubić et al. (2005). Consider an arc set A containing two directed arcs (i,j) and (j,i) for each edge $(i,j) \in E$, but no arc entering the root. A directed rooted tree is called a *Steiner arborescence*. Given a directed set of arcs, the STPRBH can be written as the problem of finding a Steiner arborescence rooted at vertex 1 and spanning a subset Y of vertices with maximum revenue $\sum_{i \in Y} r_i$.

Directed Dantzig-Fulkerson-Johnson (DDFJ) formulation

$$\text{Maximize} \sum_{i \in V} r_i y_i \tag{5.9}$$

subject to

$$\sum_{k|(k,i)\in A} x_{ki} = y_i, \qquad i \in V\backslash\{1\}, \tag{5.10}$$

$$\sum_{(i,j)\in A|i\in S,\, j\in V\backslash S} x_{ij} \geq y_k, \qquad S \subset V, 1 \in S, k \in V\backslash S, \tag{5.11}$$

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \leq b, \tag{5.12}$$

$$\sum_{t=2}^{\ell} x_{i_{t-1},i_t} \leq h, \qquad P = (i_1 = 1, \ldots, i_\ell) \in \mathcal{P}_h, \tag{5.13}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in A, \tag{5.14}$$

$$y_1 = 1, \tag{5.15}$$

$$y_i \in \{0,1\}, \qquad i \in V\backslash\{1\}. \tag{5.16}$$

Constraints (5.10) guarantee that if a vertex is selected, it has an indegree of one and vice versa. Constraints (5.11) guarantee that the selected vertices are connected and are therefore called *connectivity constraints*. Note that because constraints (5.11) have a variable right-hand side, connectivity does not mean that the solution will span all vertices, but only that the solution will be connected. Also note that in this case variables y_i could be written in terms of the variables x_{ij}, but they are kept in the model for clarity. Fischetti (1991) has shown that these constraints can be rewritten as a directed version of the subtour elimination constraints (5.3). A number of studies have shown that for several variants of the STP and MSTP, directed models are better than their undirected counterpart (see, e.g., Chopra and Rao, 1994a, 1994b; Feremans et al., 2002; Ljubić et al., 2005). For this reason, we prefer model (5.9)–(5.16) to model (5.1)–(5.8).

5.2.2 Miller-Tucker-Zemlin formulation

A different directed formulation can be obtained by adapting the MTZ constraints. Gouveia (1995b) has used this idea to propose a formulation for the MST with hop constraints, while Khoury et al. (1993) have used the same set of constraints in a pure STP formulation. Later, Voß (1999) has adapted these formulations to the STP with hop

constraints. The basic idea of the MTZ constraints is to introduce potential variables associated with each vertex of the graph and impose that in every path of the solution, the potential variables become larger when the distance from the root vertex increases. Let u_i be a real-valued potential variable associated with vertex i. Again, h is the maximum number of arcs between any vertex and the root vertex. One can model the STPRBH as follows:

Miller-Tucker-Zemlin (MTZ) formulation

$$\text{Maximize} \sum_{i \in V} r_i y_i \tag{5.17}$$

subject to

$$\sum_{k|(k,i)\in A} x_{ki} = y_i, \qquad i \in V\backslash\{1\}, \tag{5.18}$$

$$x_{ij} \leq \sum_{k|(k,i)\in A, k\neq j} x_{ki} \qquad (i,j) \in A, i \neq 1, \tag{5.19}$$

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \leq b, \tag{5.20}$$

$$u_1 = 0, \tag{5.21}$$

$$y_i \leq u_i \leq h y_i, \qquad i \in V\backslash\{1\}, \tag{5.22}$$

$$(h+1)x_{ij} + u_i - u_j + (h-1)x_{ji} \leq h \qquad (i,j) \in A, \tag{5.23}$$

$$y_1 = 1, \tag{5.24}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in A, \tag{5.25}$$

$$y_i \in \{0,1\}, \qquad i \in V\backslash\{1\}. \tag{5.26}$$

Constraints (5.19) and (5.21)–(5.23) guarantee that the solution is connected and cycle-free. Indeed, constraints (5.23) imply that each vertex in a path has an associated potential variable u_i larger than that of its predecessor, which is impossible in a circuit. Constraints (5.21)–(5.22) guarantee that the largest potential in the solution does not exceed h.

Some valid inequalities can be added to the model. For example, for vertices i with zero revenue, constraints (5.22) can be tightened, as

$$y_i \leq u_i \leq (h-1)y_i \qquad i \in V\backslash\{1\}, r_i = 0, \tag{5.27}$$

which are valid since there is always an optimal solution containing no unprofitable vertices as leaves. The right-hand side of these constraints can be further strengthened, as proposed by Voß (1999), by considering the arcs originating at the root node:

$$y_i \leq u_i \leq (h-1)y_i - (h-2)x_{1i} \qquad i \in V\backslash\{1\}, r_i = 0. \tag{5.28}$$

The consideration of the arcs emanating from the root enables the strengthening of constraints (5.22) also for the case of profitable vertices:

$$y_i \leq u_i \leq hy_i - (h-1)x_{1i} \qquad i \in V\backslash\{1\}, r_i > 0. \tag{5.29}$$

Finally, Voß also proposes several strengthenings of constraints (5.23), depending on the type of the vertices involved: terminal vertices or Steiner vertices. We can adapt one of these modifications to the case where the two vertices involved in the constraint are unprofitable. In this case, we know that none of the vertices is a leaf and, therefore, the difference between their potentials is bounded by $h-1$. Constraints (5.23) then become:

$$hx_{ij} + u_i - u_j + (h-2)x_{ji} \leq h-1 \qquad (i,j) \in A, r_i = r_j = 0. \tag{5.30}$$

5.2.3 Garcia-Gouveia Hop formulation

Garcia (1994) and Gouveia (1999) have worked with position variables (also called time-indexed variables) in order to model subtour elimination constraints. The authors mention the possibility of using such variables in situations where a hop limit is imposed. Indeed, let each arc variable x_{ij} be replaced by a set of variables $x_{ij}^p, p = 1, \ldots, h$, where x_{ij}^p is equal to one if arc (i,j) is in position p in the solution, i.e., p hops away from the root, and equal to zero otherwise. In the presence of these disaggregated variables, constraints (5.19) of the MTZ model can be strengthened and rewritten as

$$x_{ij}^p \leq \sum_{k|(k,i)\in A, k\neq j} x_{ki}^{p-1} \qquad (i,j) \in A, p = 2, \ldots, h, \tag{5.31}$$

and the STPRBH can be formulated as:

Garcia-Gouveia Hop (GG-Hop) formulation

$$\text{Maximize} \sum_{i \in V} r_i y_i \tag{5.32}$$

subject to

$$\sum_{k|(k,i)\in A} x_{ki} = y_i, \qquad i \in V\backslash\{1\}, \tag{5.33}$$

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \leq b, \tag{5.34}$$

$$x_{ij}^p \leq \sum_{k|(k,i)\in A, k\neq j} x_{ki}^{p-1} \qquad (i,j) \in A, i \neq 1, p = 2, \ldots, h. \tag{5.35}$$

$$x_{ij} = \sum_{p=1}^{h} x_{ij}^p, \qquad (i,j) \in A, \tag{5.36}$$

$$x_{ij}^1 = 0, \qquad (i,j) \in A, i \neq 1, \tag{5.37}$$

$$x_{1j}^p = 0, \qquad j \in V\backslash\{1\}, p = 2, \ldots, h, \tag{5.38}$$

$$x_{ki}^h = 0, \qquad (k,i) \in A, r_i = 0 \tag{5.39}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{5.40}$$

$$y_1 = 1, \tag{5.41}$$

$$y_i \in \{0,1\}, \qquad i \in V\backslash\{1\}. \tag{5.42}$$

The two-index variables x_{ij} could be rewritten in terms of the three-index variables x_{ij}^p only, but they are kept for clarity. The strengthened constraints (5.35) not only ensure the connectivity of the solution, but also guarantee that the solution is cycle-free and that the hop limit is respected. Note also that constraints (5.37) and (5.38) fix some variables based on the fact that no arc originating at the root may be more than one hop away from the root and, conversely, no intermediate arc may be at exactly one hop from the root. Similarly, (5.39) states that there is always an optimal solution where all leafs are profitable vertices.

In the presence of three-index position variables x_{ij}^p, we propose four new families of valid inequalities:

$$\sum_{p=1}^{h} p x_{ij}^{p} \geq \sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^{p} - h + (h+1)x_{ij}, \qquad (i,j) \in A, \tag{5.43}$$

$$x_{ij}^{p} \geq \sum_{k|(k,i)\in A} x_{ki}^{p-1} + x_{ij} - 1, \qquad (i,j) \in A, i \neq 1, p = 2, \ldots, h, \tag{5.44}$$

$$\sum_{k|(i,k)\in A} x_{ik}^{p} \geq \sum_{k|(k,i)\in A} x_{ki}^{p-1}, \qquad i \in V, i \neq 1, r_i = 0, p = 2, \ldots, h, \tag{5.45}$$

$$x_{ij}^{p} \leq 1 - \sum_{k|(k,i)\in A} \sum_{t=p}^{h} x_{ki}^{t} - \sum_{t=1}^{p-1} x_{ij}^{t} \qquad (i,j) \in A, i \neq 1, p = 2, \ldots, h. \tag{5.46}$$

Inequalities (5.43) are a linearization of the constraints

$$\sum_{p=1}^{h} p x_{ij}^{p} \geq (\sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^{p} + 1)x_{ij}, \qquad (i,j) \in A, \tag{5.47}$$

which indicate that the hop associated with a given arc depends on the number of arcs in the path between the root and its origin vertex (given by $\sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^{p}$). In other words, any arc (i,j) leaving vertex i is one more hop away from the root than the arc entering vertex i. Inequalities (5.44) mean that if arc (i,j) is in the solution, its position should exceed that of its predecessor by one. Inequalities (5.45) state that no unprofitable vertex i is a leaf in the solution. These are a three-index version of the valid inequalities

$$\sum_{k|(i,k)\in A} x_{ik} \geq \sum_{k|(k,i)\in A} x_{ki}, \qquad i \in V, i \neq 1, r_i = 0. \tag{5.48}$$

Finally, inequalities (5.46) state that an arc cannot be p hops away from the root if its origin vertex has any incoming arc which is at least p hops away from the root or if another three-index variable associated with the arc is already set to one.

We will now prove that the GG-Hop formulation implies the MTZ formulation. This is done in the following propositions, which show that the MTZ model constraints (5.19), (5.22) and (5.23) are redundant for the GG-Hop model in the presence of the linking constraints

$$u_i = \sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^{p}. \qquad i \in V \backslash \{1\}. \tag{5.49}$$

Note that these constraints only define the u_i variables and do not strengthen the original GG-Hop formulation.

Proposition 5.1 *Constraints (5.19) are redundant for the GG-Hop formulation.*

Proof: For any arc $(i,j) \in A, i \neq 1$, summing up the $h-1$ constraints (5.35) associated with the arc yields:

$$\sum_{p=2}^{h} x_{ij}^{p} \leq \sum_{p=2}^{h} \sum_{k|(k,i)\in A, k\neq j} x_{ki}^{p-1}.$$

Since $i \neq 1$, it follows that $x_{ij}^{1} = 0$ and therefore, $\sum_{p=2}^{h} x_{ij}^{p} = \sum_{p=1}^{h} x_{ij}^{p} = x_{ij}$. This implies

$$x_{ij} \leq \sum_{p=2}^{h} \sum_{k|(k,i)\in A, k\neq j} x_{ki}^{p-1} \leq \sum_{k|(k,i)\in A, k\neq j} x_{ki}.$$

□

Proposition 5.2 *Constraints (5.23) are redundant for the GG-Hop formulation.*

Proof: We prove that constraints (5.23) are redundant for the GG-Hop formulation by showing that in the presence of the disaggregated variables and constraints (5.35), the left-hand side of any of these constraints is smaller than h. Consider the original MTZ constraint (5.23) associated with arc (i,j):

$$(h+1)x_{ij} + u_i - u_j + (h-1)x_{ji} \leq h.$$

If vertex i is the root vertex, the result follows easily since $u_i = u_1 = 0$ and $(j,1) \notin A$. Moreover, using (5.36) and (5.49), the left-hand side of (5.23) becomes

$$\begin{aligned} L &= (h+1)\sum_{p=1}^{h} x_{1j}^{p} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A} p x_{kj}^{p} \\ &= (h+1)\sum_{p=1}^{h} x_{1j}^{p} - \sum_{p=1}^{h} p x_{1j}^{p} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq 1} p x_{kj}^{p}, \\ &= \sum_{p=1}^{h} (h+1-p) x_{1j}^{p} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq 1} p x_{kj}^{p} \leq h. \end{aligned}$$

If i is not the root vertex, we again use (5.49) to write

$$u_i = \sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^{p}, \text{ and } u_j = \sum_{p=1}^{h} p x_{ij}^{p} + \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} p x_{kj}^{p}. \tag{5.50}$$

Using (5.36) and (5.50), we can rewrite the left-hand side of (5.23) as

$$\begin{aligned} L &= (h+1)x_{ij} + u_i - u_j + (h-1)x_{ji} \\ &= (h+1)\sum_{p=1}^{h} x_{ij}^{p} + \sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^{p} - \sum_{p=1}^{h} p x_{ij}^{p} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} p x_{kj}^{p} + (h-1)\sum_{p=1}^{h} x_{ji}^{p}. \end{aligned}$$

Defining $a = -\sum_{p=1}^{h}\sum_{k|(k,j)\in A,k\neq i} px_{kj}^p + (h-1)\sum_{p=1}^{h} x_{ji}^p$, we can write

$$L = \sum_{p=1}^{h}(h+1-p)x_{ij}^p + \sum_{p=1}^{h}\sum_{k|(k,i)\in A} px_{ki}^p + a. \tag{5.51}$$

Constraints (5.35) and (5.51) imply

$$\begin{aligned}
L &\leq hx_{ij}^1 + \sum_{p=1}^{h-1}\sum_{k|(k,i)\in A,k\neq j}(h-p)x_{ki}^p + \sum_{p=1}^{h}\sum_{k|(k,i)\in A} px_{ki}^p + a \\
&= hx_{ij}^1 + h\sum_{k|(k,i)\in A} x_{ki}^h + h\sum_{p=1}^{h-1}\sum_{k|(k,i)\in A} x_{ki}^p - \sum_{p=1}^{h-1}(h-p)x_{ji}^p + a \\
&= hx_{ij}^1 + h\sum_{p=1}^{h}\sum_{k|(k,i)\in A} x_{ki}^p - \sum_{p=1}^{h-1}(h-p)x_{ji}^p + a.
\end{aligned}$$

We now show that $-\sum_{p=1}^{h-1}(h-p)x_{ji}^p + a \leq 0$. Again, we use (5.35) to obtain an upper bound:

$$\begin{aligned}
& -\sum_{p=1}^{h-1}(h-p)x_{ji}^p + a \\
= \quad & -\sum_{p=1}^{h-1}(h-p)x_{ji}^p - \sum_{p=1}^{h}\sum_{k|(k,j)\in A,k\neq i} px_{kj}^p + (h-1)\sum_{p=1}^{h} x_{ji}^p, \\
= \quad & \sum_{p=1}^{h}(p-1)x_{ji}^p - \sum_{p=1}^{h}\sum_{k|(k,j)\in A,k\neq i} px_{kj}^p \\
\leq \quad & \sum_{p=1}^{h-1}\sum_{k|(k,j)\in A,k\neq i} px_{kj}^p - \sum_{p=1}^{h}\sum_{k|(k,j)\in A,k\neq i} px_{kj}^p \\
= \quad & -\sum_{k|(k,j)\in A,k\neq i} hx_{kj}^h \leq 0.
\end{aligned}$$

It follows that

$$L \leq hx_{ij}^1 + h\sum_{p=1}^{h}\sum_{k|(k,i)\in A} x_{ki}^p \leq h,$$

which proves our proposition since (5.37) implies $hx_{ij}^1 = 0$ and (5.33) implies

$$h\sum_{p=1}^{h}\sum_{k|(k,i)\in A} x_{ki}^p = h\sum_{k|(k,i)\in A}\sum_{p=1}^{h} x_{ki}^p = h\sum_{k|(k,i)\in A} x_{ki} = hy_i \leq h.$$

□

Proposition 5.3 *Constraints (5.28) and (5.29) are redundant for the GG-Hop formulation.*

Proof: Constraints (5.33), (5.36), (5.38), (5.39) and (5.49) imply (5.28) because, for any unprofitable vertex i,

$$\begin{aligned} u_i &= \sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^p = \sum_{p=1}^{h} \sum_{k|(k,i),k\neq 1\in A} p x_{ki}^p + x_{1i}^1 \\ &= \sum_{p=1}^{h} \sum_{k|(k,i)\in A,k\neq 1} p x_{ki}^p + (h-1)x_{1i}^1 - (h-2)x_{1i}^1 \\ &\leq (h-1)\sum_{p=1}^{h} \sum_{k|(k,i)\in A,k\neq 1} x_{ki}^p + (h-1)x_{1i}^1 - (h-2)x_{1i}^1 \\ &= (h-1)\sum_{p=1}^{h} \sum_{k|(k,i)\in A} x_{ki}^p - (h-2)x_{1i} = (h-1)y_i - (h-2)x_{1i} \end{aligned}$$

and,

$$u_i = \sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^p \geq \sum_{p=1}^{h} \sum_{k|(k,i)\in A} x_{ki}^p = \sum_{k|(k,i)\in A} x_{ki} = y_i.$$

Similar arguments can be used to show that (5.29) is also redundant for GG-Hop:

$$\begin{aligned} u_i &= \sum_{p=1}^{h} \sum_{k|(k,i)\in A} p x_{ki}^p = \sum_{p=1}^{h} \sum_{k|(k,i),k\neq 1\in A} p x_{ki}^p + x_{1i}^1 \\ &= \sum_{p=1}^{h} \sum_{k|(k,i)\in A,k\neq 1} p x_{ki}^p + h x_{1i}^1 - (h-1)x_{1i}^1 \\ &\leq h\sum_{p=1}^{h} \sum_{k|(k,i)\in A,k\neq 1} x_{ki}^p + h x_{1i}^1 - (h-1)x_{1i}^1 \\ &= h\sum_{p=1}^{h} \sum_{k|(k,i)\in A} x_{ki}^p - (h-1)x_{1i} = h y_i - (h-1)x_{1i}. \end{aligned}$$

□

Proposition 5.4 *The GG-Hop model dominates the MTZ model.*

Proof: The result follows from Propositions 5.1, 5.2 and 5.3 and from the fact that constraints (5.28) and (5.29) are liftings of (5.22). □

Finally, we can use similar arguments to prove that the MTZ model with the lifted constraints (5.30) is also implied by the GG-Hop formulation, as shown by the following proposition.

Proposition 5.5 *Constraints (5.30) are redundant for the GG-Hop formulation.*

Proof: The arguments are very similar to those used to prove Proposition 5.2. Here, we also use (5.39), which states that for any unprofitable vertex i, $x^h_{ik} = 0, (i,k) \in A$. We prove that the left-hand side of (5.30) associated with an arc (i,j) cannot exceed $h-1$ in the presence of the three-index variables x^p_{ij} and the associated constraints. Again, we consider two cases. If i is the root vertex, then the left-hand side L of (5.30) can be written as

$$\begin{aligned} L &= hx_{1j} - u_j \\ &= h\sum_{p=1}^{h} x^p_{1j} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq 1} px^p_{kj} - \sum_{p=1}^{h} px^p_{1j} \\ &= \sum_{p=1}^{h}(h-p)x^p_{1j} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq 1} px^p_{kj} \\ &\leq (h-p)x_{1j} \leq (h-1). \end{aligned}$$

Otherwise, we know that $x^1_{ik} = 0$ and therefore,

$$\begin{aligned} L &= hx_{ij} + u_i - u_j + (h-2)x_{ji} \\ &= h\sum_{p=1}^{h} x^p_{ij} + \sum_{p=1}^{h} \sum_{k|(k,i)\in A, k\neq j} px^p_{ki} + \sum_{p=1}^{h} px^p_{ji} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} px^p_{kj} \\ &\qquad - \sum_{p=1}^{h} px^p_{ij} + (h-2)\sum_{p=1}^{h} x^p_{ji} \\ &= \sum_{p=1}^{h}(h-p)x^p_{ij} + \sum_{p=1}^{h} \sum_{k|(k,i)\in A, k\neq j} px^p_{ki} + \sum_{p=1}^{h}(h+p-2)x^p_{ji} - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} px^p_{kj} \\ &\leq \sum_{p=1}^{h} \sum_{k|(k,i)\in A, k\neq j} (h-p-1)x^p_{ki} + \sum_{p=1}^{h} \sum_{k|(k,i)\in A, k\neq j} px^p_{ki} + \sum_{p=1}^{h}(h+p-2)x^p_{ji} \\ &\qquad - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} px^p_{kj} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{p=1}^{h} \sum_{k|(k,i)\in A, k\neq j} (h-1)x_{ki}^p + \sum_{p=1}^{h}(h+p-2)x_{ji}^p - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} p x_{kj}^p \\
&= \sum_{p=1}^{h} \sum_{k|(k,i)\in A} (h-1)x_{ki}^p + \sum_{p=1}^{h}(p-1)x_{ji}^p - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} p x_{kj}^p \\
&\leq \sum_{p=1}^{h} \sum_{k|(k,i)\in A} (h-1)x_{ki}^p + \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} p x_{kj}^p - \sum_{p=1}^{h} \sum_{k|(k,j)\in A, k\neq i} p x_{kj}^p \\
&= \sum_{p=1}^{h} \sum_{k|(k,i)\in A} (h-1)x_{ki}^p \leq (h-1) \sum_{k|(k,i)\in A} x_{ki} \leq h-1.
\end{aligned}
$$

□

5.3 Branch-and-cut Algorithms

We have presented four mathematical formulations for the STPRBH, based on quite different ideas. Models UDFJ and DDFJ use the classical Dantzig-Fulkerson-Johnson subtour elimination constraints, while MTZ model uses the Miller-Tucker-Zemlin constraints, and GG-Hop is based on three-index position variables. We have shown that model MTZ and its lifted version are dominated by GG-Hop and, therefore, they will not be considered in the tests. Moreover, as we have explained in Section 5.2.1, we concentrate on the directed DDFJ formulation.

Because of the exponential number of constraints of DDFJ, it is impractical to apply commercial solvers to this formulation, even for mid-size instances. We propose a branch-and-cut algorithm containing two separation procedures which generate violated constraints of type (5.11) and (5.13) only when these are found to be violated during the branch-and-cut process. Section 5.3.1 explains the two separation procedures and the overall branch-and-cut algorithms.

The number of constraints in formulations GG-Hop is polynomial and, therefore, one can make use of commercial LP solvers for the problem as long as the instance size is not too large. However, it is possible to relax constraints (5.35) and add them as cuts in the model when they are found to be violated during the branching process. Section 5.3.2 presents this second branch-and-cut approach.

5.3.1 Branch-and-cut for formulation DDFJ

We consider the DDFJ model. The idea of our branch-and-cut algorithm is to initially relax connectivity constraints (5.11) and hop constraints (5.13) and add them as cuts only when they become violated. In order to reinforce the formulation at the root node, we do not relax all of constraints (5.11). Instead, the following constraints are kept:

$$x_{ij} \leq \sum_{k|(k,i)\in A, k\neq j} x_{ki} \qquad (i,j) \in A,\ i \neq 1. \tag{5.52}$$

Note that constraints (5.52) are equivalent to constraints (5.11) associated with sets $S = \{i, j\}$, for which $(i, j) \in A$. Indeed, these constraints can be obtained by adding the term $\sum_{(p\neq i,j)\in A} x_{pj}$ to both sides of (5.52). In addition to these constraints, we also explicitly impose the two-vertex subtour elimination constraints which are implied by (5.10) and (5.11):

$$x_{ij} + x_{ji} \leq 1 \qquad (i,j) \in E. \tag{5.53}$$

For the relaxed constraints, it is necessary to develop appropriate separation procedures. The first separation procedure takes care of the connectivity of the solution. This procedure is adapted from the work of Ljubić et al. (2005). It consists of identifying disconnected vertices by means of a maximum flow algorithm, and of adding the associated violated constraints of type (5.11). The second separation procedure identifies violated hop constraints of type (5.13) by finding paths from the root to any vertex containing more arcs than the allowed hop limit. These two routines are now presented in detail.

