Survavibility in Multilayer Networks : models and Polyhedra

Raouia Taktak

N° attribué par la bibliothèque

Université Paris-Dauphine

ÉCOLE DOCTORALE DE DAUPHINE

THÈSE

présentée par

Raouia TAKTAK

pour obtenir le grade de

Docteur de l'université Paris-Dauphine
Spécialité : Informatique

Survivability in Multilayer Networks: Models and Polyhedra.

Soutenue publiquement le 04 juillet 2013 devant le jury :

A.R.	Mahjoub	Directeur de thèse	Université Paris-Dauphine, France
V.	Gabrel	Co-encadrant	Université Paris-Dauphine, France
L.	Gouveia	Rapporteur	Universidade de Lisboa, Portugal
A.	Quilliot	Rapporteur	Université Blaise Pascale, Clermont II, France
S.	Borne	Examinateur	Université Paris 13, France
E.	Gourdin	Examinateur	Orange Labs, Issy-les-moulineaux, France
S.Th.	McCormick	Examinateur	University of British Columbia, Canada
E.	Uchoa	Examinateur	Universidade Federal Fluminense, Brasil

Abstract

With the explosive growth of data traffic, telecommunication networks have evolved toward a model of high-speed IP routers interconnected by intelligent optical core networks. Moreover, as telecommunication networks play a crucial role for the transfer of important data, they must present a minimum degree of survivability against accidental breakdowns. One of the most useful strategies to ensure a network's survivability, is to satisfy some connectivity requirements, that is to guarantee a certain number of disjoint paths between some pairs of the nodes in the network.

This thesis deals with a problem related to survivability issues in multilayer IP-over-WDM networks. Given a set of traffic demands for which we know a survivable logical routing in the IP layer, our purpose is to determine the corresponding survivable topology in the WDM layer. More precisely, suppose we are given an IP-over-WDM network with weights on the WDM layer, a set of demands and two-disjoint paths for each demand. The problem is to determine in the WDM layer a minimum weight subgraph containing two routing paths for each demand. These paths must be node-disjoint and go in the same order through the optical switches corresponding to the routers visited in the paths of the IP layer.

We show that in addition to its importance in the telecommunication context, the problem is very interesting from a theoretical point of view. We then prove that, under some conditions, the problem is NP-hard even for a single demand. Moreover, we propose four integer linear programming formulations for the problem. The first one, called cut formulation, is induced by the so-called cut inequalities which are in an exponential number. We carry out a polyhedral investigation of the convex hull of its feasible solutions. We identify several families of valid inequalities and discuss their facial aspect. We then devise separation routines for these inequalities and develop a Branch-and-Cut algorithm. The second formulation is a path formulation which uses the paths between the origin-destination of the demands' sections. Based on this formulation, we devise a Branch-and-Price algorithm for the problem. In addition, we present a third formulation which only uses the design variables. This formulation is given in the case when we only have 3 terminals. For 4 terminals and more, such a formulation seems to be a hard problem. Finally, the last formulation is a compact formulation which, in addition to the design variables, uses a further family of variables. We show that this formulation performs well and gives a tighter bound for the problem.

Key words : IP-over-WDM networks, survivability, complexity, polytope, facet, Branch-and-Cut algorithm, Branch-and-Price algorithm, extended formulation.

Résumé

Paralèllement à l'augmentation significative du volume des informations échangées, à la multiplication des services et à la diversification des données, les réseaux de télécommunications évoluent vers une structure multicouche qui s'avère la plus adaptée à tous ces changements. L'architecture IP-sur-WDM, exploitant la technologie de la fibre optique, présente en particulier une infrastructure prometteuse pour les réseaux futurs. De plus, étant impliqué dans le transport de données importantes, ces réseaux doivent être dotés d'un niveau de fiabilité suffisant, leur permettant de rétablir les connexions en cas de panne de l'un des équipements dans le réseau.

Dans cette thèse, nous nous intéressons à un problème de fiabilité dans les réseaux multicouches IP-sur-WDM. Etant donné un ensemble de demandes pour lesquelles on connaît une topologie fiable dans la couche IP, le problème consiste à sécuriser la couche optique WDM en y cherchant une topologie fiable. Plus précisément, on suppose données un réseau IP-sur-WDM avec des poids sur la couche WDM, un ensemble de demandes et deux chemins sommet-disjoint routant chaque demande dans la couche IP. Le problème est de déterminer dans la couche WDM un sous-graphe de poids minimum contenant pour chaque demande deux chemins de routage. Ces chemins doivent être sommet-disjoints et doivent visiter les brasseurs de la couche WDM correspondant aux routeurs visités dans les chemins de la couche IP, tout en gardant le même ordre de passage.