Connectivity separation procedure

The connectivity separation procedure exploits the fact that constraints (5.11) imply the connectivity of all selected vertices in the graph, in particular, the connectivity of the root to all other selected vertices. Indeed, given a partition $\{S, V\backslash S\}$ of V and a cut $C = \{(i, j) : i \in S, j \in V\backslash S\}$ in which the root and a given vertex k are on different sides of the cut, i.e., $1 \in S$ and $k \in V\backslash S$, let X_C be the sum of the x variables belonging to the cut: $X_C = \sum_{(i,j)\in C} x_{ij}$. The connectivity constraints guarantee that $X_C \geq y_k$. Now, if one uses the current x solution at a given node of the branching tree as arc capacities, it is possible to use a maximum flow algorithm to find the minimum cut, in terms of X_C, between the root and every other vertex k of the graph. If X_C is smaller than y_k, then a violated connectivity constraint has been identified and can be added to the problem.

In order to identify more violated inequalities at each iteration, Ljubić et al. (2005) have proposed the use of *nested cuts* and *back-cuts*. The nested cuts were introduced by Koch and Martin (1998). Once a violated cut is found, these cuts consist of setting to one the values of the x variables in the cut and of solving the minimum cut problem again, in the hope of finding another cut that does not include all of the previous arcs. The process is repeated until no more cuts can be identified or an upper limit is reached. The back-cuts were introduced by Chopra and Rao (1994a). The idea is simply to reverse the flows and look for a minimum cut with i as origin vertex and the root vertex as destination. In our implementation, as suggested by Ljubić et al. (2005), the nested cuts and the back-cuts are combined, maximizing the number of violated constraints found at each iteration.

Hop constraints separation procedure

The search for violated hop constraints consists of exploring the current tree for paths containing more arcs than the hop limit. We concentrate on paths rooted at vertex 1 and we explore the subgraph induced by the positive y_i and x_{ij} variables. For a path $P = (i_1 = 1, \ldots, i_\ell)$, define $X_P = \sum_{t=2}^{\ell} x_{i_{t-1},i_t}$ as the sum of the x_{ij} variables in the path. The separation procedure gradually extends a path P rooted at 1 until the number of vertices in the path reaches $h + 2$ and the sum of associated variables exceeds h, or this sum is less than the number of vertices minus two. In other words, these conditions are:

1) $\ell = h + 2$ and $X_P > h$,

2) $X_P < \ell - 2$.

In case 1) a violated hop constraint has been found and the cut $X_P \leq h$ is added to the model. In case 2) it is useless to continue exploring the branch, since no violated inequalities will be found. The complete separation procedure is described in **Algorithm 5.1**, where $f[i] = 1$ if vertex i can still been visited in the search ($i = 1, \ldots, n$), and 0 otherwise.

The idea of **Algorithm 5.1** is to sequentially visit the connected vertices in the induced subgraph until one of conditions 1) or 2) is met. Again, if condition 1) is satisfied, a violated cut has been identified and is added to the model. In both cases, the algorithm blocks the access to the last vertex in the path and returns to the previous vertex, from which the search is resumed. The procedure continues until the root vertex is reached, with no other vertices to visit.

Algorithm 5.1 Hop separation procedure

$\ell = 1$; $P = (i_\ell = 1)$; $X_P = 0$; $f = [0, 1, 1, \dots 1]$;
loop
  if there exists $(i_\ell, j) \in A$, with $f[j] = 1$ **then**
    $\ell = \ell + 1$;
    $i_\ell = j$;
    $f[i_\ell] = 0$;
    $X_P = X_P + x_{i_{\ell-1}, i_\ell}$;
    if $\ell = h + 2$ and $X_P > h$ **then**
      Add the cut $X_P \leq h$;
      $f[k] = 1$, for all k such that $(i_\ell, k) \in A, k \neq i_{\ell-1}$;
      $X_P = X_P - x_{i_{\ell-1}, i_\ell}$;
      $\ell = \ell - 1$;
    else
      if $X_P < \ell - 2$ **then**
        $X_P = X_P - x_{i_{\ell-1}, i_\ell}$;
        $\ell = \ell - 1$;
      end if
    end if
  else
    if $\ell = 1$ **then**
      Stop.
    else
      $X_P = X_P - x_{i_{\ell-1}, i_\ell}$;
      $f[k] = 1$, for all k such that $(i_\ell, k) \in A, k \neq i_{\ell-1}$;
      $\ell = \ell - 1$;
    end if
  end if
end loop

5.3.2 Branch-and-cut for formulation GG-Hop

Several algorithms can be used to solve the GG-Hop formulation. The simplest approach is to solve GG-Hop directly with a commercial branch-and-cut solver. Variants of this approach are obtained by relaxing constraints (5.35) at the root of the search tree and

dynamically generating these constraints whenever they are found to be violated. Similarly, valid inequalities (5.11) and (5.13) can be generated in the search tree. We have also tested the possibility of including (5.43)–(5.46) in the GG-Hop formulation but this did not prove to be beneficial and this option was dropped after some preliminary tests.

5.3.3 Summary of the algorithms and branch-and-cut template

We have fully tested the five algorithms summarized in Table 5.1. The general branch-and-cut template is presented in **Algorithm 5.2**.

Algorithm	Basic model	Relaxed constraints	Valid inequalities
$A1$	GG-Hop		
$A2$	GG-Hop		(5.11), (5.13)
$A3$	GG-Hop	(5.35)	
$A4$	GG-Hop	(5.35)	(5.11), (5.13)
$A5$	DDFJ + (5.52), (5.53)	(5.11), (5.13)	

Table 5.1: Summary of the algorithms

5.4 Computational Experiments

The branch-and-cut algorithms just described were implemented within the CPLEX 9.1.3 framework with standard settings and run on an AMD Opteron machine with a 2390MHz CPU and 16Gb RAM, under Linux. In all cases, branching priority was given to the variables associated with edges incident to the root vertex and the branching node selection was performed according to the best bound rule.

5.4.1 Test instances

We have tested the proposed valid inequalities and branch-and-cut algorithms on the sets of Steiner instances B and C obtained from the OR-Library (Beasley, 1990). We have adapted these instances to the STPRBH by using the terminal vertices as profitable vertices with revenue generated randomly according to a discrete uniform distribution over the interval $[1, 100]$. The Steiner vertices were attributed a zero revenue. For each instance of the first group, 12 scenarios were analyzed, obtained from four values for the hop limit, $h = 3, 6, 9$ and 12, and three values for the budget, $b = s, s/5$ and $s/10$,

Algorithm 5.2 Branch-and-cut template

```
while constraints added do
  Solve root node.
  Look for violated relaxed constraints.
  Look for violated valid inequalities.
end while
if solution is integer then
  END.
else
  Start branching.
  while there exist active nodes do
    Branch or change current node.
    while constraints added do
      Solve current node.
      Look for violated relaxed constraints.
      Look for violated valid inequalities.
    end while
  end while
end if
```

where $s = \sum_{(i,j)\in E} c_{ij}$. For the second group of instances, we used $h = 5, 15$ and 25 and $b = s, s/10$ and $s/30$. Since vertex 1 is not always a terminal vertex in the original instances, the root vertex was chosen as the terminal vertex with the smallest index. The instance sizes range from 50 vertices and 63 arcs to 500 vertices and 625 arcs.

It is interesting to observe how the budget and hop constraints interact, making the solutions of different scenarios for the same instance completely different. Instance *Steinb*8 is a good example and is detailed in Table 5.2 which shows, for each scenario, the optimal solution value, the number of profitable vertices reached, and the percentage of the total revenue collected.

The table is linked to Figures 5.1 and 5.2 which depict the solutions. The values inside the vertices are the vertex numbers, while the values next to the arcs are their cost. A vertex is represented by a circle if its revenue is zero and by a square, otherwise.

For the first three scenarios, there is no budgetary limit and, therefore, the hop constraint is the only restriction preventing the solution from reaching all profitable vertices.

Scenario	b	h	Optimal solution value	Profitable vertices reached	% Revenue collected	Figure
1	s	3	85	2	10	5.1(a)
2	s	6	581	13	70	5.1(b)
3	s	9	836	19	100	5.1(c)
4	s	12	836	19	100	5.1(c)
5	$s/5$	3	85	2	10	5.1(a)
6	$s/5$	6	535	11	64	5.1(d)
7	$s/5$	9	761	16	91	5.1(e)
8	$s/5$	12	832	18	99	5.2(a)
9	$s/10$	3	85	2	10	5.1(a)
10	$s/10$	6	346	6	41	5.2(b)
11	$s/10$	9	537	10	64	5.2(c)
12	$s/10$	12	537	10	64	5.2(c)

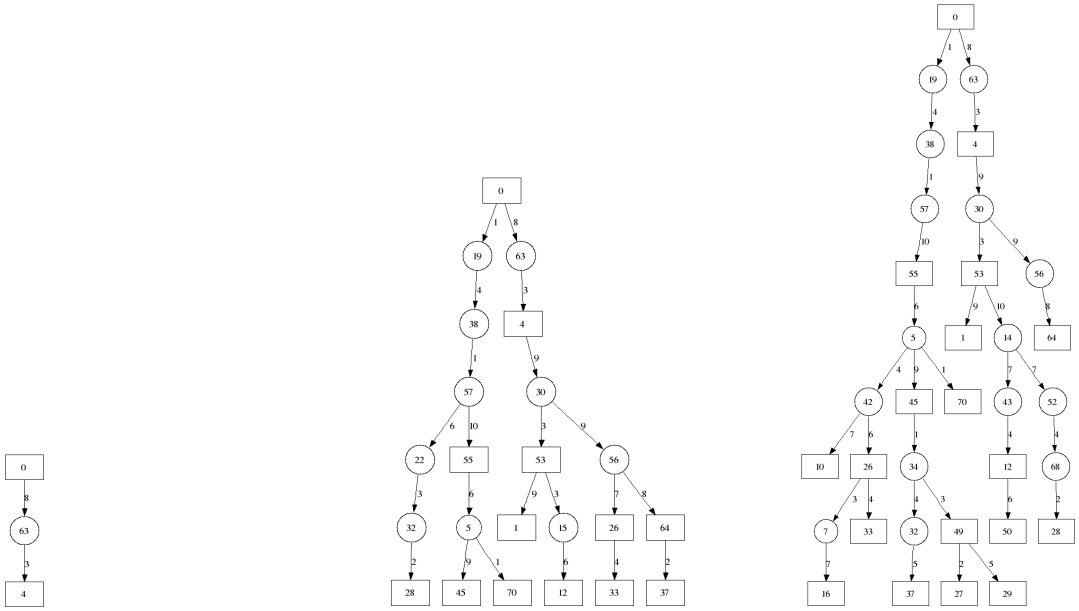
Table 5.2: Optimal solutions for instance *Steinb8* for the 12 different scenarios

For $h = 3$, only two vertices can be reached, yielding a total revenue of 85. For $h = 6$, the total revenue increases to 581 (given by seven profitable vertices) but it is only for $h = 9$ and $h = 12$ that all 19 profitable vertices are reached. A similar analysis can be made for a constant value of the hop limit, for example, $h = 9$: in this case, the reduction in the budget reduces the total collectable revenue from 836 ($b = s$) to 761 ($b = s/5$) and finally to 537 ($b = s/10$). In the sequel, we describe the results for all the instances.

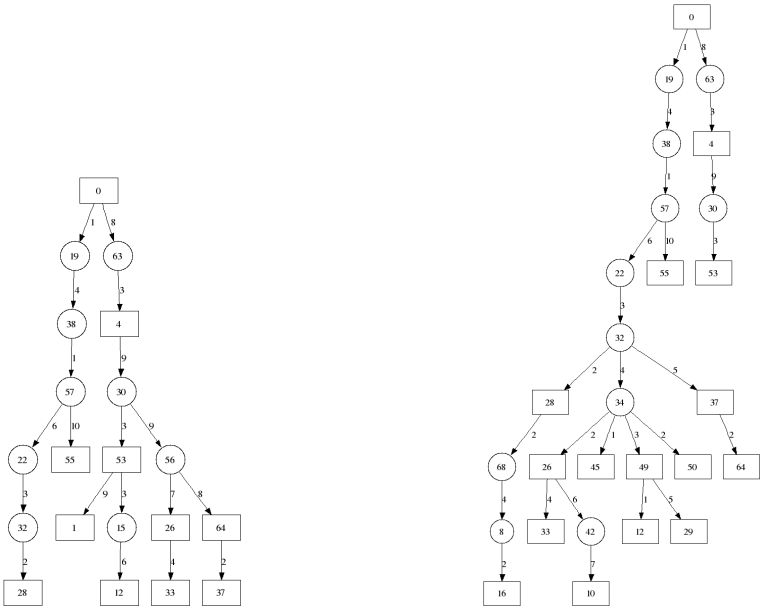
5.4.2 Complete tests

We were first interested in the effect of adding violated inequalities (5.11) and (5.13) while solving the polynomial formulation GG-Hop with and without the relaxation of constraints (5.35). The tests showed that, although for some instances the cuts could help achieve an earlier convergence, it is not usually beneficial to add these constraints and, for this reason, algorithms $A2$ and $A4$ were discarded. Complete tests were, therefore, run on algorithms $A1$, $A3$ and $A5$.

Algorithms $A1$, $A3$ and $A5$ were tested on the two sets of instances described in Section 5.4.1. A maximum CPU time of two hours was allowed for the solution of each scenario of each instance. In Tables 5.4 to 5.14 we present the detailed results of our computational tests. Besides the time needed by the algorithm, we also report the number of cuts (5.11) and (5.13) for algorithms $A5$ and the number of cuts (5.35) for algorithm $A3$. In this case, we indicate the percentage of the total number of constraints (5.35) that were added to the model. For each algorithm, the tables also show the number of branch-and-cut nodes



(a) Scenarios 1,5,9 (b) Scenario 2 (c) Scenarios 3,4

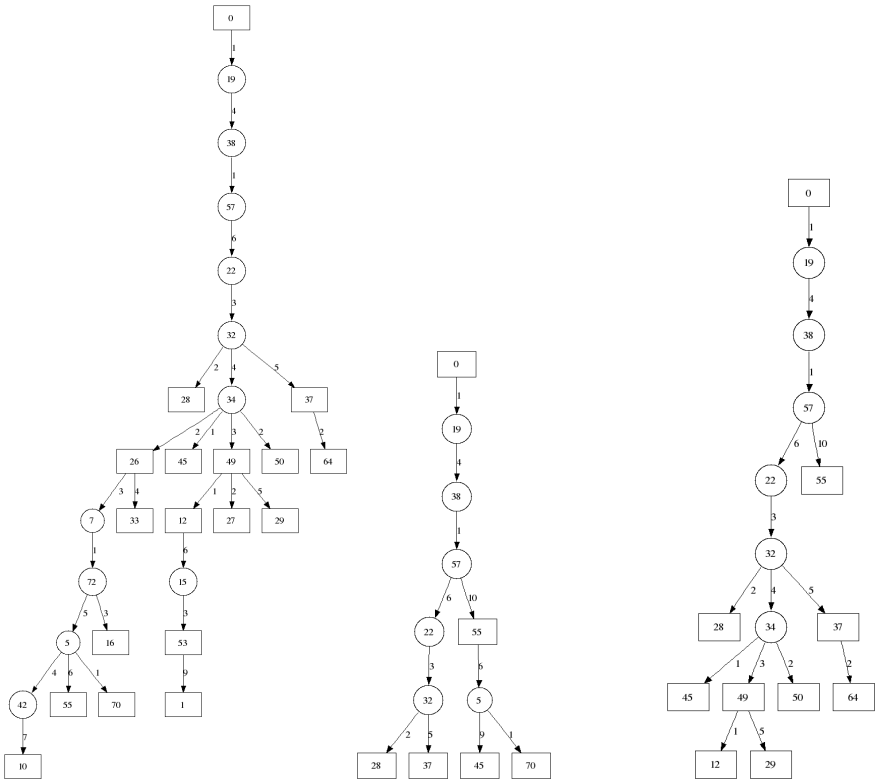


(d) Scenario 6 (e) Scenario 7

Figure 5.1: Solutions for instance *Steinb8* in the different scenarios.

explored. In these tables, when the time limit is not sufficient for the algorithm to converge to a proven optimal solution, we present the MIP gap in brackets. The entry "[inf]" indicates that no feasible solution was identified within the time limit.

For the scenarios with a small hop limit h, $A1$ is clearly superior. Indeed, it is the best



(a) Scenario 8 (b) Scenario 10 (c) Scenarios 11,12

Figure 5.2: Solutions for instance *Steinb8* in the different scenarios (continued).

for all instances *Msteinb* with $h = 3$, and in all instances *Steinc* with $h = 5$. The reason for this behavior seems to be the fact that formulation GG-Hop is extremely strong and its size is reasonable for small values of h. This hypothesis can be confirmed by the fact that a great proportion of these instances are solved at the root node. For small values of h, $A3$ is also effective but less than $A1$. However, formulation GG-Hop quickly grows with h and, for this reason, the relaxation of constraints (5.35) eventually starts to pay off, as one can see in a direct comparison of algorithms $A1$ and $A3$ for the *Steinb* instances with $h = 12$. For instances *Steinc*, $A1$ seems to be always superior to $A3$.

Interestingly, Algorithm $A5$ behaves in a complete different manner. Indeed, for small values of h, the number of generated constraints (5.11) and (5.13) is huge, making the algorithm very inefficient. As soon as h grows, however, fewer cuts are needed for convergence and $A5$ clearly becomes the most efficient approach (see, for example, the results for instances *Steinb* for $h = 12$ in Tables 5.10 and 5.11).

Table 5.3 summarizes the results of these tests. For the three possible solution methods

and the two groups of instances, we indicate the number of scenarios for which the method was the best and the number of scenarios the method could not solve within the imposed time limit. One can observe again the clear polarity between methods $A1$ and $A5$. Indeed, for instances with a small h, $A1$ seems to be the most efficient method, while for instances with a large h, $A5$ yields better results. As indicated earlier, this was somehow expected since for increasing values of h two phenomena happen: on the one hand, the size of formulation GG-Hop quickly increases and, on the other hand, the number of cuts necessary to find a feasible solution in algorithm $A5$ becomes smaller.

		A1		A3		A5	
Group	h	Best	Unsolved	Best	Unsolved	Best	Unsolved
Steinb	3	54	0	1	0	0	29
Steinb	6	48	0	5	0	1	18
Steinb	9	21	0	8	0	25	3
Steinb	12	6	0	14	0	34	0
SteinC	5	30	0	0	0	0	30
SteinC	15	23	0	1	0	6	13
SteinC	25	3	1	2	3	20	0
Total		185	1	31	3	86	93

Table 5.3: Summary of results

As a final remark, note that the instances used in the tests are rather sparse. There are two reasons for this: a) several real network design problems occur on topologies defined by streets and neighborhoods which yield sparse graphs, and b) hop constraints make more sense in sparse graphs, since in complete graphs there is always a direct connection between every pair of vertices. Nevertheless, we have performed some tests to better understand the effect of increasing density without changing the way of computing the budget parameter. Our results indicate that algorithm $A1$ is importantly affected by the increase on density, becoming much slower. The performance of algorithms $A3$ and $A5$, in turn, seems to be quite robust to arc density. These results are somehow expected since the increase of the number of arcs greatly affects the number of variables in $A1$, but has little influence in $A5$. The relaxation of constraints (5.35) becomes more evident as density increases, thus giving $A3$ a clear advantage over $A1$. We have performed further tests in which the density was multiplied by a factor θ and the budget values were taken as s, $s/5\theta$ and $s/10\theta$. In this case, A3 clearly becomes the best algorithm.

5.5 Conclusions

We have proposed several formulations for a modified version of the Steiner Tree Problem with revenues, including budget and hop constraints. These formulations were used to develop branch-and-cut algorithms for the problem. Computational tests have shown that the proposed algorithms are capable of solving a majority of scenarios for instances with up to 500 vertices and 625 arcs. Moreover, our results indicate that the best algorithm depends on the number of allowed hops. For small hop values, algorithms based on position variables are clearly superior, while for large hop values the algorithm based on Dantzig-Fulkerson-Johnson formulation is the most efficient.