Nous montrons que le problème est d'une importance cruciale non seulement d'un point de vue pratique mais aussi sur le plan théorique. Nous montrons que le problème est NP-complet même dans le cas d'une seule demande. Ensuite, nous proposons quatre formulations en termes de programmes linéaires en nombres entiers pour le problème. La première, dite formulation coupes, est induite par des inégalités dites de coupes et contient un nombre exponentiel de contraintes. Nous menons une investigation approfondie du polyèdre associé à cette formulation. Nous identifions de nouvelles familles de contraintes valides et étudions leur aspect facial. Nous décrivons également des procédures de séparation pour ces inégalités et développons un algorithme de coupes et branchements pour le problème. Une deuxième formulation, basée sur les chemins et utilisant plutôt un nombre exponentiel de variables, est considérée dans un second temps. Nous proposons un algorithme de branchements et génération de colonnes pour cette formulation. Par la suite, nous discutons d'une troisième formulation qui utilise uniquement les variables naturelles du problème. Nous montrons que cette formulation est valide dans le cas de 3 terminaux par demande. Nous discutons aussi du cas de 4 terminaux et plus. Enfin, nous présentons une formulation étendue compacte qui, en plus des variables naturelles, utilise une autre famille de variables. Nous montrons que cette formulation fournit une bonne borne inférieure et permet de résoudre efficacement le problème.

Mots clés : Réseaux de télécommunications IP-sur-WDM, sécurisation, complexité, polytope, facette, algorithme de coupes et branchements, algorithme de branchements et génération de colonnes, formulation étendue.

Contents

Introduction

In the few past years, telecommunication networks have witnessed an explosive growth of data traffic and a multiplication of various applications and services. This rapid evolution has implied a need for a new promising architecture that enables an efficient management of huge amount of diverse data. Telecommunication networks have hence evolved toward a multilayer architecture consisting of a stack of subnetworks, called layers, each characterized with appropriate protocols and technologies. In fact, thanks to the recent technologies, and mainly the advances in the optical systems, the networks' structure have steadily progressed to an IP[1]-over-WDM[2] model. This two-layer model is particularly considered as an important opportunity for telecommunication carriers who want to vary services and add more multimedia applications. Moreover, recent IP-over-WDM networks are managed by protocols like GMPLS[3] which play a crucial role to ensure the interoperability between the IP and WDM levels.

The migration to a multilayer architecture has engendered new challenges for telecommunication operators since network design has to fit to the best to the multilevel model. Moreover, as telecommunication networks are involved in several domains and regularly conducting huge amounts of important information, they must prove a minimum degree of robustness and survivability. In reality, the performance of a network does not only depend on its capability to transfer data, but also on its ability to maintain services in the event of accidental failures. In order to ensure a continuous routing of information even in the case of a breakdown of some components of the network, a possible strategy consists in providing protection paths. More formally, this can be translated by some connectivity requirements in the network, that is to ensure a minimum number of disjoint paths between all or some pairs of nodes of the network. Survivability issues become more and more important in the context of IP-over-WDM networks because of the strong interaction between the different layers. In this sense,

[1]Internet Protocol
[2]Wavelength Division Multiplexing
[3]Generalized MultiProtocol Label Switching

it is necessary to establish recovery strategies at the two layers so as to define the protection responsibilities of each one and coordinate them to avoid confused actions against the same failure.

The multilayer survivable network design problem has been widely studied in the literature. Different variants of the problem have been considered and solved using several approaches of resolution such as the polyhedral methods. These methods have proved to be a powerful tool to efficiently tackle hard combinatorial problems. Initiated by Edmonds in the context of the matching problem [45], this technique consists in reducing the resolution of a combinatorial problem to that of one or more linear programs. This is in particular based on giving a complete (or a partial) description of the polytope of solutions with a system of linear inequalities. The polyhedral approach has been proved to be very efficient when applied to many combinatorial optimization problems such as the Travelling Salesman Problem, the Network Design Problem and the Max-Cut Problem.

In this thesis, we use the above-mentioned technique, as well as other exact methods, to study a survivability problem in the multilayer IP-over-WDM networks, called the *Multilayer Survivable Optical Network Design* problem (MSOND problem). Consider an IP-over-WDM network consisting of a logical IP layer over an optical WDM layer. The IP layer is composed of IP routers interconnected by logical links and the WDM layer consists of optical switches interconnected by optical fibres. We suppose given a set of demands between the IP routers such that for each demand we know two node-disjoint paths routing it in the logical IP layer. The MSOND problem consist in finding, for each demand, two optical paths routing it in the WDM layer. These paths must be node-disjoint and go in the same order through the optical switches corresponding to the routers visited in the logical paths of the IP layer.

We propose several integer linear programming formulations for the problem and devise efficient exact cutting planes based algorithms.

This thesis is organized as follows.

In Chapter 1, we present basic notions of combinatorial optimization and the notation that will be used throughout the manuscript.

Chapter 2 introduces the practical context of the problem. Different notions related to the multilayer telecommunication networks are presented. In particular, we focus on the multilayer IP-over-WDM networks and present possible strategies to ensure their survivability. A state-of-the-art on the survivable network design problem as well as its application in the multilayer context is also given.

In Chapter 3, we introduce the Multilayer Survivable Optical Network Design problem (MSOND problem). We show that this problem is closely related to the Steiner Travelling Salesman problem. Moreover, we prove that it is an NP-hard problem even for the simple cases. Chapters 4 to 7 present various integer linear programming formulations as well as efficient exact algorithms used to model and solve the MSOND problem, respectively.

In Chapter 4, we propose a formulation based on the so-called cut inequalities. We discuss the associated polytope and present several classes of valid inequalities. We also investigate the facial structure of these inequalities. Based on these results, we devise, in Chapter 5, a Branch-and-Cut algorithm. We describe the separation routines and present substantial computational results.

In Chapter 6, we propose a path formulation for the MSOND problem, using a polynomial number of inequalities, yet an exponential number of variables. We propose a Branch-and-Price algorithm for this formulation. We discuss the associated pricing problem and prove that it reduces to a classical Shortest Path Problem. We also describe an appropriate branching rule and propose a primal heuristic.

Finally, in Chapter 7, we discuss two further formulations. The first one, called natural formulation, is given in terms of the design variables. We show that the formulation is valid in the case when there are only 3 terminals. The second formulation, called extended formulation, is given using an extra family of variables in addition to the design variables. We present experimental results which show that this formulation provide a tighter bound and performs very well for the resolution of the MSOND problem.

Chapter 1

Preliminary notions

Contents

This chapter is devoted to give some preliminary notions about combinatorial optimization, complexity theory, polyhedral approaches and exact methods of resolution. In particular, we explain the principles of cutting planes and branch-and-cut method, as well as column generation and branch-and-price method. We then present extended formulations and briefly discuss their importance in combinatorial optimization. Finally, we give some basic definitions and notations related to graph theory that will be used throughout the manuscript.

1.1 Combinatorial optimization

Combinatorial Optimization is a branch of operations research related to the computer science and applied mathematics. It aims to study optimization problems where the set of feasible solutions is discrete or can be reduced to a discrete one. Combinatorial optimization deals with problems that can be formulated as follows. Let $E = \{e_1, \ldots, e_n\}$ be a finite set called *basic set*, where with each element e_i is associated a weight $c(e_i)$. Let $\mathcal{F}$ be a family of subsets of E. If $F \in \mathcal{F}$, then $c(F) = \sum_{e_i \in F} c(e_i)$ is the weight of F. The problem consists in finding an element F^* of $\mathcal{F}$ whose weight is minimum or maximum. The set $\mathcal{F}$ represents the set of feasible solutions of the problem.

The term *optimization* means that we are looking for the best feasible solution among the elements of $\mathcal{F}$. The term *combinatorial* refers to the discrete structure of $\mathcal{F}$. In general, this structure is related to a discrete underlying one, which is, most of the time a graph.

It is also worth to mention that, in general, the number of feasible solutions $|\mathcal{F}|$ is exponential, which makes it difficult or even impossible to solve the associated combinatorial optimization problem with an enumerative procedure. Such a problem is hence considered as a hard problem.

Effective methods have therefore been developed to formulate and solve this type of problems. In the literature, we find various methods to solve combinatorial optimization problems such as graph theory, linear and non-linear programming, integer programming, etc. In particular, polyhedral approaches have proved to be powerful for optimally solving these problems. This will be detailed in further sections of the chapter.