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb1	50	63	1	140	0	**0.01**	0	102 (40.48 %)	0.03	279	4245	744	1.77
			5	140	0	**0.01**	0	78 (30.95 %)	0.04	816	2718	666	2.56
			10	140	0	**0.01**	0	78 (30.95 %)	0.04	178	283	100	0.31
Msteinb2	50	63	1	182	0	**0.01**	0	123 (48.81 %)	**0.01**	445	4902	1738	5.79
			5	182	0	**0.01**	0	126 (50.00 %)	0.05	247	1724	740	2.36
			10	182	0	**0.01**	4	118 (46.83 %)	0.07	72	508	206	0.53
Msteinb3	50	63	1	253	0	**0.01**	0	163 (64.68 %)	0.02	9	268	194	0.3
			5	253	0	**0.01**	0	154 (61.11 %)	0.05	4	174	132	0.25
			10	253	0	**0.01**	0	140 (55.56 %)	0.05	4	151	104	0.18
Msteinb4	50	100	1	341	0	**0.02**	1	199 (49.75 %)	0.07	15609	486329	32432	[inf]
			5	341	0	**0.01**	130	252 (63.00 %)	0.53	17751	154921	14198	[inf]
			10	341	0	**0.01**	110	151 (37.75 %)	0.81	26908	147184	17087	[16.83%]
Msteinb5	50	100	1	588	0	**0.02**	0	153 (38.25 %)	0.05	24853	740421	32770	[inf]
			5	588	0	**0.02**	72	142 (35.50 %)	0.36	19146	281447	17530	[inf]
			10	565	0	**0.02**	0	58 (14.50 %)	0.05	38848	109378	10819	4416.86
Msteinb6	50	100	1	1040	0	**0.02**	11	250 (62.50 %)	0.11	23816	534587	31211	[inf]
			5	1035	0	**0.01**	68	163 (40.75 %)	0.33	24005	365437	31894	[31.84%]
			10	610	10	**0.04**	120	152 (38.00 %)	0.51	41167	78719	9904	4432.77
Msteinb7	75	94	1	87	0	**0.02**	0	211 (56.12 %)	0.04	28	1279	528	2.08
			5	87	0	**0.01**	4	183 (48.67 %)	0.09	4	657	292	0.84
			10	87	0	**0.01**	78	174 (46.28 %)	0.29	6	421	210	0.5
Msteinb8	75	94	1	85	0	**0.01**	0	206 (54.79 %)	0.04	116	2154	830	3.31
			5	85	0	**0.01**	0	197 (52.39 %)	0.08	40	970	570	2.72
			10	85	0	**0.01**	8	176 (46.81 %)	0.12	16	324	238	0.75
Msteinb9	75	94	1	596	0	**0.01**	0	231 (61.44 %)	0.03	9521	262941	42102	4228.27
			5	596	0	**0.01**	2	190 (50.53 %)	0.11	7566	73353	15871	1087.73
			10	483	3	**0.02**	80	175 (46.54 %)	0.29	1132	7056	2278	19.49

Table 5.4: Results - Instances MStein, $h = 3$

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb10	75	150	1	319	0	**0.02**	10	359 (59.83 %)	0.18	7115	1409190	47418	[inf]
			5	319	0	**0.02**	391	388 (64.67 %)	2.04	6383	432980	22686	[inf]
			10	319	0	**0.02**	125	275 (45.83 %)	1.48	14440	304253	22341	[64.47%]
Msteinb11	75	150	1	316	0	**0.02**	4	393 (65.50 %)	0.15	6893	1100426	45110	[inf]
			5	316	0	**0.02**	26	352 (58.67 %)	0.46	6737	353237	25042	[inf]
			10	305	0	**0.02**	213	329 (54.83 %)	3.07	6102	178547	21538	[76.02%]
Msteinb12	75	150	1	1169	0	**0.02**	0	418 (69.67 %)	0.09	6015	906773	35515	[inf]
			5	1169	0	**0.02**	78	349 (58.17 %)	0.7	6734	664978	41020	[inf]
			10	1017	1	**0.03**	116	202 (33.67 %)	0.71	12710	335980	28226	[inf]
Msteinb13	100	125	1	147	0	**0.02**	0	241 (48.20 %)	0.06	2171	377891	51584	[81.27%]
			5	147	0	**0.02**	47	234 (46.80 %)	0.39	6306	183147	30340	[75.94%]
			10	147	0	**0.02**	46	175 (35.00 %)	0.51	2386	18519	4848	162.37
Msteinb14	100	125	1	263	0	**0.02**	0	272 (54.40 %)	0.07	189	14482	3400	54.13
			5	263	0	**0.02**	0	252 (50.40 %)	0.14	872	12783	4146	95.79
			10	263	0	**0.01**	42	253 (50.60 %)	0.56	162	4380	1538	15.09
Msteinb15	100	125	1	1061	0	**0.02**	0	280 (56.00 %)	0.05	1639	709689	68728	[61.41%]
			5	1061	0	**0.01**	0	247 (49.40 %)	0.17	26630	121736	25932	[33.34%]
			10	830	0	**0.02**	51	173 (34.60 %)	0.55	1491	6735	2132	28.03
Msteinb16	100	200	1	479	0	**0.03**	0	473 (59.13 %)	0.19	3196	1048853	38092	[inf]
			5	479	0	**0.02**	28	460 (57.50 %)	0.65	5941	232061	13442	[inf]
			10	479	0	**0.03**	495	522 (65.25 %)	7.34	4118	177306	18746	[inf]
Msteinb17	100	200	1	254	0	**0.03**	0	602 (75.25 %)	0.2	1875	126786	12748	[inf]
			5	254	0	**0.02**	28	516 (64.50 %)	0.74	4805	121313	13210	[inf]
			10	254	0	**0.03**	0	464 (58.00 %)	0.47	3193	178010	18231	[76.62%]
Msteinb18	100	200	1	1298	0	**0.03**	0	542 (67.75 %)	0.15	6331	1334390	43226	[inf]
			5	1298	0	**0.03**	5	454 (56.75 %)	0.37	3254	540237	30572	[inf]
			10	1132	2	**0.04**	522	461 (57.63 %)	3.37	3424	563325	34466	[79.32%]

Table 5.5: Results - Instances MStein, $h = 3$ (continued)

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb1	50	63	1	403	0	**0.02**	0	191 (30.32 %)	0.05	4351	3026	598	8.15
			5	403	0	**0.02**	23	57 (9.05 %)	0.07	102	182	78	0.32
			10	341	0	0.02	0	23 (3.65 %)	0.02	0	0	0	**0.01**
Msteinb2	50	63	1	612	0	**0.01**	2	359 (56.98 %)	0.33	14483	8200	2004	53.78
			5	491	7	**0.04**	243	304 (48.25 %)	0.68	436	578	296	1.45
			10	300	9	**0.04**	91	118 (18.73 %)	0.21	65	6	62	0.24
Msteinb3	50	63	1	963	0	**0.01**	0	401 (63.65 %)	0.22	5256	9587	1464	18.7
			5	807	0	**0.02**	170	424 (67.30 %)	0.64	720	1154	296	1.75
			10	624	0	**0.02**	4	85 (13.49 %)	0.11	11	45	36	0.1
Msteinb4	50	100	1	455	5	0.19	0	264 (26.40 %)	**0.13**	20	1181	88	0.33
			5	455	10	**0.22**	125	332 (33.20 %)	0.98	939	11760	702	6.92
			10	447	54	**0.69**	163	255 (25.50 %)	1.04	546	4410	496	11.35
Msteinb5	50	100	1	666	0	**0.05**	2	232 (23.20 %)	0.14	31	2104	90	0.53
			5	666	3	**0.24**	98	352 (35.20 %)	1	16	1116	58	0.38
			10	643	4	0.13	25	85 (8.50 %)	**0.12**	73	237	26	0.21
Msteinb6	50	100	1	1269	4	0.39	11	480 (48.00 %)	**0.27**	26	5029	210	0.87
			5	1257	18	**0.59**	94	358 (35.80 %)	0.87	33813	57648	3696	740.81
			10	839	36	**0.65**	237	267 (26.70 %)	1.04	1392	5325	538	14.19
Msteinb7	75	94	1	627	0	**0.02**	15	572 (60.85 %)	0.33	27591	115272	21226	[6.97%]
			5	627	0	**0.02**	30	224 (23.83 %)	0.3	195	706	330	1.69
			10	432	2	**0.03**	212	219 (23.30 %)	0.6	43	70	80	0.38
Msteinb8	75	94	1	581	0	**0.02**	0	571 (60.74 %)	0.79	34992	128342	23778	6612.82
			5	535	0	**0.02**	169	514 (54.68 %)	1.57	31888	39172	9957	2307.75
			10	346	0	**0.03**	344	494 (52.55 %)	2.16	507	2076	798	4.9
Msteinb9	75	94	1	1698	0	**0.02**	0	497 (52.87 %)	0.18	82814	238660	26620	[3.57%]
			5	1343	0	**0.04**	39	240 (25.53 %)	0.35	1787	4997	1478	17.47
			10	816	12	**0.1**	210	235 (25.00 %)	0.66	66	24	22	0.3

Table 5.6: Results - Instances MStein, $h = 6$

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb10	75	150	1	702	3	0.45	4	508 (33.87 %)	**0.33**	15377	1993727	42816	[inf]
			5	702	7	**0.68**	80	511 (34.07 %)	1.67	749	10127	424	9.69
			10	668	77	**1.2**	289	318 (21.20 %)	2.17	1636	13036	694	34.24
Msteinb11	75	150	1	893	4	**0.18**	0	430 (28.67 %)	0.19	16751	1384101	34858	[inf]
			5	893	8	**0.35**	583	727 (48.47 %)	9.23	14482	340611	18360	[inf]
			10	829	3	**0.54**	619	625 (41.67 %)	7.41	2007	22939	2330	80.31
Msteinb12	75	150	1	1867	2	**0.24**	0	663 (44.20 %)	0.28	18908	1464622	36796	[inf]
			5	1847	20	**0.76**	278	606 (40.40 %)	3.54	10881	1105823	42612	[inf]
			10	1384	3	**0.09**	48	215 (14.33 %)	0.64	370	11652	1132	14.92
Msteinb13	100	125	1	785	0	**0.05**	0	516 (41.28 %)	0.18	10178	187968	23580	1415.72
			5	674	31	**0.14**	378	503 (40.24 %)	1.87	16585	15337	5337	696.88
			10	452	8	**0.1**	172	394 (31.52 %)	1.2	164	340	164	1.7
Msteinb14	100	125	1	1296	0	**0.02**	4	773 (61.84 %)	1.18	27541	250445	27430	[19.67%]
			5	977	1	**0.04**	497	525 (42.00 %)	2.45	1026	4922	1968	30.07
			10	595	0	**0.04**	19	89 (7.12 %)	0.3	0	1	10	0.05
Msteinb15	100	125	1	2555	0	**0.04**	0	774 (61.92 %)	0.38	18788	457920	42774	[inf]
			5	1858	12	**0.13**	173	230 (18.40 %)	1.09	313	1801	430	4.3
			10	1040	38	**0.16**	252	297 (23.76 %)	1.19	177	50	38	0.81
Msteinb16	100	200	1	840	1	**0.31**	18	1414 (70.70 %)	11.59	5564	1900715	39986	[inf]
			5	840	15	**0.68**	407	1082 (54.10 %)	13.34	9278	505404	17587	[inf]
			10	767	106	**2.65**	437	805 (40.25 %)	8.21	7428	98879	6036	[4.12%]
Msteinb17	100	200	1	1299	0	**0.06**	19	1588 (79.40 %)	6.58	1885	1600101	46970	[inf]
			5	1299	0	**0.06**	400	1225 (61.25 %)	18.37	9059	930397	30941	[inf]
			10	1091	0	**0.07**	442	819 (40.95 %)	9.04	11986	273874	24689	[inf]
Msteinb18	100	200	1	2585	10	1.42	11	1088 (54.40 %)	**0.94**	12412	2463977	28620	[inf]
			5	2575	8	**0.84**	43	581 (29.05 %)	1.77	6870	880530	24466	[inf]
			10	1917	0	**0.09**	160	562 (28.10 %)	2.97	13420	580741	26769	[inf]

Table 5.7: Results - Instances MStein, $h = 6$ (continued)

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb1	50	63	1	467	0	**0.03**	5	236 (23.41 %)	0.1	13	372	82	0.17
			5	431	17	0.17	3	78 (7.74 %)	0.05	2	0	2	**0.03**
			10	341	0	0.03	0	26 (2.58 %)	0.03	0	0	0	**0.01**
Msteinb2	50	63	1	696	0	**0.02**	0	394 (39.09 %)	0.14	30	934	192	0.45
			5	600	7	0.09	31	228 (22.62 %)	0.19	16	6	12	**0.07**
			10	300	56	0.31	59	60 (5.95 %)	**0.18**	62	0	64	0.24
Msteinb3	50	63	1	1205	0	**0.02**	9	733 (72.72 %)	3.31	18	689	94	0.23
			5	924	4	**0.08**	87	324 (32.14 %)	0.37	10	46	16	0.09
			10	649	0	0.02	0	80 (7.94 %)	0.02	0	0	0	**0.01**
Msteinb4	50	100	1	455	1	0.23	18	1048 (65.50 %)	6.65	0	17	2	**0.02**
			5	455	0	0.08	4	219 (13.69 %)	0.15	0	13	2	**0.04**
			10	455	0	0.36	2	143 (8.94 %)	0.18	2	424	4	**0.08**
Msteinb5	50	100	1	666	3	0.42	0	359 (22.44 %)	**0.14**	9	2460	46	0.36
			5	666	7	1.36	10	382 (23.88 %)	0.56	5	1110	30	**0.2**
			10	652	0	0.09	0	37 (2.31 %)	0.05	0	0	0	**0.02**
Msteinb6	50	100	1	1269	2	0.59	0	391 (24.44 %)	0.25	0	479	24	**0.08**
			5	1262	18	**2.03**	221	630 (39.38 %)	2.34	456	5284	322	10.38
			10	903	7	0.29	4	160 (10.00 %)	0.18	5	6	4	**0.05**
Msteinb7	75	94	1	674	2	**0.07**	16	764 (50.80 %)	0.55	394	2127	380	2.66
			5	662	0	0.04	0	202 (13.43 %)	0.1	0	2	2	**0.03**
			10	432	21	**0.12**	50	163 (10.84 %)	0.25	21	3	18	0.14
Msteinb8	75	94	1	836	0	**0.04**	4	477 (31.72 %)	0.27	8338	35111	4342	112.73
			5	761	36	**0.26**	439	783 (52.06 %)	5.08	4182	4391	1086	28.87
			10	537	2	0.14	223	431 (28.66 %)	1.84	6	115	26	**0.13**
Msteinb9	75	94	1	1761	2	**0.15**	14	926 (61.57 %)	3.23	52369	134183	12568	3138.36
			5	1388	0	**0.06**	3	217 (14.43 %)	0.12	8	33	10	0.09
			10	816	107	0.86	142	351 (23.34 %)	0.61	60	2	14	**0.26**

Table 5.8: Results - Instances MStein, $h = 9$

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb10	75	150	1	702	3	2.14	0	432 (18.00 %)	**0.26**	26	5045	190	1.67
			5	702	5	1.28	89	577 (24.04 %)	1.55	21	3414	108	**1.19**
			10	694	1	0.29	47	217 (9.04 %)	0.96	2	27	10	**0.16**
Msteinb11	75	150	1	893	5	**3.1**	34	1850 (77.08 %)	15.63	35616	1748774	24754	[inf]
			5	893	7	6.47	105	846 (35.25 %)	2.75	6	1501	46	**0.32**
			10	855	4	1.59	213	598 (24.92 %)	3.78	1	136	12	**0.11**
Msteinb12	75	150	1	1867	5	4.12	37	1537 (64.04 %)	4.23	58	3862	114	**1.47**
			5	1866	3	**0.9**	16	519 (21.63 %)	1.08	98	14165	396	5.97
			10	1401	48	1.18	53	247 (10.29 %)	**0.46**	168	369	78	1.43
Msteinb13	100	125	1	785	2	**0.17**	13	826 (41.30 %)	0.45	242	12946	1378	12.25
			5	745	8	0.41	28	213 (10.65 %)	0.3	11	8	20	**0.23**
			10	465	69	0.87	162	342 (17.10 %)	1.21	70	33	44	**0.54**
Msteinb14	100	125	1	1403	0	**0.05**	10	1296 (64.80 %)	10.76	18	1863	176	1.42
			5	1033	38	**0.44**	120	365 (18.25 %)	0.89	128	580	248	2.31
			10	595	27	0.21	118	139 (6.95 %)	0.85	0	0	10	**0.05**
Msteinb15	100	125	1	2555	1	**0.19**	20	1341 (67.05 %)	6.09	1715	18306	1336	26.69
			5	1891	0	0.06	0	120 (6.00 %)	0.06	0	0	0	**0.03**
			10	1086	138	2.39	202	252 (12.60 %)	1.26	31	6	10	**0.21**
Msteinb16	100	200	1	840	5	**3.79**	26	2045 (63.91 %)	36.87	5209	206654	2576	167.4
			5	840	6	3.39	77	841 (26.28 %)	2.84	10	5015	194	**1.82**
			10	800	21	7.4	10	223 (6.97 %)	**0.49**	801	33077	1494	118.41
Msteinb17	100	200	1	1299	5	4.4	0	623 (19.47 %)	**0.44**	20732	1777216	19200	[inf]
			5	1299	14	**4.91**	367	1614 (50.44 %)	21.89	9364	627702	17550	[inf]
			10	1178	32	**2.34**	986	1256 (39.25 %)	23.74	15242	115951	6176	5320.1
Msteinb18	100	200	1	2585	8	**7.77**	31	2098 (65.56 %)	29.19	1228	174680	1846	75.04
			5	2585	0	0.86	12	482 (15.06 %)	**0.72**	2824	297448	4606	1613
			10	1997	8	0.99	7	354 (11.06 %)	**0.41**	667	28636	992	35.4

Table 5.9: Results - Instances MStein, $h = 9$ (continued)

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb1	50	63	1	467	0	0.31	1	202 (14.57 %)	0.07	3	24	28	**0.06**
			5	431	2	0.14	3	50 (3.61 %)	0.05	2	0	2	**0.03**
			10	341	0	0.04	0	27 (1.95 %)	0.03	0	0	0	**0.01**
Msteinb2	50	63	1	696	0	**0.04**	8	324 (23.38 %)	0.13	2	35	14	0.05
			5	600	30	0.37	51	332 (23.95 %)	0.42	13	0	6	**0.05**
			10	300	77	0.77	55	64 (4.62 %)	**0.17**	62	0	64	0.24
Msteinb3	50	63	1	1205	0	**0.03**	2	535 (38.60 %)	0.17	50	1136	166	0.48
			5	931	39	0.45	101	405 (29.22 %)	0.55	24	27	8	**0.09**
			10	649	0	0.03	0	124 (8.95 %)	0.04	0	0	0	**0.01**
Msteinb4	50	100	1	455	4	2.38	0	393 (17.86 %)	0.24	0	2	2	**0.02**
			5	455	18	6.77	58	449 (20.41 %)	0.77	0	0	2	**0.03**
			10	455	4	1.9	0	92 (4.18 %)	0.11	0	375	4	**0.04**
Msteinb5	50	100	1	666	3	2.82	8	578 (26.27 %)	0.59	0	1005	10	**0.05**
			5	666	6	5.54	0	185 (8.41 %)	**0.1**	6	1066	22	0.2
			10	652	0	0.14	0	84 (3.82 %)	0.07	0	0	0	**0.02**
Msteinb6	50	100	1	1269	3	2.83	6	502 (22.82 %)	0.33	0	286	20	**0.07**
			5	1262	456	21.66	223	687 (31.23 %)	2.39	0	1001	12	**0.07**
			10	903	28	2.11	4	135 (6.14 %)	0.14	6	0	4	**0.06**
Msteinb7	75	94	1	674	1	0.39	3	486 (23.50 %)	0.17	0	181	60	**0.14**
			5	662	0	0.14	1	276 (13.35 %)	0.16	0	0	2	**0.02**
			10	432	56	0.85	109	345 (16.68 %)	0.79	21	0	18	**0.14**
Msteinb8	75	94	1	836	3	**0.21**	0	532 (25.73 %)	0.23	39	1474	262	1.05
			5	832	7	0.57	225	673 (32.54 %)	2.55	0	118	12	**0.05**
			10	537	25	0.88	628	732 (35.40 %)	5.97	11	74	36	**0.15**
Msteinb9	75	94	1	1761	0	**0.07**	0	747 (36.12 %)	0.3	2	26	10	0.09
			5	1388	4	0.35	42	448 (21.66 %)	0.69	44	90	40	**0.33**
			10	816	107	1.2	154	139 (6.72 %)	0.43	60	3	14	**0.26**

Table 5.10: Results - Instances MStein, $h = 12$

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
Msteinb10	75	150	1	702	9	10.65	0	173 (5.24 %)	**0.13**	17	7909	150	1.55
			5	702	14	21.13	13	523 (15.85 %)	**0.64**	21	4675	74	1.21
			10	694	2	3.48	3	176 (5.33 %)	0.34	2	1	10	**0.13**
Msteinb11	75	150	1	893	5	8.32	12	834 (25.27 %)	**0.92**	12	7363	186	1.45
			5	893	11	16.9	117	1044 (31.64 %)	3.86	1	131	28	**0.14**
			10	855	44	5.57	544	861 (26.09 %)	9.88	2	169	12	**0.1**
Msteinb12	75	150	1	1867	5	10.86	5	619 (18.76 %)	**0.37**	15	10956	100	1.37
			5	1866	0	0.58	0	466 (14.12 %)	0.28	4	341	8	**0.18**
			10	1401	58	1.33	69	261 (7.91 %)	**0.5**	130	200	26	0.68
Msteinb13	100	125	1	785	3	0.77	7	1052 (38.25 %)	0.74	0	7	2	**0.03**
			5	745	35	1.54	27	312 (11.35 %)	**0.31**	26	3	22	0.32
			10	465	96	1.67	382	653 (23.75 %)	3.58	55	5	30	**0.41**
Msteinb14	100	125	1	1403	2	**0.28**	5	917 (33.35 %)	0.54	12	979	194	1.38
			5	1038	52	0.75	73	533 (19.38 %)	0.86	65	2	18	**0.37**
			10	595	21	0.39	6	161 (5.85 %)	0.39	0	0	10	**0.05**
Msteinb15	100	125	1	2555	0	**0.16**	0	997 (36.25 %)	0.4	10	845	122	1.12
			5	1891	0	0.09	0	294 (10.69 %)	0.12	0	0	0	**0.03**
			10	1109	2	0.29	3	149 (5.42 %)	**0.19**	21	0	8	0.21
Msteinb16	100	200	1	840	1	10.31	5	1595 (36.25 %)	**1.44**	80	42144	516	9.72
			5	840	7	28.64	116	1381 (31.39 %)	4.8	14	5972	120	**2.27**
			10	800	7	16.45	114	620 (14.09 %)	**3.03**	104	2622	128	4.09
Msteinb17	100	200	1	1299	2	10.11	26	2770 (62.95 %)	38.44	69	59844	352	**9.11**
			5	1299	6	19.32	112	1261 (28.66 %)	6.73	1	2420	58	**0.52**
			10	1225	0	0.29	3	256 (5.82 %)	0.27	2	2	0	**0.12**
Msteinb18	100	200	1	2585	2	16.83	5	1229 (27.93 %)	**0.97**	21	12678	166	4.09
			5	2585	4	20.03	83	938 (21.32 %)	**3.4**	242	38828	358	13.77
			10	1997	122	14.64	302	721 (16.39 %)	**5.2**	390	5088	94	5.29

Table 5.11: Results - Instances MStein, $h = 12$ (continued)

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
steinc1_10	500	625	1	8	0	**0.11**	0	2959 (59.18 %)	34.76	40	277910	22912	[inf]
steinc1_100			1	71	0	**0.12**	0	2959 (59.18 %)	34.67	40	277910	22912	[inf]
steinc1_10			10	8	0	**0.12**	998	3121 (62.42 %)	186.06	362	161716	23259	[inf]
steinc1_100			10	71	0	**0.12**	1460	3217 (64.34 %)	536.13	285	139990	20097	[inf]
steinc1_10			30	8	0	**0.13**	1253	2045 (40.90 %)	161.54	1502	29800	5843	[inf]
steinc1_100			30	71	0	**0.12**	3104	2531 (50.62 %)	416.89	1207	26545	5265	[inf]
steinc2_10			1	32	0	**0.11**	22	2865 (57.30 %)	10.75	112	92822	21332	[inf]
steinc2_100			1	328	0	**0.11**	22	2865 (57.30 %)	10.8	112	92822	21332	[inf]
steinc2_10			10	32	0	**0.13**	1595	2831 (56.62 %)	308.48	189	68055	9182	[inf]
steinc2_100			10	328	0	**0.13**	1244	3125 (62.50 %)	202.69	324	77146	9158	[inf]
steinc2_10			30	32	0	**0.13**	1440	1937 (38.74 %)	105.54	2549	30787	10650	[inf]
steinc2_100			30	328	0	**0.13**	1365	1937 (38.74 %)	124.22	460	31898	12087	[66.77%]
steinc3_10			1	151	0	**0.12**	0	2906 (58.12 %)	4.99	29	166681	19320	[inf]
steinc3_100			1	1519	0	**0.13**	0	2906 (58.12 %)	5.07	30	129354	17600	[inf]
steinc3_10			10	151	0	**0.13**	613	2496 (49.92 %)	66.16	266	105634	19968	[inf]
steinc3_100			10	1519	0	**0.12**	307	2356 (47.12 %)	33.79	1462	111538	17429	[inf]
steinc3_10			30	95	14	**0.2**	887	1398 (27.96 %)	48.15	1810	79964	13165	[34.42%]
steinc3_100			30	968	45	**0.25**	1719	1591 (31.82 %)	65.66	1249	77391	14692	[91.68%]
steinc4_10			1	115	0	**0.11**	0	3141 (62.82 %)	2.99	127	102621	20354	[inf]
steinc4_100			1	1148	0	**0.12**	0	3147 (62.94 %)	3.06	163	103229	22120	[inf]
steinc4_10			10	115	0	**0.12**	109	2857 (57.14 %)	12.08	291	77732	16095	[inf]
steinc4_100			10	1148	0	**0.12**	182	2931 (58.62 %)	30.2	572	71038	15558	[inf]
steinc4_10			30	84	4	**0.14**	1537	3043 (60.86 %)	102.04	1589	53224	14747	[90.50%]
steinc4_100			30	854	8	**0.15**	1785	1816 (36.32 %)	109.94	1430	40269	14014	[96.39%]
steinc5_10			1	258	0	**0.12**	0	3220 (64.40 %)	1.31	92	759835	28310	[inf]
steinc5_100			1	2600	0	**0.11**	0	3226 (64.52 %)	1.32	98	907096	34754	[inf]
steinc5_10			10	258	0	**0.12**	227	2897 (57.94 %)	14.48	901	141186	16085	[inf]
steinc5_100			10	2600	0	**0.12**	78	2893 (57.86 %)	24.97	459	138829	17203	[inf]
steinc5_10			30	154	0	**0.12**	1404	1860 (37.20 %)	138.3	1196	74855	15840	[inf]
steinc5_100			30	1584	0	**0.12**	683	1619 (32.38 %)	71.8	2236	111998	15738	[14.36%]