During the last decades, combinatorial optimization has developed considerably from both theoretical and practical points of view. Indeed, many real-world problems from areas as diverse as transport, telecommunications, biology, VLSI circuit and statistical physics have been formulated and solved using efficient combinatorial optimization techniques.

These techniques have been proved to be effective from a complexity point of view. And this shows that combinatorial optimization is closely related to other fundamental theories, especially algorithmic and complexity theories, issues that will be discussed in the next section.

1.2 Algorithmic and complexity theory

The interest to computational theory and complexity began with the works of Cook [30], Edmonds [44] and Karp [83]. Algorithmic and complexity theory is a branch of computer science whose objective is to classify problems according to their inherent difficulty. In particular, problems of combinatorial optimization are considered as either ”easy” or ”difficult” problems. For more details on this topic, the reader is referred to [56].

A *problem* is a question to which we wish to find an answer. This question usually depends on some input parameters. A problem is posed by giving a list of these parameters as well as the properties that these parameters must satisfy. An *instance* of a problem is obtained by giving specific values to all its input parameters. An *algorithm* is a sequence of elementary operations that, when given an instance of a problem as input, gives the solution of this problem as output. The number of input parameters necessary to describe an instance of a problem is called the *size* of that problem.

An algorithm is said to be in $O(f(n))$ if there exists $c > 0$ and $n_0 \in \mathbb{N}$ such that the number of elementary operations that are necessary to solve an instance of size n is at most $c.f(n)$ for all $n \geq n_0$. If f is a polynomial function, then the algorithm is said to be polynomial. A problem belongs to the *class* P if, for each instance of the problem, there exists an algorithm that is polynomial in the size of the instance, allowing the resolution of the problem. Problems belonging to class P are said to be *easy*.

A *decision problem* is a question concerning the existence, for a given instance, of a configuration such that this configuration satisfies some properties. In other words, the solution to a decision problem can be one of the answers: *yes* or *no*. Let $\mathcal{P}$ be a decision problem and $\mathcal{I}$ the corresponding instances whose answer is yes. $\mathcal{P}$ belongs to the class NP (Non-deterministic Polynomial) if there exists a polynomial algorithm allowing to check if the answer of each instance of $\mathcal{I}$ is yes. It is clear that the class P is contained in the class NP (see Figure 1.1). And, in reality, the difference between P and NP has never been proved, however the conjecture is considered highly probable.

Among the problems that belong to the class NP, some problems are classified in a class called *NP-complete*. The NP-completeness is based on the notion of polynomial reduction. A decision problem P_1 is polynomially reduced to a decision problem P_2 if there exists a polynomial function f, such that for each instance I of P_1, the answer is yes if and only if the answer of $f(I)$ for P_2 is yes as well. This will be denoted by $P_1 \alpha P_2$. A problem P is said to be *NP-complete*, if it belongs to the class NP and if

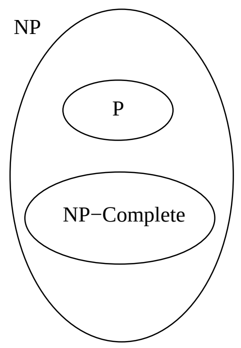


Figure 1.1: Relations between P, NP and NP-Complete

there exists a problem Q, known to be NP-complete, such that $Q\alpha P$. In practice, this theory was first used by Cook [30] who proves that SAT (the Satisfiability Problem) is NP-complete.

With every optimization problem is associated a decision problem. Moreover, every optimization problem whose associated decision problem is NP-complete is called *NP-hard*. Note that most of the combinatorial optimization problems are NP-hard.

Among the methods used to solve them, the polyhedral approach has been shown very efficient. This method is discussed in the following section.

1.3 Polyhedral approach and Branch-and-Cut

1.3.1 Elements of the polyhedral theory

Pioneered by the work of Jack Edmonds [45] for the matching problem, polyhedral approaches have shown to be powerful techniques for formulating, analysing and solving hard combinatorial optimization problems. These techniques consist in reducing the resolution of a combinatorial optimization problem to the resolution of a linear program, and this by describing (completely or partially) the convex hull of its solutions using a linear system of inequalities. This may often lead to polynomial time algorithms providing exact or approximate solutions, help efficiently solve hard combinatorial problems and provide nice structural min-max relations.

In this section, we present only the basic notions for polyhedral theory. For a deeper study of this approach, the reader is referred to the works of Grötschel et al. [67], Schrijver [123] and Mahjoub [97].

Let $n \in \mathbb{N}$ be a positive integer and $x \in \mathbb{R}^n$. We say that x is a *linear combination* of $x_1, \ldots, x_k \in \mathbb{R}^n$, if there exist k scalar $\lambda_1, \lambda_2, \ldots, \lambda_k$ such that $x = \sum_{i=1}^{k} \lambda_i x_i$. If $\sum_{i=1}^{k} \lambda_i = 1$, then x is said to be an *affine combination* of $x_1, \ldots, x_k$. Moreover, if $\lambda_i \geq 0$ for all $i \in \{1, \ldots, k\}$ with $\sum_{i=1}^{k} \lambda_i = 1$, we say that x is a *convex combination* of $x_1, \ldots, x_k$.

Given a set $S = \{x_1, \ldots, x_k\} \in \mathbb{R}^{n \times k}$, the *convex hull* of S is the set of points $x \in \mathbb{R}^n$ which are convex combination of $x_1, \ldots, x_k$ (see Figure 1.2), that is

$$conv(S) = \{x \in \mathbb{R}^n | x \text{ is a convex combination of } x_1, \ldots, x_k\}.$$

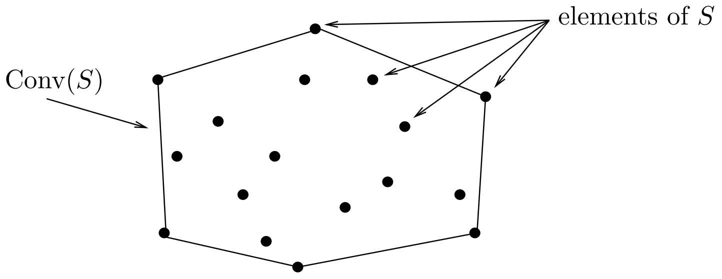


Figure 1.2: A convex hull

The points $x_1, \ldots, x_k \in \mathbb{R}$ are *linearly independent* if the unique solution of the system $x = \sum_{i=1}^{k} \lambda_i x_i = 0$ is $\lambda_i = 0$, $i = 1, \ldots, k$.

They are *affinely independent* if the unique solution of the system

$$x = \sum_{i=1}^{k} \lambda_i x_i = 0, \ \sum_{i=1}^{k} \lambda_i = 1,$$

is $\lambda_i = 0$, $i = 1, \ldots, k$.

A *polyhedron* P is the set of solutions of a linear system $Ax \leq b$, that is $P = \{x \in \mathbb{R}^n | Ax \leq b\}$, where A is an m-row n-columns matrix and $b \in \mathbb{R}^m$. A *polytope* is a bounded polyhedron. A point x of P will be also called a *solution* of P.

A polyhedron $P \subseteq \mathbb{R}^n$ is said of *dimension* p if the maximum number of solutions of P that are affinely independent is $p + 1$. We denote by $dim(P) = p$. We also have that $dim(P) = p - rank(A^=)$ where $A^=$ is the submatrix of inequalities of A that are satisfied with equality by all the solutions of P (implicit equalities). The polyhedron P is said to be full dimensional if $dim(P) = n$.

An inequality $ax \leq \alpha$ is *valid* for a polyhedron $P \subseteq \mathbb{R}^n$ if for every solution $\overline{x} \in P$, $a\overline{x} \leq \alpha$. This inequality is said to be *tight* for a solution $\overline{x} \in P$ if $a\overline{x} = \alpha$. The inequality $ax \leq \alpha$ is *violated* by $\overline{x} \in P$ if $a\overline{x} > \alpha$. Let $ax \leq \alpha$ be a valid inequality for the polyhedron P. $F = \{x \in P | ax = \alpha\}$ is called a *face* of P. We also say that F is a *face induced by* $ax \leq \alpha$. If $F \neq \emptyset$ and $F \neq P$, we say that F is a *proper* face of P. If F is a proper face and $dim(F) = dim(P) - 1$, then F is called a *facet* of P. We also say that $ax \leq \alpha$ induces a facet of P or is a *facet defining* inequality.