Table 5.12: Results - Instances steinc, $h = 5$

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
steinc1_10	500	625	1	27	0	**0.48**	44	7660 (43.77 %)	43.99	100	1293143	25266	[inf]
steinc1_100			1	274	0	**0.47**	44	7660 (43.77 %)	43.98	100	1293143	25266	[inf]
steinc1_10			10	27	0	**0.47**	325	2978 (17.02 %)	70.68	2823	57394	8468	[inf]
steinc1_100			10	274	0	**0.49**	797	5114 (29.22 %)	347.12	1975	40568	6840	[11,67%]
steinc1_10			30	27	0	**0.49**	493	1641 (9.38 %)	49.8	5	6	16	1.74
steinc1_100			30	274	0	**0.48**	403	1558 (8.90 %)	29.79	5	6	16	1.74
steinc2_10			1	59	3	**2.55**	227	10525 (60.14 %)	520.89	117	1127462	16528	[inf]
steinc2_100			1	604	3	**2.57**	227	10525 (60.14 %)	521.19	117	1127462	16528	[inf]
steinc2_10			10	59	3	**2.57**	613	3319 (18.97 %)	79.3	2423	34708	5454	[5,08%]
steinc2_100			10	604	3	**2.52**	1840	5177 (29.58 %)	450.5	320	6427	1180	229.07
steinc2_10			30	53	216	58.32	12	290 (1.66 %)	**2.57**	15	1011	48	7.03
steinc2_100			30	546	319	224.28	211	1114 (6.37 %)	11.98	2	290	32	**3.17**
steinc3_10			1	439	3	**49.95**	77	10263 (58.65 %)	900.85	580833	13260	1569	7105.42
steinc3_100			1	4463	3	**50.01**	65	10126 (57.86 %)	996.52	644089	14220	988	[inf]
steinc3_10			10	289	90	114.05	2834	4548 (25.99 %)	397.77	82	683	120	**28.93**
steinc3_100			10	2971	310	190.13	4354	4658 (26.62 %)	663.3	464	1447	192	**85.81**
steinc3_10			30	129	3	1.32	430	837 (4.78 %)	24.25	0	41	8	**0.5**
steinc3_100			30	1343	0	1	643	996 (5.69 %)	35.2	0	56	8	**0.43**
steinc4_10			1	648	1	**19.28**	25	9754 (55.74 %)	368.48	126	1395060	23054	[inf]
steinc4_100			1	6566	1	**19.21**	25	9672 (55.27 %)	306.99	126	1395060	23054	[inf]
steinc4_10			10	336	20	**15.05**	826	2958 (16.90 %)	163.98	5722	36147	5306	2740.75
steinc4_100			10	3458	214	**77.28**	3150	4676 (26.72 %)	509.52	1152	97257	12709	[inf]
steinc4_10			30	134	13	**13.54**	460	1424 (8.14 %)	37.41	290	4268	1464	348.86
steinc4_100			30	1380	31	**9.75**	119	1021 (5.83 %)	9.8	148	113	90	23.36
steinc5_10			1	1248	0	**45.69**	121	11855 (67.74 %)	871.98	3770	589718	7092	[inf]
steinc5_100			1	12533	0	**45.58**	59	11578 (66.16 %)	760.04	3064	529800	7034	[inf]
steinc5_10			10	494	8	**5.83**	1693	3817 (21.81 %)	251.73	1044	61998	4638	1234.44
steinc5_100			10	5032	22	**7.13**	1400	3540 (20.23 %)	129.9	262	44232	3516	1465.93
steinc5_10			30	182	11	**8.63**	1349	2558 (14.62 %)	82.8	202	1990	458	93.38
steinc5_100			30	1857	93	22.24	638	1668 (9.53 %)	50.2	60	70	44	**9.97**

Table 5.13: Results - Instances steinc, $h = 15$

Instance					A1		A3			A5			
Name	n	$\|E\|$	b	Opt	B&B nodes	Seconds	B&B nodes	Cuts (%)	Seconds	B&B nodes	Sep	Hops	Seconds
steinc1_10	500	625	1	27	2	**8.8**	17	6547 (21.82 %)	14.49	28	18727	982	222.51
steinc1_100			1	274	2	8.76	17	6547 (21.82 %)	14.55	28	18727	982	222.87
steinc1_10			10	27	2	8.8	1490	8301 (27.67 %)	878.27	28	1655	272	43.76
steinc1_100			10	274	2	8.77	1257	6922 (23.07 %)	682.47	28	1655	272	43.81
steinc1_10			30	27	2	9.07	1473	2973 (9.91 %)	230.37	7	0	38	**2.86**
steinc1_100			30	274	2	8.97	553	2459 (8.20 %)	56.54	7	0	38	**2.85**
steinc2_10			1	59	0	5.72	2	1414 (4.71 %)	**2.73**	0	178	82	7.24
steinc2_100			1	604	0	5.72	2	1414 (4.71 %)	**2.73**	0	178	82	7.3
steinc2_10			10	59	0	**5.64**	430	4207 (14.02 %)	57.57	0	178	82	7.26
steinc2_100			10	604	0	**5.69**	908	4733 (15.78 %)	183.89	0	178	82	7.26
steinc2_10			30	53	348	431.84	376	1174 (3.91 %)	24.98	17	1001	60	**8.5**
steinc2_100			30	546	296	362.61	390	1716 (5.72 %)	24.9	14	292	30	**4.12**
steinc3_10			1	439	0	223.75	166	17584 (58.61 %)	257.3	0	1002	8	**2.16**
steinc3_100			1	4463	0	184.55	84	15069 (50.23 %)	120.87	0	1002	8	**2.15**
steinc3_10			10	289	501	4293.36	7728	7496 (24.99 %)	3309.14	64	35	56	**18.46**
steinc3_100			10	2979	223	[inf]	8601	8351 (27.84 %)	[0.23%]	204	1001	64	**36.3**
steinc3_10			30	129	47	153.97	626	1920 (6.40 %)	44.77	0	0	8	**0.51**
steinc3_100			30	1343	3	8.17	130	1464 (4.88 %)	13.47	0	6	8	**0.44**
steinc4_10			1	648	1	539.17	101	19869 (66.23 %)	[0.31%]	8	9412	328	112.65
steinc4_100			1	6566	1	542.52	66	19077 (63.59 %)	[inf]	8	9412	328	112.65
steinc4_10			10	341	5	14.8	1547	4811 (16.04 %)	141.98	3	0	4	**2.34**
steinc4_100			10	3504	30	30.81	514	3361 (11.20 %)	70.68	0	0	4	**0.88**
steinc4_10			30	136	109	1149.3	181	1901 (6.34 %)	26.07	5	0	10	**2.51**
steinc4_100			30	1396	120	115.14	702	2218 (7.39 %)	57.45	46	0	18	**8.64**
steinc5_10			1	1248	5	1009.69	46	18999 (63.33 %)	150.38	3	54477	224	**98.48**
steinc5_100			1	12533	8	1208.92	46	18999 (63.33 %)	150.31	3	54477	224	**98.47**
steinc5_10			10	495	401	139.93	2212	5367 (17.89 %)	550.42	49	1088	54	**21.03**
steinc5_100			10	5044	521	305.44	3242	5150 (17.17 %)	878.85	21	545	36	**10.74**
steinc5_10			30	183	324	1568.97	235	1739 (5.80 %)	18.7	81	0	40	**13.58**
steinc5_100			30	1860	206	89.11	490	2270 (7.57 %)	29.82	76	0	16	**7.08**

Table 5.14: Results - Instances steinc, $h = 25$

Chapter 6

Fast heuristics for the Steiner tree problem with revenues, budget and hop constraints

Abstract

This chapter describes and compares three heuristics for a variant of the Steiner tree problem with revenues, which includes budget and hop constraints. First, a greedy method which obtains good approximations in short computational times is proposed. This initial solution is then improved by means of a destroy-and-repair method or a tabu search algorithm. Computational results compare the three methods in terms of accuracy and speed.

Keywords: Prize collecting, network design, Steiner tree problem, budget, hop constraints, heuristics, tabu search.

6.1 Introduction

The purpose of this chapter is to develop heuristics for the *Steiner Tree Problem with Revenues, Budget and Hop Constraints* (STPRBH). This problem is a generalization of several well-studied problems such as the *Minimum Spanning Tree Problem* (MSTP) and the *Steiner Tree Problem* (STP) described as follows. Let $G = (V, E)$ be a graph with vertex set $V = \{1, \ldots, n\}$, where vertex 1 is the *root* vertex, and edge set $E = \{(i, j) : i, j \in V, i < j\}$. In what follows, (i, j) must be interpreted as (j, i) if $i > j$. Each vertex

has an associated revenue $r_i \geq 0$, and each edge has an associated cost $c_{ij} \geq 0$. Also, let T be a tree in G, and let $v(T)$ and $e(T)$ be the sets of vertices and edges belonging to T, respectively. The STPRBH is the problem of determining a tree T of maximal revenue $r(T) = \sum_{i \in v(T)} r_i$, rooted at vertex 1, and subject to two constraints: a budget constraint limiting the network cost $\sum_{(i,j) \in e(T)} c_{ij}$ to a maximum value b, and hop constraints limiting the number of edges h_i between $i \in v(T)$ and the root vertex to a maximum value h.

The STP is a well-studied problem and so is its version with revenues. The latter problem is useful in situations where one must decide whether or not to serve certain vertices. The idea is that one is not obliged to serve all vertices as in the MSTP, or specified vertices as in the STP. Instead, the number and location of the vertices to be served are part of the decision problem. For example, a cable television company planning to expand its network can decide whether or not to serve a certain area, based on the combination of the expected revenues that will come from the clients in that area and the network expansion cost. Steiner tree problems with revenues have been used to solve design problems in local access networks (Canuto et al., 2001; Cunha et al., 2003), and in heating networks (Ljubić et al., 2005). In network expansion problems, one is usually faced with budgetary constraints that come from the fact that a limited investment budget is available for a certain area or a certain period of time. Although it is quite easy to understand the practical aspect of these constraints, Chapter 4 showed that budget considerations are almost always ignored when dealing with STP with revenues.

Concerning the hop constraints, their inclusion enables the consideration of reliability or transmission delays in the network. LeBlanc et al. (1999) have investigated the effect of hop constraints both on the reliability and on the maintenance of the signal quality in packet-switched telecommunication networks. Gouveia and Magnanti (2003) also use hop constraints to guarantee quality of service in telecommunications networks. Other uses of hop constraints have also been proposed in the literature. Balakrishnan and Altinkemer (1992) have imposed hop constraints to generate alternative communications networks, while Voß (1999) has mentioned the utility of these constraints to model time durability constraints when solving some classes of lot-sizing problems by means of spanning tree models.

In this chapter we propose heuristic methods for the STPRBH. Chapter 5 has shown the limits of exact algorithms for this problem. The heuristic methods proposed here are intended to obtain good approximations for large scale problems for which exact algorithms fail to converge. Several exact and heuristic methods already exist for the STP with profits

(see Chapter 4) and for spanning problems with hop constraints but, to our knowledge, no one has ever proposed approximate algorithms for the more general STPRBH. The closest problems solved by means of heuristic methods seem to be the MSTP and the STP with hop constraints. Lagrangean relaxation heuristics have been proposed for the MSTP with hop constraints (Gouveia, 1995b; Gouveia, 1996; Gouveia, 1998; Gouveia and Requejo, 2001) and for the STP with hop constraints (Gouveia, 1998), while Voß (1999) has developed a greedy method and a tabu search algorithm for the hop constrained STP.

We first propose a fast greedy algorithm which is later followed by one of the following two improvement phases: a simple local search that destroys part of the solution and rebuilds it in a greedy fashion, or a more flexible tabu search algorithm that explores the solution space by means of two simple neighborhoods. In the remainder of this chapter we describe these three algorithms in detail. In the next section, we provide a mathematical formulation for the STPRBH in order to help understand the ideas behind the heuristics. Then, the algorithms themselves are described. The greedy and the destroy-and-repair approaches are presented in Sections 6.3 and 6.4, respectively. Section 6.5 describes the tabu search algorithm while computational results are provided in Section 6.6. The paper ends with some conclusions in Section 6.7.

6.2 Mathematical formulation

We first recall the undirected Dantzig-Fulkerson-Johnson model for the STPRBH, which was presented in Chapter 5. Constraints (6.4) and (6.5) of this model are useful to understand our heuristics. Let x_{ij} and y_i be binary variables associated with edges $(i,j) \in E$ and vertices $i \in V$, respectively. Variable y_i is equal to 1 if vertex i belongs to the solution ($y_1 = 1$), and to 0 otherwise. Similarly, variable x_{ij} is equal to 1 if edge (i,j) belongs to the solution, and to 0 otherwise. For $S \subseteq V$, define $E(S)$ as the set of edges with both end vertices in S. Let also $P = (i_1 = 1, \ldots, i_\ell)$ denote a path originating at the root vertex and containing ℓ vertices. Finally, define $\mathcal{P}_h$ as the set of paths P with $\ell = h + 2$, i.e., paths with $h + 1$ edges. The STPRBH can then be written as:

$$\text{Maximize} \sum_{i \in V} r_i y_i \tag{6.1}$$

subject to

$$\sum_{(i,j)\in E} x_{ij} = \sum_{i\in V} y_i - 1, \tag{6.2}$$

$$\sum_{(i,j)\in E(S)} x_{ij} \leq \sum_{i\in S\setminus\{k\}} y_i, \qquad k \in S \subseteq V,\ |S| \geq 2, \tag{6.3}$$

$$\sum_{(i,j)\in E} c_{ij} x_{ij} \leq b, \tag{6.4}$$

$$\sum_{t=2}^{\ell} x_{i_{t-1},i_t} \leq h, \qquad P = (i_1 = 1, \ldots, i_\ell) \in \mathcal{P}_h, \tag{6.5}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in E, \tag{6.6}$$

$$y_1 = 1, \tag{6.7}$$

$$y_i \in \{0,1\}, \qquad i \in V\setminus\{1\}. \tag{6.8}$$

The goal is to maximize the collected revenues while respecting the budget and hop constraints, (6.4) and (6.5), respectively. Constraint (6.2) guarantees that in the solution the number of edges is equal to the number of vertices minus one, and constraints (6.3) are the connectivity constraints.

6.3 Greedy algorithm

The idea of our greedy algorithm is to iteratively construct a rooted tree while maintaining the feasibility of constraints (6.4) and (6.5). We start with a solution containing only the root vertex. At each step of the algorithm, one adds a path connecting a non-selected profitable vertex to the existing solution. The algorithm stops when it cannot find a profitable vertex that can be added to the solution without incurring a violation of either the budget or the hop constraints. Two basic steps are needed to implement this idea. First, one must be able to produce feasible paths connecting non-selected profitable vertices to the existing solution. Second, one must evaluate these paths and choose the best one (in a greedy sense). These two steps are now detailed.

6.3.1 Finding feasible paths

Adding at each step of the algorithm a path not creating cycles to the existing solution guarantees the satisfaction of constraints (6.2) and (6.3). In order to respect the budget

constraint, the sum of the edge costs in the added path plus the costs of the edges already in the solution must not exceed the budget value. Therefore, one has interest in finding shortest paths connecting profitable vertices to the current tree. On the other hand, the number of the edges in a path must be limited, in order to respect the hop constraints. It is important to note that it does not suffice to guarantee that the added paths have less than h edges. Instead, one must also ensure that the number of edges in the complete path from the added profitable vertex to the root vertex (which might include edges that were already in the existing solution) is limited to h.

Based on these two considerations, it is easy to conclude that a building block for constructing feasible paths is the hop constrained shortest path problem, i.e., the problem of determining a shortest path between two vertices containing at most h edges. This problem can be solved efficiently by dynamic programming (Lawler, 1976). Let $L(i, j, \ell)$ represent the cost of a shortest path between an origin vertex i and a destination vertex j, containing at most ℓ edges. Then

$$\begin{aligned}
&L(i,i,0) = 0, && i \in V,\\
&L(i,j,0) = \infty, && i,j \in V, j \neq i,\\
&L(i,j,\ell) = \min\{L(i,j,\ell-1), \min_{k|(k,j)\in E}\{L(i,k,\ell-1) + c_{kj}\}, && i,j \in V, \ell \geq 1.
\end{aligned}$$

Note that, contrary to what happens in the shortest path problem, an optimal subpath between two vertices is dependent on the complete path to which it belongs. This happens due to the hop constraints, as shown in Figure 6.1. Costs are shown next to the edges. The figure depicts optimal shortest paths from vertex 1 to the other vertices in the graph, with a hop limit of 3. Note that the optimal subpath from vertex 1 to vertex 4 includes vertices 2 and 3 if vertex 4 is the destination, but includes only vertex 6 if vertex 5 is the destination. In implementation terms, this means that one must keep a precedence array for each vertex of the graph vertices while solving the problem.

With an algorithm for the shortest path problem with hop constraints, one can evaluate the cost of connecting the non-selected profitable vertices to the existing tree and develop a simple algorithm. Let T be the current solution. At the first iteration, $v(T) = \{1\}$ and the candidate destination vertices j are those with $L(1, j, h) \leq b$ and $r_j > 0$. Note that if a path violates the hop constraint, the hop constrained shortest path will return ∞ and, therefore, this path will not be considered for insertion in the solution. When at least one path has been added, one must not only consider the new paths that originate at the root vertex, but also all paths that are originated at other vertices in $v(T)$. The algorithm

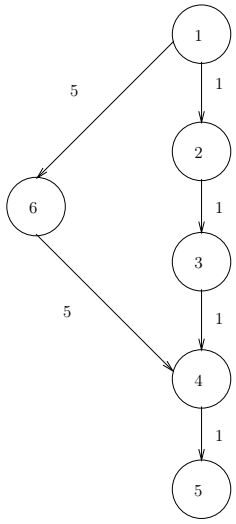


Figure 6.1: Example of different optimal subpaths between two vertices depending on the complete path.

considers for insertion all shortest paths such that

$$L(i, j, h - h_i) \leq b - \sum_{(i,j)\in e(T)} c_{ij}, \qquad i \in v(T), j \notin v(T), r_j > 0.$$

Instead of running the hop constrained shortest path algorithm for all vertices in the current solution, one can simply set the costs of the edges in $e(T)$ to 0 and run the algorithm a single time, with the root vertex as the origin. Because of the dynamic programming structure, a single run is able to compute the hop constrained shortest distances from the root to all the other vertices, saving computation time.

6.3.2 Evaluating the paths

At each step, a single new path is added to the solution. The criterion used to choose the path to be added is an important part of the algorithm. If one makes a choice based only on profits, there is a risk of adding very costly paths that will soon consume the whole budget. On the other hand, if one chooses to add shortest paths, some vertices with a high revenue but at a moderate distance from the current solution may be ignored. Based on preliminary tests, we found that the best strategy was to consider both the costs and the revenues when deciding on the path to be added. We tested several ratios r_j^α / c^β, where α and β are two non-negative numbers, r_j is the revenue of the profitable vertex to be added, and c is the sum of the cost of the edges in the path. We found that the best parameter

setting was $\alpha = 3$ and $\beta = 1$, which was used in all subsequent tests.

Algorithm 6.1 Greedy algorithm for the STPRBH

T is initially the root vertex: $v(T) = \{1\}$.
loop
  $\overline{p} = 0, \overline{c} = 0, \overline{j} = -1$.
  for all vertices $j \notin v(T)$ **do**
    $c = L(1, j, h)$;
    $p = r_j^3/c$.
    if $\sum_{(i,j)\in e(T)} c_{ij} + c > b$ **then**
      $p = 0$.
    end if
    if $p \geq \overline{p}$ **then**
      $\overline{j} = j$;
      $\overline{c} = c$;
      $\overline{p} = p$.
    end if
  end for
  if $\overline{p} = 0$ **then**
    END
  end if
  P is the path from 1 to $\overline{j}$: $P = (i_1 = 1, \ldots, i_\ell = \overline{j})$;
  $c_{i_t, i_{t+1}} = 0$, $t = 1, \ldots, \ell - 1$;
  $v(T) = v(T) \cup v(P)$;
  destroy possible cycles.
end loop
Return T.

Algorithm 6.1, described in the following figure, presents the complete greedy algorithm in details. A few comments on this algorithm are in order. First, note that when adding a path one can also add other profitable vertices in addition to the end vertex. It has proved to be slightly better not to consider these profits when deciding which path should be added. This way, the algorithm tends to add shorter paths first, thus reducing the risk of making wrong decisions due to the myopic character of the choices. Second, because of the hop constraints, one can sometimes introduce cycles when adding a path.

Consider again Figure 6.1 and the situation where $h = 3$, all vertices are profitable and, after the first iteration of the greedy algorithm $v(T) = \{1, 2, 3, 4\}$. Assuming that the budget constraint is not violated, the algorithm will try to add vertex 5 to the solution, using the edges $(1, 6)$ and $(6, 4)$ in order to respect the hop constraint and therefore forming a cycle. When this situation happens, the algorithm destroys the cycle by inspecting vertices with two incoming edges (in this case, vertex 4) and eliminating the edge that was inserted first (in this case, edge $(3, 4)$). This way, the cycle is eliminated and the hop constraints are respected for the entire graph.

6.4 Destroy-and-repair algorithm

The idea of destroying part of the solution and reconstructing it in a different way in order to obtain different and hopefully better solutions has already been used to solve other combinatorial problems. Ropke and Pisinger (2006) have employed such a technique to solve vehicle routing problems with time windows, and Pisinger and Ropke (2007) have applied it to a more general class of routing problems. Voß (1999) has successfully used a similar idea to solve the STP with hop constraints. For the STPRBH, we propose a very simple procedure that consists of sequentially blocking part of the current solution and rerunning the greedy algorithm.

A feasible solution to the STPRBH consists of a tree. Given an initial solution, we simply set the cost of some edges to infinity, one at a time, and we rerun the greedy algorithm with the modified costs. This is done for all edges of the current solution that are incident to a leaf vertex, as shown in Figure 6.2, and the best obtained solution is retained. The procedure is repeated for the new solution until there is no gain obtained with the blocking of a single edge. **Algorithm 6.2** schematically presents the procedure.

Other ways of preventing the greedy algorithm from repeatedly reaching the same solution can be used. For example, one could block all edges in the solution, one at a time. We tested this idea and, although it was much more time consuming, the results were not much better than the ones obtained with **Algorithm 6.2**. In a way this was expected, since the greedy algorithm has a tendency to reconstruct a similar solution if an intermediate edge has been blocked. By concentrating on the edges incident to profitable leaf vertices we increase the probability of really blocking that solution and obtaining something new.

We also tested other neighborhoods such as blocking a vertex of the solution (by setting the cost of all its incident edges to infinity). The results turned out to be worse than those

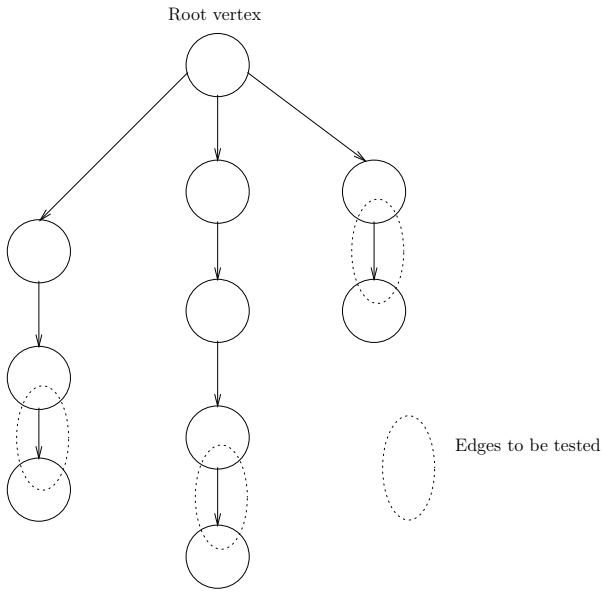


Figure 6.2: Edges incident to leaf vertices.

obtained with the original algorithm. In this case, we think that the explanation comes from the fact that blocking a vertex is probably too restrictive and by doing so we may eliminate several good solutions.