If P is full dimensional, then $ax \leq \alpha$ is a facet of P if and only if F is a proper face and there exists a facet of P induced by $bx \leq \beta$ and a scalar $\rho \neq 0$ such that $F \subseteq \{x \in P | bx = \beta\}$ and $b = \rho a$.

If P is not full dimensional, then $ax \leq \alpha$ is a facet of P if and only if F is a proper face and there exists a facet of P induced by $bx \leq \beta$, a scalar $\rho \neq 0$ and $\lambda \in \mathbb{R}^{q \times n}$ (where q is the number of lines of matrix $A^=$) such that $F \subseteq \{x \in P | bx = \beta\}$ and $b = \rho a + \lambda A^=$.

An inequality $ax \leq \alpha$ is *essential* if it defines a facet of P. It is *redundant* if the system $A'x \leq b'\}$ obtained by removing this inequality from $Ax \leq b$ defines the same polyhedron P. This is the case when $ax \leq \alpha$ can be written as a linear combination of inequalities of the system $A'x \leq b'$. A *complete minimal linear description* of a polyhedron consists of the system given by its facet defining inequalities and its implicit equalities.

A solution is an *extreme point* of a polyhedron P if and only if it cannot be written as the convex combination of two different solutions of P. It is equivalent to say that x induces a face of dimension 0. The polyhedron P can also be described by its extreme points. In fact, every solution of P can be written as a convex combination of some extreme points of P.

Figure 1.3 illustrates the main definitions given is this section.

Consider a combinatorial optimization problem $\mathcal{P}$. Let E be its basic set, $c(.)$ the weight function associated with its variables and $\mathcal{S}$ the set of its feasible solutions. Suppose that $\mathcal{P}$ consists in finding an element of $\mathcal{S}$ whose weight is maximum. The problem $\mathcal{P}$ can be hence written as $\max\{cx | x \in \mathcal{S}\}$. If $F \subseteq E$, then the 0-1 vector $x^F \in \mathbb{R}^E$ such that $x^F(e) = 1$ if $e \in F$ and $x^F(e) = 0$ otherwise, is called the *incidence vector* of F. The polyhedron $P(\mathcal{S}) = conv\{x^S | S \in \mathcal{S}\}$ is called the *polyhedron of*

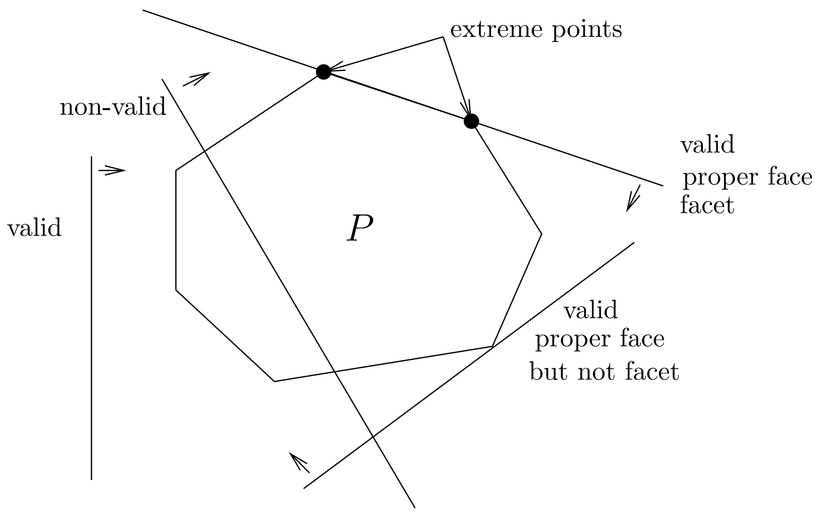


Figure 1.3: Valid inequality, facet and extreme points

the solutions of $\mathcal{P}$ or *polyhedron associated with* $\mathcal{P}$. $\mathcal{P}$ is thus equivalent to the linear program $\max\{cx|x \in P(\mathcal{S})\}$. Notice that the polyhedron $P(\mathcal{S})$ can be described by a set of a facet defining inequalities. And when all the inequalities of this set are known, then solving $\mathcal{P}$ is equivalent to the resolution of a linear program.

Recall that the objective of the polyhedral approach for combinatorial optimization problems is to reduce the resolution of $\mathcal{P}$ to that of a linear program. Generally, it is difficult to characterize a polyhedron of a combinatorial optimization problem by a system of linear inequalities. In particular, when the problem is NP-hard there is a very little hope to find such a characterization. In addition, the number of inequalities describing this polyhedron is in general exponential. Therefore, even if we know the complete description of that polyhedron, its resolution remains in practice a hard task because of the large number of inequalities.