6.5 Tabu search algorithm

A tabu search algorithm was also developed for the STPRBH. The main idea of tabu search is to define a neighborhood for a solution and let the algorithm explore the search space by moving to the best allowed neighbor solutions, while defining some forbidden or tabu moves in order to avoid cycling.

We define too basic moves for the STPRBH. The first is an ***add*** move. It consists of adding a path originating at a vertex of the current solution and ending at a profitable vertex. This is very similar to the basic step of the greedy algorithm presented in Section 6.3, since this move also makes use of the shortest path with hop constraints procedure and, therefore, maintains hop-feasibility during the whole search. In order to increase the flexibility of the algorithm, we let the ***add*** move violate the budget constraint. The idea is to design an algorithm capable of exploring infeasible parts of the search space in the hope of reaching feasible solutions that could not be reached otherwise. To encourage the

Algorithm 6.2 Destroy-and-repair algorithm for the STPRBH

T = solution of **Algorithm 6.1**;
best solution $\overline{T} = T$.
loop
  $\overline{r} = r(\overline{T})$.
  for all the edges (i,j) incident to a leaf vertex **do**
    store original cost: $\overline{c}_{ij} = c_{ij}$;
    $c_{ij} = \infty$;
    T = solution of **Algorithm 6.1** (with modified costs).
    if $r(T) > r(\overline{T})$ **then**
      $\overline{T} = T$.
    end if
    Restore original cost: $c_{ij} = \overline{c}_{ij}$.
  end for
  if $r(\overline{T}) = \overline{r}$ **then**
    END
  end if
end loop
$T = \overline{T}$;
return T

identification of feasible solutions during the search, a penalty is associated with trees that violate the budget. The idea of letting the tabu search explore infeasible areas of the search space has been used successfully in other combinatorial problems (see, e.g., Gendreau et al., 1994).

The second move is ***remove***, which consists of the deletion of a branch of the solution. It is the counterpart of the ***add*** move in the sense that the ***add*** move is driven by revenue collection, while the ***remove*** move is driven by budget feasibility. In the following, the two moves are described in detail. A third move, which is called ***destroy*** is also described. We have used this move only occasionally to help the search algorithm escape from local optima.

6.5.1 Add move

An ***add*** move is like a single step of the greedy algorithm described in Section 6.3: one adds the hop constrained shortest path connecting the best non-selected vertex to the current tree. This is done by setting the edge costs in the current solution to 0 and running the hop constrained shortest path procedure, which returns the hop constrained shortest paths originating at the root and ending at each vertex of the graph. Once a vertex has been added, it is declared tabu for a number of iterations. This way, the branch connecting the previous solution to the newly added vertex will not be deleted by the ***remove*** move until its tabu status has been lifted.

The tabu mechanism helps avoid cycling and forces the algorithm to explore new areas of the search space, differentiating the tabu search from the greedy algorithm presented earlier. Another strategy is used to increase the diversification of the solutions. At each iteration, one considers only a subset of randomly selected vertices: the algorithm arbitrarily chooses a subset of the possible vertices and adds the best one. Note that this is not done to save any computational time since the needed information was already available from the hop constrained shortest path procedure even for the discarded vertices, but as a diversification mechanism.

6.5.2 Remove move

The ***remove*** move consists of eliminating branches of the current solution. It was made necessary as a counterpart of the ***add*** move in order to help the algorithm identify budget-feasible solutions. We carefully select the branches to be eliminated in order to introduce only local perturbations in the current solution. This is done by connecting a vertex with degree larger or equal to three to a leaf of the current tree. We call the vertices with degree larger or equal to three *breaking points*, since it is at these vertices that the solution is cut. Figure 6.3 depicts an example of a tree solution. The breaking points are highlighted while the possible paths to be cut are shown in dotted lines.

6.5.3 Destroy move

A ***destroy*** move that erases a whole subtree of the current solution has also been developed. This move acts by randomly selecting a vertex in the current solution and deleting the whole subtree rooted at that vertex. This move was only used after the algorithm spent a certain number of iterations without improving the best known solution

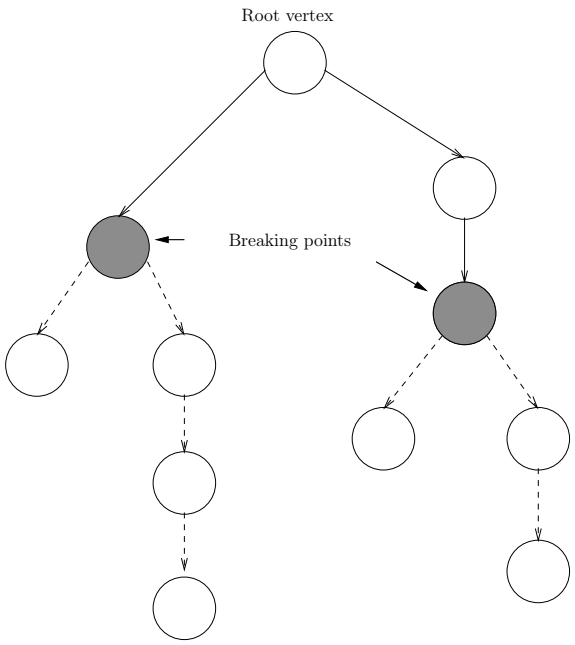


Figure 6.3: Breaking points and possible paths to be removed

value.

The three moves just described have been incorporated within the tabu search algorithm. The ***add*** and ***remove*** moves are evaluated together. First, the gain of all ***add*** moves is computed. The gain of one ***add*** move connecting the profitable vertex i to the solution is given by the value of the revenue of the added vertex, minus the difference between the penalties after and before the move:

$$\gamma^{+} = r_i - \pi(T') + \pi(T''),$$

where $\pi(T)$ is the penalty associated with a solution T, and T' and T'' are the solutions before and after the move, respectively. For a given solution T with total network cost $\sum_{(i,j)\in e(T)} c_{ij}$, the penalty is given by:

$$\pi(T) = \begin{cases} 0, & \text{if} \quad \sum_{(i,j)\in e(T)} c_{ij} \leq b \\ \phi \times \left[\sum_{(i,j)\in e(T)} c_{ij} - b\right], & \text{if} \quad \sum_{(i,j)\in e(T)} c_{ij} > b, \end{cases}$$

where ϕ is a penalty parameter updated dynamically during the search.

In a complementary manner, the ***remove*** move yields a gain equal to:

$$\gamma^{-} = -r_i - \pi(T') + \pi(T'').$$

At each iteration, the move with the best gain is performed and the penalty parameter ϕ is updated. If the new solution is budget feasible, ϕ is halved, thus encouraging the algorithm to look for budget-infeasible solutions. Similarly, if the new solution is budget-infeasible, ϕ is doubled.

Algorithm 6.3 presents the tabu search algorithm in detail. It stops after a preset number of iterations.

Algorithm 6.3 Tabu search algorithm for the STPRBH

T = solution of **Algorithm 6.1**;
$\overline{T} = T$;
$f = 1$.
loop
  Randomly select a subset of the profitable vertices not in the tree and compute gain;
  compute gain for remove moves;
  perform move with best gain and update T;
  make move tabu for 5 iterations.
  if solution is budget-feasible **then**
    $\phi = \phi/2$.
    if $r(T) > r(\overline{T})$ **then**
      $\overline{T} = T$.
    end if
  else
    $\phi = 2\phi$.
  end if
  if no improvement in more than 100 iterations **then**
    apply destroy move.
  end if
  if stopping criterion reached **then**
    END
  end if
end loop

6.6 Computational experiments

We have tested the algorithms just described on several instances obtained from the OR-Library (Beasley, 1990). The STP instances contain two set of vertices: *mandatory* vertices which must be present in the solution, and *Steiner* vertices which can be used if their inclusion in the solution reduces the total cost. We have adapted these instances for the STTPBH by using the mandatory vertices as profitable vertices with revenue given by a discrete uniform distribution on 1 and 100. The Steiner vertices were attributed a zero revenue. Different values for the budget and for the hop limit were attributed to each instance, thus creating several scenarios.

Tables 6.3 to 6.6 show the complete results for the mid-sized instances for which the exact methods presented in Chapter 5 were able to converge within a time limit of two hours. In the tables, we present the size of the instances, the parameters of each scenario and the optimal solution value. We also provide the gap and the computation time for each of the proposed methods: the greedy algorithm, the destroy-and-repair algorithm, and the tabu search algorithm. For this algorithm, we have run two tests. In the first one, the tabu search was run for 2000 iterations, while in the second it was allowed to run five times longer, for 10,000 iterations.

We use these instances in order to evaluate the gap between the solutions obtained by the heuristics and the optimal solutions. Table 6.1 summarizes the results obtained for the first group of instances. In this table, for each method, we report the mean and maximum gap, as well as the mean and maximum computation times.

Method	Gap	Max Gap	Seconds	Max seconds
Greedy	2.50 %	31.97 %	< 0.01	0.02
Destroy-and-repair	1.13 %	10.10 %	0.09	1.44
Tabu (2000)	0.52 ± 0.08 %	12.99 ± 7.55 %	0.38	0.90
Tabu (10,000)	0.24 ± 0.05 %	8.46 ± 8.01 %	1.80	4.21

Table 6.1: Summary of results for the mid-sized instances

We first observe that even the simplest of the three algorithms can solve a large number of instances to optimality. Indeed, the greedy algorithm finds an optimal solution for 78 out of the 144 instances. Note that we did not consider instances with a budget equal to the total network cost since these instances are easy and the greedy algorithm always finds an optimal solution. It is clear, however, that for some instances the greedy algorithm

can be trapped in a poor quality local optimum. In the worst of the cases, the algorithm generated a solution with an optimality gap almost equal to 32%. Finally, concerning the computational time, since the algorithm is based on quick shortest paths calculations, a maximum of 0.02 seconds was required to solve any instance.

Starting from the solution of the greedy algorithm, both the destroy-and-repair were able to reduce the optimality gaps. The first method reduced the mean gap from 2.50% to 1.13% and the maximum gap from 32% to 10.10%. The tabu search was able to further reduce these gaps. The mean gap (averaged over ten runs) was reduced to 0.52% when 2000 iterations were allowed, and to 0.24% when 10,000 iterations were allowed. The maximum gaps for these two cases were 12.99% and 8.46%.

To better evaluate the efficiency of the improvement routines, we consider a second group of larger instances with up to 500 vertices and 12,500 edges. For these instances, the exact algorithms described in Chapter 5 generally did not obtain solutions within the allowed time limit of two hours, often failing to even solve the linear relaxation at the root node. For this reason, we compare the results obtained by the destroy-and-repair and the tabu search methods to the results obtained by the greedy algorithm. The complete set of results is presented in Tables 6.7 to 6.12. In these tables we present the value of the solution found by the greedy algorithm and the percentage improvement obtained by the other methods. We also show the computational times for each instance. For the tabu search heuristic, the results are averaged over ten runs.

Table 6.2 presents a summary of the results presented in Tables 6.7 to 6.12. We can see that even for large instances, the greedy algorithm is very fast, taking on average 1.94 seconds, with a maximum of 31.78 seconds. Still concerning the times, it is interesting to see how the destroy-and-repair can easily become very computationally inefficient, taking more than two hours to solve some of the instances. The tabu search, even when run for 10,000 iterations, is still faster than the destroy-and-repair algorithm. Moreover, its computational time for difficult instances is much lower than that of the destroy-and-repair.

Method	Improvement	Max improvement	Seconds	Max seconds
Greedy	-	-	1.94	31.78
Destroy-and-repair	1.31 %	14.36 %	312.50	6493.60
Tabu (2000)	1.41 ± 0.06 %	20.80 ± 2.60 %	44.98	287.64
Tabu (10,000)	2.00 ± 0.04 %	23.05 ± 2.14 %	237.53	1668.6

Table 6.2: Summary of results for the larger instances

Concerning the improvements, the destroy-and-repair algorithm yields mean and maximum improvements of 1.31% and 14.36%, respectively. The tabu search can increase these percentages to 1.41% and 20.80% when run for 2000 iterations, and obtains considerably better solutions when 10,000 iterations are allowed. It then yields 2.00% average improvement in comparison to the greedy algorithm, and a maximum improvement of more than 23%.

6.7 Conclusions

We have proposed three heuristics for a modified version of the Steiner Tree Problem with revenues including budget and hop constraints. Chapter 5 has shown that exact algorithms based on existing formulations cannot be applied to large instances of this problem. The heuristics we have proposed are based on simple construction and tabu search mechanisms. The greedy algorithm is extremely fast but may be trapped in a local optimum due to its myopic nature. The destroy-and-repair method is a simple modification of the greedy algorithm which considerably improves its solutions, albeit at the cost of a large increase in computation time. Finally, the tabu search is capable of consistently identifying optimal solutions while maintaining the computational times at a reasonable level.

Instance						Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Optimum	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
MSteinb1	50	63	5	3	140	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.1	0.00 ± 0.00	0.44
			10	3	140	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.43
MSteinb2			5	3	182	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.43
			10	3	182	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.08	0.00 ± 0.00	0.38
MSteinb3			5	3	253	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.37
			10	3	253	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.07	0.00 ± 0.00	0.36
MSteinb4		100	5	3	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.06	0.00 ± 0.00	0.31
			10	3	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.06	0.00 ± 0.00	0.3
MSteinb5			5	3	588	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.07	0.00 ± 0.50	0.31
			10	3	565	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.68
MSteinb6			5	3	1035	6.28	0.00	2.90	0.02	0.53 ± 1.17	0.14	0.00 ± 0.86	0.68
			10	3	610	5.08	0.00	1.80	0.00	4.00 ± 1.45	0.14	0.43 ± 1.79	0.68
MSteinb7	75	94	5	3	87	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.16	0.00 ± 0.00	0.78
			10	3	87	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.78
MSteinb8			5	3	85	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.16	0.00 ± 0.00	0.78
			10	3	85	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.74
MSteinb9			5	3	596	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.16	0.00 ± 0.00	0.74
			10	3	483	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.18	0.00 ± 0.00	0.76
MSteinb10		150	5	3	319	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.44
			10	3	319	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.44
MSteinb11			5	3	316	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.1	0.00 ± 0.00	0.42
			10	3	305	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.1	0.00 ± 0.00	0.44
MSteinb12			5	3	1169	0.00	0.00	0.00	0.03	0.00 ± 0.00	0.09	0.00 ± 1.50	0.43
			10	3	1017	2.36	0.00	1.28	0.03	0.21 ± 0.55	0.09	0.00 ± 1.39	0.44
MSteinb13	100	125	5	3	147	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.14	0.00 ± 0.00	0.64
			10	3	147	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.65
MSteinb14			5	3	263	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.14	0.00 ± 0.00	0.66
			10	3	263	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.2	0.00 ± 0.00	0.95
MSteinb15			5	3	1061	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.2	0.00 ± 1.00	0.95
			10	3	830	2.17	0.00	2.17	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.95
MSteinb16		200	5	3	479	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.87
			10	3	479	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.18	0.00 ± 0.50	0.87
MSteinb17			5	3	254	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.2	0.00 ± 0.00	0.87
			10	3	254	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.22	0.00 ± 0.00	0.93
MSteinb18			5	3	1298	0.00	0.00	0.00	0.06	0.00 ± 0.00	0.21	0.00 ± 3.00	0.93
			10	3	1132	5.04	0.00	4.33	0.06	1.42 ± 0.98	0.22	0.20 ± 2.61	0.95

Table 6.3: Results for instances MSteinb, $h = 3$

Instance						Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Optimum	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
MSteinb1	50	63	5	6	403	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.74
			10	6	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.73
MSteinb2			5	6	491	2.65	0.00	0.00	0.01	2.65 ± 0.00	0.17	2.65 ± 1.31	0.72
			10	6	300	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.13	0.00 ± 0.00	0.59
MSteinb3			5	6	807	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.13	0.00 ± 0.50	0.59
			10	6	624	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.12	0.00 ± 0.00	0.6
MSteinb4		100	5	6	455	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.87
			10	6	447	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.87
MSteinb5			5	6	666	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.19	0.00 ± 1.00	0.89
			10	6	643	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.26	0.00 ± 0.50	1.2
MSteinb6			5	6	1257	2.78	0.00	1.59	0.08	0.60 ± 0.44	0.25	0.40 ± 3.76	1.19
			10	6	839	2.50	0.00	2.50	0.01	1.47 ± 0.63	0.26	0.86 ± 0.35	1.18
MSteinb7	75	94	5	6	627	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.15	0.00 ± 0.50	0.76
			10	6	432	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.17	0.00 ± 0.00	0.77
MSteinb8			5	6	535	5.79	0.00	0.00	0.02	0.64 ± 1.34	0.17	0.00 ± 0.87	0.76
			10	6	346	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.26	0.00 ± 0.00	1.22
MSteinb9			5	6	1343	8.71	0.00	4.69	0.06	0.10 ± 0.31	0.26	0.00 ± 2.94	1.22
			10	6	816	7.97	0.00	6.50	0.02	0.00 ± 0.00	0.27	0.00 ± 1.00	1.23
MSteinb10		150	5	6	702	0.00	0.00	0.00	0.04	0.00 ± 0.00	0.22	0.00 ± 2.00	1.07
			10	6	668	1.95	0.00	0.75	0.05	0.13 ± 0.22	0.23	0.00 ± 2.44	1.08
MSteinb11			5	6	893	0.00	0.00	0.00	0.07	0.00 ± 0.00	0.24	0.00 ± 3.50	1.09
			10	6	829	31.97	0.00	1.57	0.04	10.82 ± 10.10	0.21	6.16 ± 3.24	1
MSteinb12			5	6	1847	0.27	0.00	0.05	0.31	0.23 ± 0.09	0.21	0.05 ± 15.44	1.01
			10	6	1384	3.32	0.00	3.32	0.05	2.58 ± 0.86	0.21	0.87 ± 1.96	1.01
MSteinb13	100	125	5	6	674	0.15	0.00	0.00	0.07	0.00 ± 0.00	0.21	0.00 ± 3.50	0.93
			10	6	452	16.81	0.00	0.00	0.02	2.65 ± 0.00	0.2	1.59 ± 1.13	0.95
MSteinb14			5	6	977	0.72	0.00	0.00	0.06	0.72 ± 0.00	0.2	0.72 ± 2.78	0.96
			10	6	595	0.17	0.00	0.00	0.02	0.00 ± 0.00	0.28	0.00 ± 1.00	1.2
MSteinb15			5	6	1858	16.42	0.00	8.18	0.15	2.22 ± 1.94	0.26	0.12 ± 6.85	1.2
			10	6	1040	6.92	0.00	6.92	0.02	1.66 ± 1.78	0.28	0.50 ± 0.67	1.21
MSteinb16		200	5	6	840	0.00	0.00	0.00	0.09	0.00 ± 0.00	0.24	0.00 ± 4.50	1.2
			10	6	767	4.69	0.00	0.00	0.10	0.00 ± 0.00	0.25	0.00 ± 5.00	1.2
MSteinb17			5	6	1299	0.00	0.00	0.00	0.13	0.00 ± 0.00	0.26	0.00 ± 6.50	1.21
			10	6	1091	8.62	0.00	6.51	0.14	3.89 ± 0.97	0.27	2.87 ± 5.84	1.26
MSteinb18			5	6	2575	1.28	0.01	0.27	0.75	0.36 ± 0.20	0.28	0.17 ± 37.38	1.27
			10	6	1917	2.97	0.00	0.00	0.28	2.97 ± 0.00	0.27	1.78 ± 13.26	1.28

Table 6.4: Results for instances MSteinb, $h = 6$

Instance						Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Optimum	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
MSteinb1	50	63	5	9	431	0.46	0.00	0.46	0.01	0.00 ± 0.00	0.28	0.00 ± 0.50	1.3
			10	9	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.27	0.00 ± 0.00	1.29
MSteinb2			5	9	600	2.17	0.00	0.00	0.03	0.00 ± 0.00	0.28	0.00 ± 1.50	1.31
			10	9	300	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.25	0.00 ± 0.00	1.15
MSteinb3			5	9	924	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.25	0.00 ± 1.00	1.16
			10	9	649	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.25	0.00 ± 0.50	1.17
MSteinb4		100	5	9	455	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.33	0.00 ± 0.50	1.57
			10	9	455	1.76	0.00	1.54	0.02	1.54 ± 0.00	0.34	1.54 ± 0.87	1.64
MSteinb5			5	9	666	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.37	0.00 ± 1.00	1.64
			10	9	652	1.38	0.00	0.00	0.04	0.00 ± 0.00	0.38	0.00 ± 2.00	1.86
MSteinb6			5	9	1262	0.40	0.00	0.08	0.13	0.40 ± 0.00	0.39	0.30 ± 6.39	1.88
			10	9	903	0.89	0.00	0.89	0.02	0.89 ± 0.00	0.42	0.80 ± 0.82	1.86
MSteinb7	75	94	5	9	662	0.00	0.00	0.00	0.03	0.00 ± 0.00	0.41	0.00 ± 1.50	1.86
			10	9	432	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.41	0.00 ± 0.50	1.91
MSteinb8			5	9	761	0.79	0.00	0.53	0.10	0.37 ± 0.25	0.45	0.16 ± 4.87	1.93
			10	9	537	25.51	0.00	7.26	0.02	0.00 ± 0.00	0.38	0.00 ± 1.00	1.85
MSteinb9			5	9	1388	6.84	0.00	2.31	0.10	0.00 ± 0.00	0.39	0.00 ± 5.00	1.83
			10	9	816	3.43	0.00	0.37	0.03	0.00 ± 0.00	0.38	0.00 ± 1.50	1.82
MSteinb10		150	5	9	702	0.00	0.00	0.00	0.05	0.00 ± 0.00	0.44	0.00 ± 2.50	2.09
			10	9	694	1.87	0.00	1.87	0.04	0.45 ± 0.78	0.45	0.19 ± 1.78	2.17
MSteinb11			5	9	893	0.00	0.00	0.00	0.08	0.00 ± 0.00	0.46	0.00 ± 4.00	2.17
			10	9	855	3.04	0.00	2.34	0.09	2.11 ± 0.74	0.44	1.09 ± 3.89	1.88
MSteinb12			5	9	1866	0.91	0.00	0.54	0.39	0.80 ± 0.12	0.41	0.48 ± 19.27	1.91
			10	9	1401	7.14	0.00	2.64	0.12	1.38 ± 1.16	0.45	0.34 ± 5.54	1.92
MSteinb13	100	125	5	9	745	3.09	0.00	0.00	0.16	0.00 ± 0.00	0.48	0.00 ± 8.00	2.09
			10	9	465	9.89	0.00	6.24	0.03	0.00 ± 0.00	0.45	0.00 ± 1.50	2.15
MSteinb14			5	9	1033	5.32	0.00	0.39	0.09	3.00 ± 1.23	0.47	1.56 ± 3.62	2.2
			10	9	595	0.17	0.00	0.00	0.02	0.00 ± 0.00	0.52	0.00 ± 1.00	2.53
MSteinb15			5	9	1891	14.38	0.00	10.10	0.22	1.16 ± 2.05	0.53	0.00 ± 10.50	2.51
			10	9	1086	6.26	0.00	6.17	0.06	2.37 ± 1.93	0.52	0.00 ± 2.51	2.4
MSteinb16		200	5	9	840	0.00	0.00	0.00	0.14	0.00 ± 0.00	0.53	0.00 ± 7.00	2.46
			10	9	800	0.00	0.00	0.00	0.11	0.00 ± 0.00	0.54	0.00 ± 5.50	2.47
MSteinb17			5	9	1299	0.00	0.00	0.00	0.19	0.00 ± 0.00	0.51	0.00 ± 9.50	2.5
			10	9	1178	5.18	0.00	1.95	0.20	1.64 ± 1.04	0.52	0.42 ± 9.50	2.51
MSteinb18			5	9	2585	0.39	0.01	0.19	1.17	0.14 ± 0.09	0.59	0.04 ± 58.46	2.5
			10	9	1997	9.16	0.00	6.36	0.35	1.37 ± 0.78	0.51	0.33 ± 17.09	2.54

Table 6.5: Results for instances MSteinb, $h = 9$

Instance						Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Optimum	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
MSteinb1	50	63	5	12	431	0.46	0.00	0.46	0.01	0.00 ± 0.00	0.26	0.00 ± 0.50	1.29
			10	12	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.28	0.00 ± 0.00	1.31
MSteinb2			5	12	600	2.17	0.00	0.00	0.03	0.00 ± 0.00	0.3	0.00 ± 1.50	1.3
			10	12	300	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.29	0.00 ± 0.00	1.34
MSteinb3			5	12	931	2.58	0.00	0.75	0.06	0.75 ± 0.00	0.28	0.60 ± 2.79	1.37
			10	12	649	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.28	0.00 ± 0.50	1.28
MSteinb4		100	5	12	455	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.27	0.00 ± 1.00	1.25
			10	12	455	1.76	0.00	1.54	0.03	1.54 ± 0.00	0.28	1.54 ± 1.22	1.33
MSteinb5			5	12	666	0.00	0.00	0.00	0.03	0.00 ± 0.00	0.29	0.00 ± 1.50	1.36
			10	12	652	1.38	0.00	0.00	0.04	0.00 ± 0.00	0.32	0.00 ± 2.00	1.51
MSteinb6			5	12	1262	0.00	0.00	0.00	0.08	0.00 ± 0.00	0.32	0.00 ± 4.00	1.51
			10	12	903	0.89	0.00	0.89	0.02	0.89 ± 0.00	0.32	0.89 ± 0.82	1.51
MSteinb7	75	94	5	12	662	0.00	0.00	0.00	0.04	0.00 ± 0.00	0.32	0.00 ± 2.00	1.49
			10	12	432	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.32	0.00 ± 0.50	1.48
MSteinb8			5	12	832	0.72	0.00	0.72	0.06	0.72 ± 0.00	0.33	0.65 ± 2.79	1.48
			10	12	537	25.51	0.00	7.26	0.03	0.00 ± 0.00	0.35	0.00 ± 1.50	1.59
MSteinb9			5	12	1388	6.84	0.00	2.31	0.12	0.00 ± 0.00	0.34	0.00 ± 6.00	1.59
			10	12	816	3.43	0.00	0.37	0.04	0.00 ± 0.00	0.35	0.00 ± 2.00	1.58
MSteinb10		150	5	12	702	0.00	0.00	0.00	0.06	0.00 ± 0.00	0.45	0.00 ± 3.00	1.94
			10	12	694	1.87	0.00	1.87	0.05	0.40 ± 0.62	0.41	0.00 ± 2.34	1.95
MSteinb11			5	12	893	0.00	0.00	0.00	0.09	0.00 ± 0.00	0.41	0.00 ± 4.50	1.97
			10	12	855	3.04	0.00	2.34	0.11	2.34 ± 0.00	0.43	0.47 ± 5.13	2.07
MSteinb12			5	12	1866	0.27	0.01	0.21	0.53	0.17 ± 0.12	0.44	0.00 ± 26.45	2.09
			10	12	1401	3.28	0.00	2.64	0.16	1.16 ± 0.30	0.45	0.93 ± 7.61	2.11
MSteinb13	100	125	5	12	745	4.43	0.00	0.00	0.18	0.00 ± 0.00	0.45	0.00 ± 9.00	2.03
			10	12	465	9.89	0.00	6.24	0.04	0.00 ± 0.00	0.43	0.00 ± 2.00	2.05
MSteinb14			5	12	1038	1.54	0.00	0.00	0.13	1.01 ± 0.56	0.42	0.59 ± 6.14	2.06
			10	12	595	0.17	0.00	0.00	0.03	0.00 ± 0.00	0.48	0.00 ± 1.50	2.34
MSteinb15			5	12	1891	6.82	0.00	6.82	0.15	1.81 ± 1.63	0.47	0.01 ± 6.97	2.38
			10	12	1109	8.21	0.00	8.03	0.07	2.52 ± 2.16	0.5	1.38 ± 2.54	2.37
MSteinb16		200	5	12	840	0.00	0.00	0.00	0.17	0.00 ± 0.00	0.52	0.00 ± 8.50	2.37
			10	12	800	0.00	0.00	0.00	0.13	0.00 ± 0.00	0.49	0.00 ± 6.50	2.4
MSteinb17			5	12	1299	0.00	0.01	0.00	0.26	0.00 ± 0.00	0.51	0.00 ± 13.00	2.41
			10	12	1225	7.43	0.00	3.59	0.29	2.87 ± 1.76	0.48	1.22 ± 13.54	2.32
MSteinb18			5	12	2585	0.39	0.02	0.19	1.44	0.14 ± 0.09	0.51	0.02 ± 71.96	2.33
			10	12	1997	9.16	0.01	8.81	0.43	1.10 ± 0.92	0.51	0.27 ± 21.12	2.31

Table 6.6: Results for instances MSteinb, $h = 12$

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc1_10	500	625	10	5	8	0.00	0.00	0	0.00 ± 0.00	0.96	0.00 ± 0.00	4.51
steinc1_100	500	625	10	5	71	0.00	0.00	0	0.00 ± 0.00	0.93	0.00 ± 0.00	4.56
steinc1_10	500	625	30	5	8	0.00	0.00	0	0.00 ± 0.00	0.93	0.00 ± 0.00	4.56
steinc1_100	500	625	30	5	71	0.00	0.00	0	0.00 ± 0.00	0.91	0.00 ± 0.00	4.56
steinc2_10	500	625	10	5	32	0.00	0.00	0.02	0.00 ± 0.00	0.97	0.00 ± 0.00	4.84
steinc2_100	500	625	10	5	328	0.00	0.00	0.02	0.00 ± 0.00	0.97	0.00 ± 0.00	4.76
steinc2_10	500	625	30	5	32	0.00	0.00	0.02	0.00 ± 0.00	0.97	0.00 ± 0.00	4.88
steinc2_100	500	625	30	5	328	0.00	0.00	0.02	0.00 ± 0.00	0.97	0.00 ± 0.00	4.81
steinc3_10	500	625	10	5	151	0.01	0.00	0.55	0.00 ± 0.00	1.14	0.00 ± 0.00	5.71
steinc3_100	500	625	10	5	1519	0.01	0.00	0.54	0.00 ± 0.00	1.16	0.00 ± 0.00	5.71
steinc3_10	500	625	30	5	93	0.01	1.08	0.23	1.72 ± 0.75	1.14	2.15 ± 0.00	5.71
steinc3_100	500	625	30	5	962	0.01	0.62	0.22	0.12 ± 0.26	1.16	0.14 ± 0.28	5.73
steinc4_10	500	625	10	5	115	0.01	0.00	0.21	0.00 ± 0.00	0.91	0.00 ± 0.00	4.51
steinc4_100	500	625	10	5	1148	0.01	0.00	0.2	0.00 ± 0.00	0.91	0.00 ± 0.00	4.55
steinc4_10	500	625	30	5	84	0.00	0.00	0.07	0.00 ± 0.00	0.91	0.00 ± 0.00	4.55
steinc4_100	500	625	30	5	854	0.00	0.00	0.07	0.00 ± 0.00	0.91	0.00 ± 0.00	4.55
steinc5_10	500	625	10	5	258	0.02	0.00	0.9	0.00 ± 0.00	0.99	0.00 ± 0.00	4.79
steinc5_100	500	625	10	5	2600	0.02	0.00	0.9	0.00 ± 0.00	1.05	0.00 ± 0.00	4.86
steinc5_10	500	625	30	5	141	0.01	3.55	0.29	3.83 ± 1.17	0.99	5.91 ± 0.87	4.82
steinc5_100	500	625	30	5	1459	0.01	0.00	0.15	3.10 ± 1.01	0.97	4.45 ± 0.73	4.88
steinc6_10	500	1000	20	5	27	0.00	0.00	0.07	0.00 ± 0.00	2.01	0.00 ± 0.00	9.94
steinc6_100	500	1000	20	5	274	0.01	0.00	0.07	0.00 ± 0.00	2.12	0.00 ± 0.00	10.02
steinc6_10	500	1000	50	5	27	0.00	0.00	0.07	0.00 ± 0.00	2.05	0.00 ± 0.00	9.95
steinc6_100	500	1000	50	5	274	0.00	0.00	0.06	0.00 ± 0.00	2.1	0.00 ± 0.00	10.02
steinc7_10	500	1000	20	5	49	0.01	0.00	0.17	0.00 ± 0.00	1.99	0.00 ± 0.00	9.88
steinc7_100	500	1000	20	5	503	0.01	0.00	0.16	0.00 ± 0.00	2.13	0.00 ± 0.00	9.94
steinc7_10	500	1000	50	5	46	0.01	6.52	0.25	0.00 ± 0.00	2.01	0.00 ± 0.00	9.86
steinc7_100	500	1000	50	5	471	0.01	6.79	0.25	0.00 ± 0.00	2.02	0.00 ± 0.00	9.92
steinc8_10	500	1000	20	5	201	0.02	2.99	1.81	12.49 ± 0.76	1.94	13.21 ± 0.26	8.96
steinc8_100	500	1000	20	5	2102	0.02	7.52	1.76	11.43 ± 1.04	1.85	12.40 ± 0.23	9.63
steinc8_10	500	1000	50	5	104	0.01	0.00	0.16	8.08 ± 3.08	1.86	9.94 ± 0.96	8.93
steinc8_100	500	1000	50	5	1067	0.01	7.87	0.32	9.45 ± 3.32	1.85	12.78 ± 1.49	9.1
steinc9_10	500	1000	20	5	270	0.03	1.48	3.1	2.48 ± 0.68	2.19	3.83 ± 0.89	10.76
steinc9_100	500	1000	20	5	2825	0.03	1.38	3.07	2.80 ± 1.00	2.25	3.76 ± 0.78	10.66
steinc9_10	500	1000	50	5	143	0.02	0.00	0.36	0.42 ± 0.94	2.21	1.17 ± 0.70	10.53
steinc9_100	500	1000	50	5	1492	0.01	1.47	0.69	0.40 ± 0.71	2.17	1.31 ± 1.38	11.49
steinc10_10	500	1000	20	5	331	0.03	2.72	2.83	9.85 ± 2.17	1.97	11.61 ± 0.55	9.4
steinc10_100	500	1000	20	5	3216	0.03	9.76	2.58	16.84 ± 1.64	2.12	19.17 ± 0.93	9.36
steinc10_10	500	1000	50	5	151	0.02	2.65	0.53	12.65 ± 1.51	1.91	14.13 ± 1.24	9.28
steinc10_100	500	1000	50	5	1483	0.01	1.96	0.44	20.54 ± 3.20	1.92	22.98 ± 2.28	9.3

Table 6.7: Results for instances SteinC, $h = 5$

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc11_10	500	2500	20	5	27	0.01	0.00	0.07	0.00 ± 0.00	3.51	0.00 ± 0.00	16.99
steinc11_100	500	2500	20	5	274	0.00	0.00	0.07	0.00 ± 0.00	3.46	0.00 ± 0.00	17.11
steinc11_10	500	2500	100	5	27	0.01	0.00	0.07	0.00 ± 0.00	3.46	0.00 ± 0.00	17.02
steinc11_100	500	2500	100	5	274	0.01	0.00	0.07	0.00 ± 0.00	3.5	0.00 ± 0.00	16.98
steinc12_10	500	2500	20	5	59	0.02	0.00	0.46	0.00 ± 0.00	3.5	0.00 ± 0.00	17.01
steinc12_100	500	2500	20	5	604	0.02	0.00	0.45	0.00 ± 0.00	3.54	0.00 ± 0.00	17.09
steinc12_10	500	2500	100	5	59	0.02	0.00	0.45	0.00 ± 0.00	3.62	0.00 ± 0.00	16.95
steinc12_100	500	2500	100	5	604	0.02	0.00	0.45	0.00 ± 0.00	3.53	0.00 ± 0.00	16.97
steinc13_10	500	2500	20	5	439	0.14	0.00	13.42	0.00 ± 0.00	3.69	0.00 ± 0.00	17.62
steinc13_100	500	2500	20	5	4463	0.14	0.00	13.45	0.00 ± 0.00	3.69	0.00 ± 0.00	17.35
steinc13_10	500	2500	100	5	236	0.05	2.54	3.66	1.86 ± 0.83	3.65	3.34 ± 0.68	17.08
steinc13_100	500	2500	100	5	2421	0.05	3.55	3.4	3.12 ± 1.19	3.55	4.24 ± 0.84	18.41
steinc14_10	500	2500	20	5	648	0.2	0.00	27.41	0.00 ± 0.00	3.8	0.00 ± 0.00	17.56
steinc14_100	500	2500	20	5	6566	0.21	0.00	28.21	0.00 ± 0.00	3.85	0.00 ± 0.00	17.47
steinc14_10	500	2500	100	5	339	0.07	1.18	7.3	0.00 ± 0.00	3.67	0.13 ± 0.39	17.85
steinc14_100	500	2500	100	5	3364	0.07	3.57	6.88	0.57 ± 0.96	3.6	2.85 ± 1.28	17.02
steinc15_10	500	2500	20	5	1162	0.31	0.95	108.88	1.77 ± 0.77	3.96	3.75 ± 0.46	17.81
steinc15_100	500	2500	20	5	11963	0.32	0.96	114.13	1.25 ± 0.71	3.9	2.33 ± 0.53	18.01
steinc15_10	500	2500	100	5	397	0.09	0.76	6.77	5.49 ± 2.06	3.59	8.06 ± 1.38	17.31
steinc15_100	500	2500	100	5	4108	0.08	1.75	7.08	4.11 ± 1.97	3.71	4.80 ± 1.78	17.25
steinc16_10	500	12500	100	5	27	0.15	0.00	1.31	0.00 ± 0.00	52.27	0.00 ± 0.00	256.96
steinc16_100	500	12500	100	5	274	0.14	0.00	1.32	0.00 ± 0.00	52.19	0.00 ± 0.00	257.23
steinc16_10	500	12500	200	5	27	0.14	0.00	1.31	0.00 ± 0.00	52.2	0.00 ± 0.00	256.91
steinc16_100	500	12500	200	5	274	0.14	0.00	1.32	0.00 ± 0.00	52.16	0.00 ± 0.00	256.82
steinc17_10	500	12500	100	5	59	0.3	0.00	4.78	0.00 ± 0.00	52.5	0.00 ± 0.00	258.42
steinc17_100	500	12500	100	5	604	0.3	0.00	5.32	0.00 ± 0.00	52.27	0.00 ± 0.00	257.98
steinc17_10	500	12500	200	5	59	0.3	0.00	4.82	0.00 ± 0.00	52.21	0.00 ± 0.00	258.26
steinc17_100	500	12500	200	5	604	0.3	0.00	5.36	0.00 ± 0.00	52.43	0.00 ± 0.00	258.22
steinc18_10	500	12500	100	5	439	2.54	0.00	222.81	0.00 ± 0.00	55.56	0.00 ± 0.00	264.71
steinc18_100	500	12500	100	5	4463	2.55	0.00	211.32	0.00 ± 0.00	55.56	0.00 ± 0.00	264.94
steinc18_10	500	12500	200	5	439	2.55	0.00	223.25	0.00 ± 0.00	55.7	0.00 ± 0.00	264.92
steinc18_100	500	12500	200	5	4463	2.57	0.00	211.84	0.00 ± 0.00	55.88	0.00 ± 0.00	264.46
steinc19_10	500	12500	100	5	648	3.6	0.00	420.42	0.00 ± 0.00	56.86	0.00 ± 0.00	267.25
steinc19_100	500	12500	100	5	6566	3.68	0.00	444.49	0.00 ± 0.00	57.11	0.00 ± 0.00	267.64
steinc19_10	500	12500	200	5	648	3.61	0.00	419.41	0.00 ± 0.00	57.08	0.00 ± 0.00	267.43
steinc19_100	500	12500	200	5	6566	3.72	0.00	444.14	0.00 ± 0.00	56.94	0.00 ± 0.00	267.87
steinc20_10	500	12500	100	5	1248	7.33	0.00	1558.5	0.00 ± 0.00	61.81	0.00 ± 0.00	327.03
steinc20_100	500	12500	100	5	12533	7.29	0.00	1536.2	0.00 ± 0.00	61.78	0.00 ± 0.00	276.01
steinc20_10	500	12500	200	5	1235	6.93	1.05	2824.3	0.68 ± 0.36	61.69	1.03 ± 0.05	276.89
steinc20_100	500	12500	200	5	12529	7.21	0.03	2956.9	0.02 ± 0.01	61.83	0.03 ± 0.00	277.55

Table 6.8: Results for instances SteinC, $h = 5$ (continued)

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc1_10	500	625	10	15	27	0.01	0.00	0.2	0.00 ± 0.00	4.34	0.00 ± 0.00	21.25
steinc1_100	500	625	10	15	274	0.01	0.00	0.19	0.00 ± 0.00	4.32	0.00 ± 0.00	21.31
steinc1_10	500	625	30	15	27	0.01	0.00	0.19	0.00 ± 0.00	4.26	0.00 ± 0.00	21.18
steinc1_100	500	625	30	15	274	0.01	0.00	0.19	0.00 ± 0.00	4.33	0.00 ± 0.00	21.12
steinc2_10	500	625	10	15	59	0.02	0.00	0.47	0.00 ± 0.00	4.42	0.00 ± 0.00	21.02
steinc2_100	500	625	10	15	604	0.02	0.00	0.48	0.00 ± 0.00	4.38	0.00 ± 0.00	21.06
steinc2_10	500	625	30	15	50	0.01	6.00	0.47	2.00 ± 0.00	4.28	2.00 ± 0.00	21.1
steinc2_100	500	625	30	15	533	0.01	2.44	0.51	0.00 ± 0.00	4.39	0.00 ± 0.00	21.22
steinc3_10	500	625	10	15	270	0.06	2.96	6.88	2.63 ± 1.40	4.66	4.12 ± 0.51	22.46
steinc3_100	500	625	10	15	2822	0.06	0.85	6.82	2.51 ± 0.94	4.67	2.83 ± 1.14	22.35
steinc3_10	500	625	30	15	118	0.03	3.39	1	3.98 ± 3.04	4.42	6.21 ± 1.80	21.34
steinc3_100	500	625	30	15	1155	0.02	8.14	0.96	8.44 ± 2.65	4.45	12.06 ± 1.94	22.15
steinc4_10	500	625	10	15	308	0.08	2.27	10.08	3.18 ± 1.11	4.57	4.40 ± 0.78	21.93
steinc4_100	500	625	10	15	3219	0.08	0.34	9.91	0.69 ± 0.91	4.8	2.18 ± 0.88	22.34
steinc4_10	500	625	30	15	122	0.03	0.00	0.7	2.30 ± 1.80	4.5	5.37 ± 0.93	22.36
steinc4_100	500	625	30	15	1264	0.03	0.00	0.68	2.71 ± 1.74	4.5	6.09 ± 0.91	22.01
steinc5_10	500	625	10	15	419	0.09	1.67	11.35	3.77 ± 1.72	4.62	6.50 ± 1.60	21.89
steinc5_100	500	625	10	15	4524	0.1	1.19	12.05	0.03 ± 0.09	4.53	0.54 ± 0.77	21.39
steinc5_10	500	625	30	15	143	0.03	11.89	1.32	11.26 ± 2.50	4.39	13.75 ± 1.11	21.51
steinc5_100	500	625	30	15	1481	0.03	10.53	1.27	8.71 ± 2.81	4.45	12.04 ± 1.83	21.3
steinc6_10	500	1000	20	15	27	0.01	0.00	0.18	0.00 ± 0.00	5.25	0.00 ± 0.00	25.74
steinc6_100	500	1000	20	15	274	0.01	0.00	0.18	0.00 ± 0.00	5.27	0.00 ± 0.00	25.76
steinc6_10	500	1000	50	15	27	0.01	0.00	0.18	0.00 ± 0.00	5.21	0.00 ± 0.00	25.73
steinc6_100	500	1000	50	15	274	0.01	0.00	0.18	0.00 ± 0.00	5.13	0.00 ± 0.00	25.71
steinc7_10	500	1000	20	15	59	0.02	0.00	0.54	0.00 ± 0.00	5.28	0.00 ± 0.00	25.66
steinc7_100	500	1000	20	15	604	0.03	0.00	0.52	0.00 ± 0.00	5.22	0.00 ± 0.00	25.81
steinc7_10	500	1000	50	15	59	0.02	0.00	0.5	0.00 ± 0.00	5.34	0.00 ± 0.00	25.69
steinc7_100	500	1000	50	15	604	0.02	0.00	0.49	0.00 ± 0.00	5.22	0.00 ± 0.00	25.84
steinc8_10	500	1000	20	15	293	0.1	3.41	13.03	3.41 ± 1.70	5.75	7.58 ± 1.06	28.24
steinc8_100	500	1000	20	15	2988	0.09	9.40	12.66	6.75 ± 1.56	5.9	9.88 ± 0.75	27.78
steinc8_10	500	1000	50	15	142	0.05	6.34	2.61	9.58 ± 3.31	5.7	14.40 ± 2.20	27.28
steinc8_100	500	1000	50	15	1483	0.04	14.36	2.58	11.10 ± 1.80	5.73	13.87 ± 1.05	27.39
steinc9_10	500	1000	20	15	342	0.11	3.22	15.55	1.05 ± 0.77	5.81	1.92 ± 1.38	26.81
steinc9_100	500	1000	20	15	3568	0.11	2.05	15.27	0.16 ± 0.38	5.69	0.78 ± 0.97	26.62
steinc9_10	500	1000	50	15	160	0.05	7.50	3.23	1.44 ± 1.59	5.39	3.89 ± 1.56	26.43
steinc9_100	500	1000	50	15	1626	0.05	3.01	2.92	4.43 ± 2.69	5.29	5.34 ± 1.13	26.4
steinc10_10	500	1000	20	15	488	0.14	3.48	17.91	1.93 ± 1.46	5.67	2.50 ± 1.76	27.81
steinc10_100	500	1000	20	15	5095	0.14	1.33	18.22	1.40 ± 1.18	6.23	1.47 ± 1.62	27.67
steinc10_10	500	1000	50	15	209	0.06	0.48	3.48	2.44 ± 3.00	5.54	5.42 ± 2.53	27.23
steinc10_100	500	1000	50	15	2356	0.07	2.50	3.95	0.08 ± 0.27	5.44	0.00 ± 0.00	27.18