Fortunately, a technique called the *cutting plane method* can be used to overcome this difficulty. This method is described in what follows.

1.3.2 Cutting plane method

The cutting plane method is based on a crucial result in combinatorial optimization saying that only a partial description of the polyhedron can be sufficient to solve the problem optimally.

This result comes thanks to the work of Grötschel et al. [67] (1981) who show that the difficulty of solving a linear program does not depend on the number of inequalities of that program, but on the *separation problem* associated with the inequality system of the program. Consider a polyhedron P in $\mathbb{R}^n$ and let $Ax \leq b$ be its system of inequalities. The separation problem associated with P consists in checking if the point $\overline{x} \in \mathbb{R}^n$ satisfies all the inequalities $Ax \leq b$ and, if not, to find an inequality $ax \leq \alpha$ of $Ax \leq b$ violated by $\overline{x}$ (see Figure 1.4).

Grötschel, Lovász and Schrijver [67] prove that an optimization problem (for instance $\max\{cx, Ax \leq b\}$) can be solved in polynomial time if and only if the separation problem associated with $Ax \leq b$ is polynomial as well. This equivalence has permitted an important development of the polyhedral methods in general and the cutting plane method in particular.

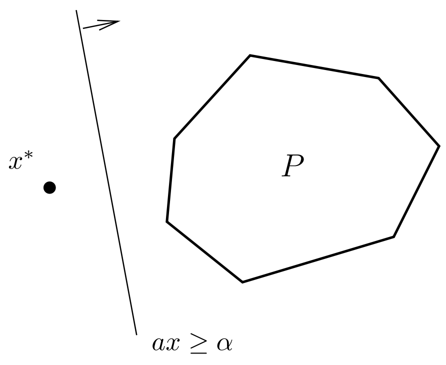


Figure 1.4: A hyperplan separating x^* and P

More precisely, the cutting plane method consists in solving successive linear programs, with possibly a large number of inequalities, by using the following steps. Let $LP = \max\{cx, Ax \leq b\}$ be a linear program and LP' a linear program obtained by considering a small number of inequalities among $Ax \leq b$. Let x^* be the optimal solution of the latter. We solve the separation problem associated with $Ax \leq b$ and x^*. This phase is called the *separation phase*. If every inequality of $Ax \leq b$ is satisfied by x^*, then x^* is also optimal for LP. If not, let $ax \leq \alpha$ be an inequality violated by x^*. Then we add $ax \leq \alpha$ to LP' and repeat this process until an optimal solution is found. Algorithm 1 summarizes the different cutting plane steps.

Note that at the end, a cutting-plane algorithm may not succeed in providing an optimal solution for the underlying combinatorial optimization problem. In this case a *Branch-and-Bound algorithm* can be used to achieve the resolution of the problem, yielding to the so-called *Branch-and-Cut algorithm*.

Algorithm 1: A cutting plane algorithm

```
Data: A linear program $LP$ and its system of inequalities $Ax \leq b$
Result: Optimal solution $x^*$ of $LP$
1 Consider a linear program $LP'$ with a small number of inequalities of $LP$;
2 Solve $LP'$ and let $x^*$ be an optimal solution;
3 Solve the separation problem associated with $Ax \leq b$ and $x^*$;
4 if an inequality $ax \leq \alpha$ of $LP$ is violated by $x^*$ then
5     Add $ax \leq \alpha$ to $LP'$;
6     Repeat step 2 ;
7 else
8     $x^*$ is optimal for $LP$;
9     return $x^*$;
```

1.3.3 Branch-and-Cut algorithm

The Branch-and-Cut method, is a combination of the Branch-and-Bound and cutting-plane methods. The basic idea of branch-and-cut is simple. In each iteration, one solves a linear relaxation of the problem using a cutting plane algorithm. New valid inequalities are then added at each iteration to the current linear program. This permits to obtain increasingly better upper bounds on the value of the optimal solution of the combinatorial optimization problem. Branching occurs only when no violated inequalities are found to cut off infeasible solutions.