Table 6.9: Results for instances SteinC, $h = 15$

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc11_10	500	2500	20	15	27	0.03	0.00	0.26	0.00 ± 0.00	9.48	0.00 ± 0.00	42.95
steinc11_100	500	2500	20	15	274	0.03	0.00	0.26	0.00 ± 0.00	9.12	0.00 ± 0.00	42.69
steinc11_10	500	2500	100	15	27	0.03	0.00	0.26	0.00 ± 0.00	9.17	0.00 ± 0.00	42.76
steinc11_100	500	2500	100	15	274	0.02	0.00	0.26	0.00 ± 0.00	9.6	0.00 ± 0.00	43
steinc12_10	500	2500	20	15	59	0.05	0.00	1.01	0.00 ± 0.00	9.77	0.00 ± 0.00	43.79
steinc12_100	500	2500	20	15	604	0.04	0.00	1.06	0.00 ± 0.00	9.81	0.00 ± 0.00	43.91
steinc12_10	500	2500	100	15	59	0.05	0.00	1.01	0.00 ± 0.00	9.7	0.00 ± 0.00	43.74
steinc12_100	500	2500	100	15	604	0.05	0.00	1.06	0.00 ± 0.00	9.24	0.00 ± 0.00	43.63
steinc13_10	500	2500	20	15	439	0.39	0.00	40.71	0.00 ± 0.00	10.9	0.00 ± 0.00	45.21
steinc13_100	500	2500	20	15	4463	0.39	0.00	40.42	0.00 ± 0.00	10.07	0.00 ± 0.00	45.93
steinc13_10	500	2500	100	15	277	0.18	9.03	17.22	4.95 ± 1.76	9.97	7.94 ± 0.81	44.85
steinc13_100	500	2500	100	15	2860	0.17	7.13	15.95	4.62 ± 1.27	9.65	6.97 ± 1.79	44.71
steinc14_10	500	2500	20	15	648	0.54	0.00	75.51	0.00 ± 0.00	10.27	0.00 ± 0.00	43.8
steinc14_100	500	2500	20	15	6566	0.53	0.00	74.82	0.00 ± 0.00	10.14	0.00 ± 0.00	44.13
steinc14_10	500	2500	100	15	369	0.21	1.90	24.28	0.05 ± 0.17	9.78	0.15 ± 0.24	42.83
steinc14_100	500	2500	100	15	3823	0.22	0.60	23.73	0.51 ± 0.77	9.79	1.19 ± 1.58	42.86
steinc15_10	500	2500	20	15	1248	1.07	0.00	245.52	0.00 ± 0.00	10.77	0.00 ± 0.00	46.03
steinc15_100	500	2500	20	15	12533	1.11	0.00	251.01	0.00 ± 0.00	10.2	0.00 ± 0.00	45.68
steinc15_10	500	2500	100	15	493	0.26	4.46	32.16	0.75 ± 0.70	9.72	0.95 ± 0.76	44.4
steinc15_100	500	2500	100	15	5153	0.27	3.92	32.93	0.17 ± 0.28	9.7	0.04 ± 0.07	44.74
steinc16_10	500	12500	100	15	27	0.39	0.00	3.53	0.00 ± 0.00	153.32	0.00 ± 0.00	746.06
steinc16_100	500	12500	100	15	274	0.39	0.00	3.53	0.00 ± 0.00	153.29	0.00 ± 0.00	745.24
steinc16_10	500	12500	200	15	27	0.4	0.00	3.49	0.00 ± 0.00	153.61	0.00 ± 0.00	744.9
steinc16_100	500	12500	200	15	274	0.39	0.00	3.5	0.00 ± 0.00	153.38	0.00 ± 0.00	743.76
steinc17_10	500	12500	100	15	59	0.83	0.00	13.01	0.00 ± 0.00	154.55	0.00 ± 0.00	1001.7
steinc17_100	500	12500	100	15	604	0.82	0.00	12.55	0.00 ± 0.00	154.23	0.00 ± 0.00	746.03
steinc17_10	500	12500	200	15	59	0.81	0.00	12.98	0.00 ± 0.00	154.58	0.00 ± 0.00	746.66
steinc17_100	500	12500	200	15	604	0.81	0.00	12.6	0.00 ± 0.00	154.47	0.00 ± 0.00	747.05
steinc18_10	500	12500	100	15	439	6.88	0.00	583.99	0.00 ± 0.00	161.32	0.00 ± 0.00	758.87
steinc18_100	500	12500	100	15	4463	6.72	0.00	560.89	0.00 ± 0.00	161.27	0.00 ± 0.00	757.6
steinc18_10	500	12500	200	15	439	6.92	0.00	583.76	0.00 ± 0.00	161.61	0.00 ± 0.00	756.99
steinc18_100	500	12500	200	15	4463	6.67	0.00	561.07	0.00 ± 0.00	161.38	0.00 ± 0.00	757.35
steinc19_10	500	12500	100	15	648	9.76	0.00	1109.5	0.00 ± 0.00	164.03	0.00 ± 0.00	760.96
steinc19_100	500	12500	100	15	6566	9.8	0.00	1193.2	0.00 ± 0.00	164.23	0.00 ± 0.00	761.01
steinc19_10	500	12500	200	15	648	9.76	0.00	1107.7	0.00 ± 0.00	164.33	0.00 ± 0.00	760.13
steinc19_100	500	12500	200	15	6566	9.79	0.00	1193.5	0.00 ± 0.00	164.44	0.00 ± 0.00	762.17
steinc20_10	500	12500	100	15	1248	19.55	0.00	4038.1	0.00 ± 0.00	175.62	0.00 ± 0.00	778.16
steinc20_100	500	12500	100	15	12533	19.87	0.00	4104.5	0.00 ± 0.00	175.69	0.00 ± 0.00	777.73
steinc20_10	500	12500	200	15	1248	19.56	0.00	4037.2	0.00 ± 0.00	175.56	0.00 ± 0.00	779.19
steinc20_100	500	12500	200	15	12533	19.84	0.00	4104.9	0.00 ± 0.00	175.9	0.00 ± 0.00	777.67

Table 6.10: Results for instances SteinC, $h = 15$ (continued)

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc1_10	500	625	10	25	27	0.02	0.00	0.31	0.00 ± 0.00	6.73	0.00 ± 0.00	32.46
steinc1_100	500	625	10	25	274	0.01	0.00	0.3	0.00 ± 0.00	6.57	0.00 ± 0.00	32.61
steinc1_10	500	625	30	25	27	0.01	0.00	0.31	0.00 ± 0.00	6.98	0.00 ± 0.00	32.31
steinc1_100	500	625	30	25	274	0.01	0.00	0.31	0.00 ± 0.00	6.85	0.00 ± 0.00	32.27
steinc2_10	500	625	10	25	59	0.03	0.00	0.74	0.00 ± 0.00	6.86	0.00 ± 0.00	32.64
steinc2_100	500	625	10	25	604	0.03	0.00	0.74	0.00 ± 0.00	6.46	0.00 ± 0.00	32.36
steinc2_10	500	625	30	25	50	0.02	6.00	0.75	2.00 ± 0.00	6.93	2.00 ± 0.00	32.39
steinc2_100	500	625	30	25	533	0.02	2.44	0.82	0.00 ± 0.00	6.88	0.00 ± 0.00	32.4
steinc3_10	500	625	10	25	270	0.1	2.96	10.95	2.74 ± 0.74	7.22	4.07 ± 0.56	34.25
steinc3_100	500	625	10	25	2822	0.1	0.85	10.7	2.83 ± 0.77	7.13	2.97 ± 0.65	33.46
steinc3_10	500	625	30	25	118	0.04	3.39	1.62	5.42 ± 2.34	6.87	7.91 ± 1.12	33.58
steinc3_100	500	625	30	25	1155	0.04	8.14	1.53	9.44 ± 3.18	7.03	11.50 ± 2.66	33.32
steinc4_10	500	625	10	25	314	0.12	2.23	16.48	2.01 ± 1.12	7.14	4.60 ± 0.97	34.48
steinc4_100	500	625	10	25	3219	0.12	1.21	23.99	1.05 ± 1.62	7.07	3.03 ± 1.40	34.59
steinc4_10	500	625	30	25	122	0.04	0.00	1.13	3.52 ± 1.45	6.91	6.47 ± 1.26	33.76
steinc4_100	500	625	30	25	1264	0.05	0.00	1.08	3.14 ± 2.28	6.72	5.12 ± 1.63	34.18
steinc5_10	500	625	10	25	426	0.16	2.58	20.3	2.11 ± 1.83	7	5.09 ± 1.54	33.7
steinc5_100	500	625	10	25	4310	0.15	5.15	18.84	3.08 ± 1.26	7.29	4.63 ± 1.23	34.18
steinc5_10	500	625	30	25	143	0.05	11.89	2.11	13.08 ± 2.87	6.5	14.53 ± 1.97	33.48
steinc5_100	500	625	30	25	1481	0.05	10.53	2.01	9.78 ± 1.47	6.68	13.23 ± 2.29	33.69
steinc6_10	500	1000	20	25	27	0.02	0.00	0.28	0.00 ± 0.00	7.92	0.00 ± 0.00	39.48
steinc6_100	500	1000	20	25	274	0.02	0.00	0.28	0.00 ± 0.00	8	0.00 ± 0.00	39.46
steinc6_10	500	1000	50	25	27	0.02	0.00	0.28	0.00 ± 0.00	7.91	0.00 ± 0.00	39.41
steinc6_100	500	1000	50	25	274	0.02	0.00	0.28	0.00 ± 0.00	7.95	0.00 ± 0.00	39.37
steinc7_10	500	1000	20	25	59	0.04	0.00	0.83	0.00 ± 0.00	7.91	0.00 ± 0.00	39.13
steinc7_100	500	1000	20	25	604	0.04	0.00	0.81	0.00 ± 0.00	7.93	0.00 ± 0.00	38.9
steinc7_10	500	1000	50	25	59	0.04	0.00	0.78	0.00 ± 0.00	7.97	0.00 ± 0.00	38.95
steinc7_100	500	1000	50	25	604	0.04	0.00	0.75	0.00 ± 0.00	7.92	0.00 ± 0.00	38.89
steinc8_10	500	1000	20	25	294	0.17	5.78	22.23	4.22 ± 1.21	8.98	7.52 ± 0.98	44.39
steinc8_100	500	1000	20	25	3302	0.19	0.24	23.11	0.55 ± 0.48	9.15	0.59 ± 0.55	43.19
steinc8_10	500	1000	50	25	138	0.07	9.42	4.37	13.12 ± 2.28	8.72	17.71 ± 1.89	42.69
steinc8_100	500	1000	50	25	1735	0.08	0.00	2.45	0.09 ± 0.27	8.81	0.44 ± 0.69	42.6
steinc9_10	500	1000	20	25	353	0.19	0.00	12.94	0.00 ± 0.00	8.63	0.00 ± 0.00	42.37
steinc9_100	500	1000	20	25	3556	0.17	1.27	23.64	0.97 ± 1.49	8.51	2.18 ± 1.10	42.35
steinc9_10	500	1000	50	25	160	0.07	5.00	5.07	2.63 ± 1.24	8.1	5.56 ± 1.86	41.26
steinc9_100	500	1000	50	25	1626	0.07	2.95	4.59	5.38 ± 1.88	8.14	6.61 ± 1.15	40.65
steinc10_10	500	1000	20	25	476	0.22	1.26	27.21	4.12 ± 0.95	9.2	6.58 ± 1.01	42.99
steinc10_100	500	1000	20	25	5187	0.24	1.91	31.47	0.87 ± 1.03	8.89	1.40 ± 1.36	44.01
steinc10_10	500	1000	50	25	210	0.09	3.81	5.24	1.33 ± 2.01	8.62	6.51 ± 2.47	42.63
steinc10_100	500	1000	50	25	2422	0.09	2.06	6.39	0.09 ± 0.29	8.74	0.00 ± 0.00	42.73

Table 6.11: Results for instances SteinC, $h = 25$

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc11_10	500	2500	20	25	27	0.03	0.00	0.42	0.00 ± 0.00	13.61	0.00 ± 0.00	67.83
steinc11_100	500	2500	20	25	274	0.04	0.00	0.4	0.00 ± 0.00	13.61	0.00 ± 0.00	68.58
steinc11_10	500	2500	100	25	27	0.03	0.00	0.4	0.00 ± 0.00	14.13	0.00 ± 0.00	69.3
steinc11_100	500	2500	100	25	274	0.03	0.00	0.41	0.00 ± 0.00	13.6	0.00 ± 0.00	67.42
steinc12_10	500	2500	20	25	59	0.07	0.00	1.58	0.00 ± 0.00	14.01	0.00 ± 0.00	69.39
steinc12_100	500	2500	20	25	604	0.07	0.00	1.64	0.00 ± 0.00	14.23	0.00 ± 0.00	69.29
steinc12_10	500	2500	100	25	59	0.07	0.00	1.61	0.00 ± 0.00	14.12	0.00 ± 0.00	68.74
steinc12_100	500	2500	100	25	604	0.08	0.00	1.64	0.00 ± 0.00	14.09	0.00 ± 0.00	68.47
steinc13_10	500	2500	20	25	439	0.61	0.00	69.58	0.00 ± 0.00	15.77	0.00 ± 0.00	71.85
steinc13_100	500	2500	20	25	4463	0.61	0.00	69.94	0.00 ± 0.00	16.18	0.00 ± 0.00	74.17
steinc13_10	500	2500	100	25	291	0.29	3.78	30.4	1.03 ± 1.17	14.7	2.79 ± 0.65	71.28
steinc13_100	500	2500	100	25	2991	0.27	2.44	31.13	1.34 ± 0.86	15.04	2.99 ± 1.84	71.41
steinc14_10	500	2500	20	25	648	0.83	0.00	122.15	0.00 ± 0.00	15.21	0.00 ± 0.00	69.42
steinc14_100	500	2500	20	25	6566	0.84	0.00	124.31	0.00 ± 0.00	14.61	0.00 ± 0.00	69.5
steinc14_10	500	2500	100	25	369	0.32	1.90	38.38	0.00 ± 0.00	13.94	0.48 ± 0.40	67.56
steinc14_100	500	2500	100	25	3823	0.33	0.60	37.58	0.27 ± 0.57	14	0.86 ± 1.06	67.85
steinc15_10	500	2500	20	25	1248	1.67	0.00	385.97	0.00 ± 0.00	16.25	0.00 ± 0.00	71.83
steinc15_100	500	2500	20	25	12533	1.7	0.00	408.3	0.00 ± 0.00	16.28	0.00 ± 0.00	76.06
steinc15_10	500	2500	100	25	510	0.43	1.96	52.87	0.00 ± 0.00	14.76	0.00 ± 0.00	69.4
steinc15_100	500	2500	100	25	5243	0.41	2.86	56.26	0.00 ± 0.00	14.62	0.15 ± 0.44	71.9
steinc16_10	500	12500	100	25	27	0.64	0.00	5.66	0.00 ± 0.00	252.9	0.00 ± 0.00	1451.4
steinc16_100	500	12500	100	25	274	0.63	0.00	5.71	0.00 ± 0.00	252.62	0.00 ± 0.00	1227.3
steinc16_10	500	12500	200	25	27	0.63	0.00	5.72	0.00 ± 0.00	252.5	0.00 ± 0.00	1222.9
steinc16_100	500	12500	200	25	274	0.63	0.00	5.67	0.00 ± 0.00	252.67	0.00 ± 0.00	1227.3
steinc17_10	500	12500	100	25	59	1.3	0.00	20.74	0.00 ± 0.00	254.07	0.00 ± 0.00	1229.2
steinc17_100	500	12500	100	25	604	1.32	0.00	20.14	0.00 ± 0.00	254.28	0.00 ± 0.00	1609.2
steinc17_10	500	12500	200	25	59	1.31	0.00	20.78	0.00 ± 0.00	253.73	0.00 ± 0.00	1596.2
steinc17_100	500	12500	200	25	604	1.31	0.00	20.12	0.00 ± 0.00	254.37	0.00 ± 0.00	1604.9
steinc18_10	500	12500	100	25	439	10.9	0.00	919.68	0.00 ± 0.00	265.35	0.00 ± 0.00	1628.7
steinc18_100	500	12500	100	25	4463	10.68	0.00	894.37	0.00 ± 0.00	264.64	0.00 ± 0.00	1625.5
steinc18_10	500	12500	200	25	439	10.88	0.00	965.95	0.00 ± 0.00	265.63	0.00 ± 0.00	1627.8
steinc18_100	500	12500	200	25	4463	10.71	0.00	1096.7	0.00 ± 0.00	264.54	0.00 ± 0.00	1632.1
steinc19_10	500	12500	100	25	648	15.42	0.00	1761	0.00 ± 0.00	270	0.00 ± 0.00	1632.8
steinc19_100	500	12500	100	25	6566	15.73	0.00	1905.3	0.00 ± 0.00	269.89	0.00 ± 0.00	1634.7
steinc19_10	500	12500	200	25	648	15.45	0.00	1760.9	0.00 ± 0.00	270.11	0.00 ± 0.00	1633.7
steinc19_100	500	12500	200	25	6566	15.78	0.00	1906.4	0.00 ± 0.00	269.71	0.00 ± 0.00	1633.6
steinc20_10	500	12500	100	25	1248	30.99	0.00	6405.6	0.00 ± 0.00	286.39	0.00 ± 0.00	1660.4
steinc20_100	500	12500	100	25	12533	31.78	0.00	6493.6	0.00 ± 0.00	287.64	0.00 ± 0.00	1664.7
steinc20_10	500	12500	200	25	1248	31.04	0.00	6404.8	0.00 ± 0.00	286.56	0.00 ± 0.00	1668.6
steinc20_100	500	12500	200	25	12533	31.76	0.00	6490.4	0.00 ± 0.00	287.14	0.00 ± 0.00	1659.5

Table 6.12: Results for instances SteinC, $h = 25$ (continued)

Chapter 7

Conclusions

In this thesis we have proposed new models and algorithms for network design problems. Important applications of these problems can be found in electrical engineering, transportation planning, telecommunications and computer science, for example. We have dealt with two general problems: the *Multicommodity Capacitated Fixed-Charge Network Design Problem* (MCFNDP) and the *Steiner Tree Problem with Profits* (STPP). The MCFNDP consists of selecting edges and vertices in a graph in order to allow the flow of some commodities between their designated origins and destinations. A fixed cost is incurred for partially or totally using the capacity of a given edge and an edge-related variable cost is paid for each unity of commodity flow. The STPP is a variant of the classical Steiner tree problem, where besides costs associated with the edges, there are also revenues associated with the vertices of the graph. The goal is to generate a tree that will maximize the collected revenue. Several additional restrictions as a budget limit or hop constraints can be added to the problem. In what follows we summarize our main contributions and we close with some future research directions.

7.1 Summary of contributions

The first part of this thesis was devoted to the MCFNDP. In Chapter 1, we have presented a classification of the most common fixed-charge network design problems. We have also studied the application of Benders decomposition to this family of problems, which are usually modeled using two groups of variables: a set of integer variables represent the opening of the edges, while a set of real variables represent the actual flow of the commodities. This structure seems to be well suited for the application of Benders decomposition,

in which the integer and real-valued variables are isolated and dealt with separately in the master problem and in the subproblem, respectively. The large variety of fixed-charge network design problems that have been successfully solved with Benders decomposition indicates that this approach is well suited for this family of problems and frequently outperforms traditional methods such as branch-and-cut and Lagrangean relaxation. The survey has also shown that many efficient solution methodologies have been proposed for these problems by combining Benders decomposition and other techniques.

In Chapter 2, we have clarified the relationships between three well known classes of inequalities used in solution algorithms for the MCFNDP: Benders, metric and cutset inequalities. We have shown that Benders inequalities associated with extreme rays are always metric, but that it can be interesting to generate other Benders inequalities, and to strengthen them into metric inequalities. This has been demonstrated for the particular case of cutset inequalities, a subclass of Benders inequalities for which we have provided a necessary and sufficient condition for them to be metric inequalities. Computational tests were performed on classical MCFNDP instances. Our results have shown that the strengthening procedure is efficient in reducing convergence times of the Benders algorithm by up to 30%. Moreover, the tests have shown that the procedure can strengthen the traditional cutset inequalities. This is true for sparse graphs, where the tests showed liftings of up to 42%.

In Chapter 3, the basic Benders decomposition framework developed to test the cut strengthening procedure was extended. Several classical extensions have been implemented on top of the basic decomposition scheme. Moreover, we have developed a simple procedure to increase the number of cuts generated while solving the relaxed problem. Although the complete method did not outperform the commercial branch-and-cut of CPLEX for the MCFNDP, our results show that the extensions greatly improve the overall performance of the algorithm. In particular, the generation of extra cuts has proved to be effective in reducing the number of iterations needed to solve the mixed-integer problem. Indeed, the number of iterations in the mixed-integer phase was consistently reduced, and for some instances, less than 10% of the original number of iterations were needed.

The second part of this thesis has dealt with the STPP. In Chapter 4, we have classified the many variants of this problem. We have also presented a summary of the most common preprocessing procedures used to reduce the size of the instances. Since these procedures are almost all developed for a single variant of the STPP, we have analyzed their validity for the other versions of the problem, concluding that they are, in general, not applicable:

out of the eight tests available for the classical STPP problem, only three can be applied to the other variants. In order to better guide our conclusions, we have compared the STPP with the similar and better studied *Travelling Salesman Problem with Profits* (TSPP). This comparison has clearly shown that the research efforts are distributed differently for the two problems. Indeed, the most studied TSPP variation seems to be the orienteering problem, in which the edge costs are dealt as a budgetary constraint. For the STPP, this variant has hardly been mentioned in the literature. This conclusion, added to the fact that budgetary constraints are usually present in real-life network design problems, motivates the development of exact and heuristic algorithms for a variant of the STPP similar to the orienteering TSPP.

In Chapter 5, we have introduced the notion of hop constraints, which impose a limit to the number of edges between a designated root vertex and all other vertices of the solution tree. Such a limit can be useful in many practical situations. For example, in telecommunications problems, hop constraints can be used to increase the reliability of the network or reduce the transmission delays. The combination of hop and budgetary constraints has led to the definition of the *Steiner Tree Problem with Revenues, Budget and Hop Constraints* (STPRBH). We have presented and compared several algorithms for the STPRBH, based on three general mathematical formulations. We prove that the model based on Miller-Tucker-Zemlin (MTZ) constraints is dominated by the model that makes use of position dependent variables. Moreover, we show that branch-and-cut algorithms based on this latter model is very efficient for small values of the hop limit, while for large hops, an algorithm that makes use of Dantzig-Fulkerson-Johnson (DFJ) subtour elimination constraints performs better. Our algorithms were able to solve mid-sized instances of up to 500 vertices and 625 edges. For example, for the same group of instances, the algorithm based on position-dependent constraints was the best in all 54 cases for a hop limit of 3 while, for the same instances and the larger hop limit of 12, the DFJ-based algorithm was the best choice for 34 of the 54 test instances. Our results also showed that heuristic methods are still needed to solve larger instances.

Finally, in Chapter 6, we have developed a greedy algorithm for the STPRBH. This algorithm is capable of obtaining good approximations within very short computational times, even for the larger instances. We have also implemented two improvement algorithms: a *destroy-and-repair* method that basically works by modifying the original edge costs and rerunning the greedy algorithm, and a tabu search that uses three neighborhoods and classical tabu mechanisms to escape from local optima. The tabu search proved to be

very efficient, obtaining mean deviations of less than 0.3% for the mid-sized instances (compared to 2.50% and 1.13% for the greedy and destroy-and-repair algorithm, respectively) and improving the solutions obtained by the greedy algorithm for the larger instances (for which the optimal solutions were not known) by almost 2% on average, and by more than 20% in some cases.

7.2 Future research directions

The MCFNDP and the STPP are graph optimization problems that have received considerable attention in the field of combinatorial optimization. In this thesis, we have dealt with both of these problems from different perspectives. Our research suggests some new research avenues.

The first part of this thesis was devoted to the study of the MCFNDP. In Chapter 2, we have focused on three classical groups of valid inequalities for this problem, and we have established relationships between them. These have enabled the development of a simple procedure to strengthen the widely used Benders and cutset inequalities. The use of the proposed procedure in other contexts is certainly an interesting topic for future research. In Chapter 3, we have proposed a Benders decomposition algorithm for the MCFNDP. Despite all the extensions proposed, the results were outperformed by the commercial branch-and-cut solver of CPLEX. Future research could focus on the generation of more efficient cuts. Indeed, the proposed extra-cuts generation procedures proved to be quite effective and a future research project should investigate the application of such procedures to similar problems.

The second part of this thesis focused on the STPP, in particular, on the versions with budget and hop constraints. Chapter 5 was dedicated to the study of mathematical models and exact algorithms for this class of problems. Due to the importance of hop constraints, future research could try to extend our methodology to other network design problems in which the objective or constraints are different. Moreover, a theoretical comparison between the model based on position-dependent variables and the model based on DFJ constraints is certainly of interest. Finally, Chapter 6 was devoted to three heuristic procedures for the STPRBH. The interest of this chapter lies mostly in the difference of the approaches. Indeed, the greedy and the destroy-and-repair algorithms are very simple and easily implementable and obtain good approximations. The tabu search heuristic is more sophisticated but also more flexible, and provides better results than the previous methods.

As for Chapter 5, future work could address extensions of the developed methodologies to other versions of the Steiner tree problem with hop constraints.

Bibliography

Altiparmak, F., Dengiz, B. and Smith, A. E. (2003). Optimal design of reliable computer networks: A comparison of metaheuristics, *Journal of Heuristics* **9**: 471–487.

Armacost, A. P., Barnhart, C. and Ware, K. A. (2002). Composite variable formulations for express shipment service network design, *Transportation Science* **36**: 1–20.

Arya, S. and Ramesh, H. (1998). A 2.5 factor approximation algorithm for the k-MST problem, *Information Processing Letters* **65**: 117–118.

Atamtürk, A. (2002a). On capacitated network design cut-set polyhedra, *Mathematical Programming* **92**: 425–437.

Atamtürk, A. (2002b). On splittable and unsplittable capacitated network design arc-set polyhedra, *Mathematical Programming* **92**: 315–333.

Awerbuch, B., Azar, Y., Blum, A. and Vempala, S. (1998). New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen, *SIAM Journal on Computing* **28**: 254–262.

Balakrishnan, A. and Altinkemer, K. (1992). Using a hop-constrained model to generate alternative communication network design, *ORSA Journal on Computing* **4**: 192–205.

Balakrishnan, A., Magnanti, T. L. and Mirchandani, P. (1994). Modeling and heuristic worst-case performance analysis of the two-level network design problem, *Management Science* **40**: 846–867.

Balakrishnan, A., Magnanti, T. L. and Mirchandani, P. (1997). Network design, in M. Dell'Amico and F. Maffioli and S. Martello (eds), *Annotated Bibliographies in Combinatorial Optimization*, Wiley, New York : 311–334.

Balakrishnan, A. and Patel, N. (1987). Problem reduction methods and a tree generation algorithm for the Steiner network problem, *Networks* **17**: 65–85.

Barahona, F. (1996). Network design using cut inequalities, *SIAM Journal of Optimization* **6**: 823–837.

Beasley, J. E. (1984). An algorithm for the Steiner problem in graphs, *Networks* **14**: 147–159.

Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail, *Journal of the Operational Research Society* **41**: 1069–1072.

Benchakroun, A., Ferland, J. A. and Cléroux, R. (1992). Distribution system planning through a generalized Benders decomposition approach, *European Journal of Operational Research* **62**: 149–162.

Benders, J. F. (1962). Partitioning procedures for solving mixed variables programming problems, *Numerische Mathematik* **4**: 238–252.

Bertsekas, D. P. and Gallager, R. G. (1987). *Data Networks*, Prentice-Hall, Englewood Cliffs.

Bienstock, D., Goemans, M. X., Simchi-Levi, D. and Williamson, D. (1993). A note on the prize collecting traveling salesman problem, *Mathematical Programming* **59**: 413–420.

Binato, S., Pereira, M. V. F. and Granville, S. (2001). A new Benders decomposition approach to solve power transmission network design problems, *IEEE Transactions on Power Systems* **16**: 235–240.

Boyce, D. E., Farhi, A. and Weischedel, R. (1973). Optimal network problem: A branch-and-bound algorithm, *Environment and Planning* **5**: 519–533.

Cai, X., McKinney, D. C., Lasdon, L. S. and Watkins, Jr., D. W. (2001). Solving large nonconvex water resources management models using generalized Benders decomposition, *Operations Research* **49**: 235–245.

Canuto, S. A., Resende, M. G. C. and Ribeiro, C. C. (2001). Local search with perturbations for the prize-collecting Steiner tree problem in graphs, *Networks* **38**: 50–58.

Carneiro, M. S., França, P. and Silveira, P. D. (1996). Long-range planning of power distribution systems: Secondary networks, *Computers & Electrical Engineering* **22**: 179–191.

Cavelluci, C. and Lyra Filho, C. (1997). Minimization of energy losses in electric power distribution systems by intelligent search strategies, *International Transactions in Operational Research* **4**: 23–33.

Chao, I-M., Golden, B. L. and Wasil, E. A. (1996). A fast and effective heuristic for the orienteering problem, *European Journal of Operational Research* **88**: 475–489.

Chawla, S., Kitchin, D., Rajan, U., Ravi, R. and Sinha, A. (2003). Profit maximizing mechanisms for the extended multicasting game, Proceedings of the 4th ACM Conference on Eletronic Commerce, San Diego : 190–191.

Chopra, S. and Rao, M. R. (1994a). The Steiner tree problem I: Formulations, compositions and extension of facets, *Mathematical Programming* **64**: 209–229.

Chopra, S. and Rao, M. R. (1994b). The Steiner tree problem II: Properties and classes of facets, *Mathematical Programming* **64**: 231–246.

Chouman, M., Crainic, T. G. and Gendron, B. (2003). A cutting-plane algorithm based on cutset inequalities for multicommodity capacitated fixed charge network design, *Publication CRT-316,* Centre de recherche sur les transports, Université de Montréal.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem, *Technical Report 388,* Graduate School of Industrial Administration, Carnegie Mellon University.

Cole, R., Hariharan, R., Lewenstein, M. and Porat, E. (2001). A faster implementation of the Goemans-Williamson clustering algorithm, SODA'01: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, Washington : 17–25.

Cordeau, J.-F., Pasin, F. and Solomon, M. M. (2006). An integrated model for logistics network design, *Annals of Operations Research* (forthcoming).

Cordeau, J.-F., Soumis, F. and Desrosiers, J. (2000). A Benders decomposition approach for the locomotive and car assignment problem, *Transportation Science* **34**: 133–149.

Cordeau, J.-F., Soumis, F. and Desrosiers, J. (2001). Simultaneous assignment of locomotives and cars to passenger trains, *Operations Research* **49**: 531–548.

Courant, R. and Robbins, H. (1941). *What is mathematics?*, Oxford University Press, New York.

Crainic, T. G., Frangioni, A. and Gendron, B. (2001). Bundle-based relaxation methods for multicommodity capacitated fixed charge network design problems, *Discrete Applied Mathematics* **112**: 73–99.

Crainic, T. G., Gendreau, M. and Farvolden, J. M. (2000). A simplex-based tabu search method for capacitated network design, *INFORMS Journal on Computing* **12**: 223–236.

Crainic, T. G., Gendron, B. and Hernu, G. (2004). A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity fixed-charge network design, *Journal of Heuristics* **10**: 525–545.

Cunha, A. S., Lucena, A., Maculan, N. and Resende, M. G. C. (2003). A relax and cut algorithm for the prize collecting Steiner problem in graphs, Proceedings of Mathematical Programming in Rio, Búzios, Brazil : 72–78.

Dahl, G., Gouveia, L. and Requejo, C. (2006). On formulations and methods for the hop-constrained minimum spanning tree problem, in P. Pardalos and M. G. C. Resende (eds), *Handbooks of Telecommunications*, Springer, New York : 493–515.

Dantzig, G. B., Fulkerson, D. R. and Johnson, S. M. (1954). Solution of a large-scale traveling-salesman problem, *Operations Research* **2**: 393–410.

Dell'Amico, M., Maffioli, F. and Sciomachen, A. (1998). A Lagrangian heuristic for the prize-collecting travelling salesman problem, *Annals of Operations Research* **81**: 289–305.

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

Duin, C. W. and Volgenant, A. (1987). Some generalizations of the Steiner problem in graphs, *Networks* **17**: 353–364.

Duin, C. W. and Volgenant, A. (1989). Reduction tests for the Steiner problem in graphs, *Networks* **19**: 549–567.

Engevall, S., Göthe-Lundgren, M. and Värbrand, P. (1998). A stronger lower bound for the node weighted Steiner tree problem, *Networks* **31**: 11–17.

Feigenbaum, J., Papadimitriou, C. H. and Shenker, S. (2001). Sharing the cost of multicast transmissions, *Journal of Computer and System Sciences* **63**: 21–41.

Feillet, D., Dejax, P. and Gendreau, M. (2006). Traveling salesman problems with profits, *Transportation Science* (forthcoming).

Feremans, C., Labbé, M. and Laporte, G. (2002). A comparative analysis of several formulations for the generalized minimum spanning tree problem, *Networks* **39**: 29–34.

Fischetti, M. (1991). Facets of two Steiner arborescence polyhedra, *Mathematical Programming* **51**: 401–419.

Fischetti, M., Salazar-González, J. J. and Toth, P. (1998). Solving the orienteering problem through branch-and-cut, *INFORMS Journal on Computing* **10**: 133–148.

Fischetti, M. and Toth, P. (1988). An additive approach for the optimal solution of the prize-collecting travelling salesman problem, in B. L. Golden and A. A. Assad (eds), *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam : 319–343.

Florian, M., Guérin, G. and Bushel, G. (1976). The engine scheduling problem on a railway network, *INFOR* **14**: 121–128.

Friedman, E. J. and Parkes, D. C. (2003). Pricing WiFi at Starbucks: Issues in online mechanism design, Proceedings of the 4th ACM Conference on Electronic Commerce : 240–241.

Gabrel, V., Knippel, A. and Minoux, M. (1999). Exact solution of multicommodity network optimization problems with general step cost functions, *Operations Research Letters* **25**: 15–23.

Gallo, G. (1983). Lower planes for the network design problem, *Networks* **13**: 411–426.

Garcia, M. (1994). *Árvores com restrições de diâmetro*, Master thesis, University of Lisbon (in Portuguese).

Garey, M. R., Graham, R. L. and Johnson, D. S. (1977). The complexity of computing Steiner minimal trees, *SIAM Journal on Applied Mathematics* **32**: 835–859.

Gascon, V., Benchakroun, A. and Ferland, J. A. (1993). Electricity distribution planning model: A network design approach for solving the master problem of the Benders decomposition method, *INFOR* **31**: 205–220.

Gavish, B. (1982). Topological design of centralized computer networks – formulations and algorithms, *Networks* **12**: 355–377.

Gavish, B. (1983). Formulations and algorithms for the capacitated minimal directed tree problem, *Journal of the Association for Computing Machinery* **30**: 118–132.

Gendreau, M., Hertz, A. and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem, *Management Science* **40**: 1276–1290.

Gendreau, M., Laporte, G. and Semet, F. (1998a). A branch-and-cut algorithm for the undirected selective travelling salesman problem, *Networks* **32**: 263–273.

Gendreau, M., Laporte, G. and Semet, F. (1998b). A tabu search heuristic for the undirected selective travelling salesman problem, *European Journal of Operational Research* **106**: 539–545.

Gendron, B. and Crainic, T. G. (1994). Relaxations for multicommodity capacitated network design problems, *Publication CRT-965,* Centre de recherche sur les transports, Université de Montréal.

Gendron, B. and Crainic, T. G. (1996). Bounding procedures for multicommodity capacitated fixed charge network design problems, *Publication CRT-96-06,* Centre de recherche sur les transports, Université de Montréal.

Gendron, B., Crainic, T. G. and Frangioni, A. (1999). Multicommodity capacitated network design, in B. Sanso and P. Soriano (eds), *Telecommunications Network Planning*, Kluwer, Boston : 1–19.

Geoffrion, A. M. (1972). Generalized Benders decomposition, *Journal of Optimization Theory and Applications* **10**: 237–260.

Geoffrion, A. M. and Graves, G. W. (1974). Multicommodity distribution system design by Benders decomposition, *Management Science* **20**: 822–844.

Ghamlouche, I., Crainic, T. G. and Gendreau, M. (2003). Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design, *Operations Research* **51**: 655–667.

Ghamlouche, I., Crainic, T. G. and Gendreau, M. (2004). Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design, *Annals of Operations Research* **131**: 109–133.

Gilbert, E. N. and Pollak, H. O. (1968). Steiner minimal trees, *SIAM Journal on Applied Mathematics* **16**: 1–29.

Günlük, O. (1999). A branch-and-cut algorithm for capacitated network design problems, *Mathematical Programming* **86**: 17–39.

Goemans, M. X. (1994). The Steiner tree polytope and related polyhedra, *Mathematical Programming* **63**: 157–182.

Goemans, M. X. and Williamson, D. P. (1995). A general approximation technique for constrained forest problems, *SIAM Journal on Computing* **24**: 296–317.

Golden, B. L., Levy, L. and Vohra, R. (1987). The orienteering problem, *Naval Research Logistics* **34**: 307–318.

Golden, B. L., Wang, Q. and Liu, L. (1988). A multifaceted heuristic for the orienteering problem, *Naval Research Logistics* **35**: 359–366.

Gouveia, L. (1995a). A $2n$ constraint formulation for the capacitated minimal spanning tree problem, *Operations Research* **43**: 130–141.

Gouveia, L. (1995b). Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints, *Computers & Operations Research* **22**: 959–970.

Gouveia, L. (1996). Multicommodity flow models for spanning trees with hop constraints, *European Journal of Operational Research* **95**: 178–190.

Gouveia, L. (1997). Using generalized capacitated trees for designing the topology of local access networks, *Telecommunications Systems* **7**: 315–337.

Gouveia, L. (1998). Using variable redefinition for computing lower bounds for minimum spanning and Steiner trees with hop constraints, *INFORMS Journal on Computing* **10**: 180–188.

Gouveia, L. (1999). Using hop-indexed models for constrained spanning and Steiner tree models, in B. Sansò and P. Soriano (eds), *Telecommunications Network Planning*, Kluwer, Boston : 21–32.

Gouveia, L. and Magnanti, T. L. (2003). Network flow models for designing diameter-constrained minimum-spanning and Steiner trees, *Networks* **41**: 159–173.

Gouveia, L. and Requejo, C. (2001). A new Lagrangean relaxation approach for the hop-constrained minimum spanning tree problem, *European Journal of Operational Research* **132**: 539–552.

Gouveia, L. and Telhada, J. (2001). An augmented arborescence formulation for the two-level network design problem, *Annals of Operations Research* **106**: 47–61.

Granville, S. and Pereira, M. V. F. (1985). Analysis of the linearized power flow model in Benders decomposition, *Technical Report SOL 85-04,* Stanford University.

Gutierrez, G. J., Kouvelis, P. and Kurawarwala, A. A. (1996). A robustness approach to uncapacitated network design problems, *European Journal of Operational Research* **94**: 362–376.

Hoang, H. H. (1973). A computational approach to the selection of an optimal network, *Management Science* **19**: 488–498.

Hoang, H. H. (1982). Topological optimization of networks: A nonlinear mixed integer model employing generalized Benders decomposition, *IEEE Transactions on Automatic Control* **27**: 164–169.

Holmberg, K. and Yuan, D. (2000). A lagrangean heuristic based branch-and-bound approach for the capacitated network design problem, *Operations Research* **48**: 461–481.

Hwang, F. K., Richards, D. S. and Winter, P. (1992). *The Steiner Tree Problem*, North-Holland, Amsterdan.

Johnson, D. S., Minkoff, M. and Phillips, S. (2000). The prize collecting Steiner tree problem: Theory and practice, Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms, San Francisco : 760–769.

Kataoka, S. and Morito, S. (1988). An algorithm for the single constraint maximum collection problem, *Journal of the Operations Research Society of Japan* **31**: 515–530.

Keller, C. P. (1989). Algorithms to solve the orienteering problem: A comparison, *European Journal of Operational Research* **41**: 224–231.

Khoury, B. N., Pardalos, P. M. and Du, D.-Z. (1993). A test problem generator for the Steiner tree problem in graphs, *ACM Transactions on Mathematical Software* **19**: 509–522.

Klau, G. W., Ljubić, I., Moser, A., Mutzel, P., Neuner, P., Pferschy, U., Raidl, G. and Weiskircher, R. (2004). Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem, *Springer Lecture Notes in Computer Science* **3102**: 1304–1315.

Klau, G. W., Ljubić, I., Mutzel, P., Pferschy, U. and Weiskircher, R. (2003). The fractional prize-collecting Steiner tree problem on trees, *Technical Report TR-186-1-03-01,* Institute of Computer Graphics and Algorithms, Vienna University of Technology.

Koch, T. and Martin, A. (1998). Solving Steiner tree problems in graphs to optimality, *Networks* **32**: 207–232.

Laporte, G. and Martello, S. (1990). The selective travelling salesman problem, *Discrete Applied Mathematics* **26**: 193–207.

Laporte, G. and Rodríguez-Martín, I. (2006). Locating cycles in transportation and telecommunications networks, *Networks* (forthcoming).

Lawler, E. (1976). *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.

LeBlanc, L., Chifflet, J. and Mahey, P. (1999). Packet routing in telecommunication networks with path and flow restrictions., *INFORMS Journal on Computing* **11**: 188–197.

Lederer, P. J. and Nambimadom, R. S. (1998). Airline network design, *Operations Research* **46**: 785–804.

Leifer, A. C. and Rosenwein, M. (1994). Strong linear programming relaxations for the orienteering problem, *European Journal of Operational Research* **73**: 517–523.

Ljubić, I., Weiskircher, R., Pferschy, U., Klau, G., Mutzel, P. and Fischetti, M. (2005). Solving the prize-collecting Steiner tree problem to optimality, Proceedings of ALENEX, Seventh Workshop on Algorithm Engineering and Experiments, Vancouver.

Lucena, A. (1991). Tight bounds for the Steiner problem in graphs, Presentation made at the TIMS XXX - SOBRAPO XXIII Joint International Meeting, Rio de Janeiro.

Lucena, A. and Resende, M. G. C. (2004). Strong lower bounds for the prize collecting Steiner problem in graphs, *Discrete Applied Mathematics* **141**: 277–294.

Magnanti, T. L. and Mirchandani, P. (1993). Shortest-paths, single origin-destination network design and associated polyhedra, *Networks* **23**: 103–121.

Magnanti, T. L., Mirchandani, P. and Vachani, R. (1995). Modeling and solving the two-facility capacitated network loading problem, *Operations Research* **43**: 142–157.

Magnanti, T. L., Mireault, P. and Wong, R. T. (1986). Tailoring Benders decomposition for uncapacitated network design, *Mathematical Programming Study* **26**: 112–154.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders decomposition: Algorithmic enhacement and model selection criteria, *Operations Research* **23**: 464–484.

Magnanti, T. L. and Wong, R. T. (1984). Network design and transportation planning: Models and algorithms, *Transportation Science* **18**: 1–56.

Mahey, P., Benchakroun, A. and Boyer, F. (2001). Capacity and flow assignment of data networks by generalized Benders decomposition, *Journal of Global Optimization* **20**: 173–193.

Mahey, P., Ourou, A., LeBlanc, L. and Chifflet, J. (1998). A new proximal decomposition algorithm for routing in telecommunications networks, *Networks* **31**: 227–238.

Margot, F., Prodon, A. and Liebling, T. M. (1994). Tree polyhedron on 2-tree, *Mathematical Programming* **63**: 183–192.

McDaniel, D. and Devine, M. (1977). A modified Benders' partitioning algorithm for mixed integer programming, *Management Science* **24**: 312–319.

Megiddo, N. (1979). Combinatorial optimization with rational objective functions, *Mathematics of Operations Research* **4**: 414–424.

Miller, C. E., Tucker, A. W. and Zemlin, R. A. (1960). Integer programming formulations and travelling salesman problems, *Journal of the Association of Computing Machinery* **7**: 326–329.

Minkowski, H. (1896). *Geometrie der Zahlen*, Teubner, Leipzig [reprinted: Chelsea, New York, 1953].

Minoux, M. (1989). Network synthesis and optimum network design problems: Models, solution methods and applications, *Networks* **19**: 313–360.

Mittenthal, J. and Noon, C. E. (1992). Insert/delete heuristic for the travelling salesman subsettour problem with one additional constraint, *Journal of the Operational Research Society* **43**: 277–283.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search, *Computers & Operations Research* **24**: 1097–1100.

Myung, Y., Lee, C. and Tcha, D. (1995). On the generalized minimum spanning tree problem, *Networks* **26**: 231–241.

Nguyen, S. (1974). An algorithm for the traffic assignement problem, *Transportation Science* **8**: 203–216.

Oliveira, G. C., Costa, A. P. C. and Binato, S. (1995). Large scale transmission network planning using optimization and heuristics techniques, *IEEE Transactions on Power Systems* **10**: 1828–1834.

Onaga, K. and Kakusho, O. (1971). On feasibility conditions of multicommodity flows in networks, *IEEE Transactions on Circuit Theory* **18**: 425–429.

Ouorou, A., Luna, H. P. L. and Mahey, P. (2001). Multicommodity network expansion under elastic demands, *Optimization and Engineering* **2**: 277–292.

Pereira, M. V. F., Pinto, L. M. V. G., Cunha, S. H. F. and Oliveira, G. C. (1985). A decomposition approach to automated generation-transmission expansion planning, *IEEE Transactions on Power Apparatus and Systems* **104**: 3074–3081.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems, *Computers & Operations Research* (forthcoming).

Prim, R. (1957). Shortest connection networks and some generalizations, *Bell System Technology Journal* **36**: 1389–1401.

Ramesh, R. and Brown, K. M. (1991). An efficient four-phase heuristic for the generalized orienteering problem, *Computers & Operations Research* **18**: 151–165.

Ramesh, R., Yoon, Y.-S. and Karwan, M. H. (1992). An optimal algorithm for the orienteering tour problem, *ORSA Journal on Computing* **4**: 155–165.

Ramirez-Rosado, I. J. and Dominguez-Navarro, J. A. (2006). New multiobjective tabu search algorithm for fuzzy optimal planning of power distribution systems, *IEEE Transactions on Power Systems* **21**: 224–233.

Randazzo, C. D. and Luna, H. P. L. (2001). A comparison of optimal methods for local access uncapacitated network design, *Annals of Operations Research* **106**: 263–286.

Randazzo, C. D., Luna, H. P. L. and Mahey, P. (2001). Benders decomposition for local access network design with two technologies, *Discrete Mathematics and Theoretical Computer Science* **4**: 235–246.

Riis, M. and Andersen, K. A. (2002). Capacitated network design with uncertain demand, *INFORMS Journal on Computing* **14**: 247–260.

Romero, R. and Monticelli, A. (1994a). A hierarchical decomposition approach for transmission network expansion planning, *IEEE Transactions on Power Systems* **9**: 373–380.

Romero, R. and Monticelli, A. (1994b). A zero-one implicit enumeration method for optimizing investments in transmissions expansion plannings, *IEEE Transactions on Power Systems* **9**: 1385–1391.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, *Transportation Science* (forthcoming).

Segev, A. (1987). The node-weighted Steiner tree problem, *Networks* **17**: 1–17.

Sellman, M., Kliewer, G. and Koberstein, A. (2002). Capacitated network design, cardinatility cuts and coupled variable fixing algorithms based on Lagrangian relaxations, *Technical Report tr-ri-02-234,* Department of Computer Science, University of Paderborn.

Shogan, A. W. (1983). Constructing a minimal cost spanning tree subject to resource constraints and flow requirements, *Networks* **13**: 169–190.

Sridhar, V. and Park, J. S. (2000). Benders-and-cut algorithm for fixed-charge capacitated network design problem, *European Journal of Operational Research* **125**: 622–632.

Sridhar, V., Park, J. S. and Gavish, B. (2000). LP-based heuristic algorithms for interconnecting token rings via source routing bridges, *Journal of Heuristics* **6**: 149–166.

Standard and Poor's (2000). Commercial transportation industry survey.

Stoer, M. and Dahl, G. (1994). A polyhedral approach to multicommodity survivable network design, *Numerische Mathematik* **68**: 149–167.

Tasgetiren, F. M. and Smith, A. E. (2000). A genetic algorithm for the orienteering problem, *In Proceedings of the 2000 Congress on Evolutionary Computation,* San Diego : 1190–1195.

Tsamasphyrou, P., Renaud, A. and Carpentier, P. (2000). Transmission network planning under uncertainty with Benders decomposition, *Lecture Notes in Economics and Mathematical Systems* **481**: 457–472.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering, *Journal of the Operational Research Society* **35**: 797–809.

Uchoa, E. (2006). Reduction tests for the prize-collecting Steiner problem, *Operations Research Letters*. (forthcoming).

Voß, S. (1999). The Steiner tree problem with hop constraints, *Annals of Operations Research* **86**: 321–345.

Wang, Q., Sun, X., Golden, B. L. and Jia, J. (1995). Using artificial neural networks to solve the orienteering problem, *Annals of Operations Research* **61**: 111–120.

Zhao, G. (2001). A log-barrier method with Benders decomposition for solving two-stage stochastic linear programs, *Mathematical Programming* **90**: 507–536.