Consider again the combinatorial problem $\mathcal{P}$ defined above and assume now that its variables are binary. The polyhedron $P(\mathcal{S})$ is often not completely known because $\mathcal{P}$ may be NP-hard. In this case, it would not be possible to solve $\mathcal{P}$ as a linear program and in general, the solution obtained from the linear relaxation of $P(\mathcal{S})$ is fractional. The resolution of $\mathcal{P}$ can then be done by combining the cutting plane method with a Branch-and-Bound algorithm. Such an algorithm is called a Branch-and-Cut algorithm. Each node of the Branch-and-Bound tree (also called *Branch-and-Cut tree*) corresponds to a linear program solved by the cutting plane method.

Suppose that $\mathcal{P}$ is equivalent to $\max\{cx | Ax \leq b, x \in \{0,1\}^n\}$ and that $Ax \leq b$ has a large number of inequalities. A Branch-and-Cut algorithm starts by creating a Branch-and-Bound tree whose root node corresponds to a linear program $LP_0 = \max\{cx | A_0 x \leq b_0, x \in \mathbb{R}^n\}$, where $A_0 x \leq b_0$ is subsystem of $Ax \leq b$ with a small number of inequalities. Then, we solve the linear relaxation of $\mathcal{P}$ that is $LR = \max\{cx | Ax \leq b, x \in \mathbb{R}^n\}$, using a cutting plane algorithm starting from the program LP_0. Let $x_0^* = (x_0^1, x_0^2, \ldots, x_0^k)$

be the optimal solution of LP_0 and $A'_0x \leq b'_0$ the set of inequalities added to LP_0 at the end of the cutting plane phase. If x^*_0 is integral, then it is optimal for $\mathcal{P}$. If x^*_0 is fractional, then we start the *branching phase*. This consists in choosing a variable, say x^1_0, having a fractional value and adding two nodes P_1 and P_2 in the Branch-and-cut tree. The nodes P_1 and P_2 correspond to the linear programs $LP_1 = \max\{cx|A_0x \leq b_0, A'_0x \leq b'_0, x^1_0 = 0, x \in \mathbb{R}^n\}$ and $LP_2 = \max\{cx|A_0x \leq b_0, A'_0x \leq b'_0, x^1_0 = 1, x \in \mathbb{R}^n\}$, respectively. We solve the linear program $LR_1 = \max\{cx|Ax \leq b, x^1_0 = 0, x \in \mathbb{R}^n\}$ (resp. $LR_2 = \max\{cx|Ax \leq b, x^1_0 = 1, x \in \mathbb{R}^n\}$) by a cutting plane method starting from LP_1 (LP_2). If the optimal solution of LR_1 (resp. LR_2) is integral then, it is feasible for $\mathcal{P}$. Its value is thus a lower bound of the optimal solution of $\mathcal{P}$ and the node P_1 (resp. P_2) becomes a *leaf* of the Branch-and-Cut tree. If this solution is fractional, then we select a variable with a fractional value and add two children to node P_1 (resp. P_2), and so on.

Remark that at some node of the Branch-and-Cut tree, the addition of a constraint $x^i = 0$ or $x^i = 1$ may make the associated linear program infeasible. Also, even if the corresponding linear program is feasible, its optimal solution may be worse than the best known lower bound of the tree. In both cases, we proceed the *pruning phase* and that node is cut off from the Branch-and-Cut tree. The algorithm ends when all the nodes have been explored and all the leaves of the tree are pruned. At the end of the algorithm, the optimal solution of $\mathcal{P}$ is the best feasible solution among the solutions obtained along the Branch-and-Bound tree.

Figure 1.5 illustrates a Branch-and-Cut tree. That is a Branch-and-Bound tree where in each node $Pi, i = 1, \ldots, 4$, a cutting plane method is used to solve the linear relaxation of node P_i.

The polyhedral approach and in particular the Branch-and-Cut method have been successfully applied to several combinatorial optimization problem that are considered difficult to solve, such as the Travelling Salesman Problem [8], the Max-Cut problem [21] and the Survivable Network Design Problem [86]. The efficiency of this approach depends on two important theoretical and practical issues. The first one consists in determining a good partial description of the convex hull of the solutions of the problem in terms of linear inequalities. The second issue is to devise efficient separation algorithms (exact or heuristic) for the identified classes of inequalities.

Note that the cutting plane method is effective when the number of variables is polynomial. However, when the number of variables is huge (for example exponential), one should resort to other appropriate methods such as the column generation method that we describe briefly in the following section.

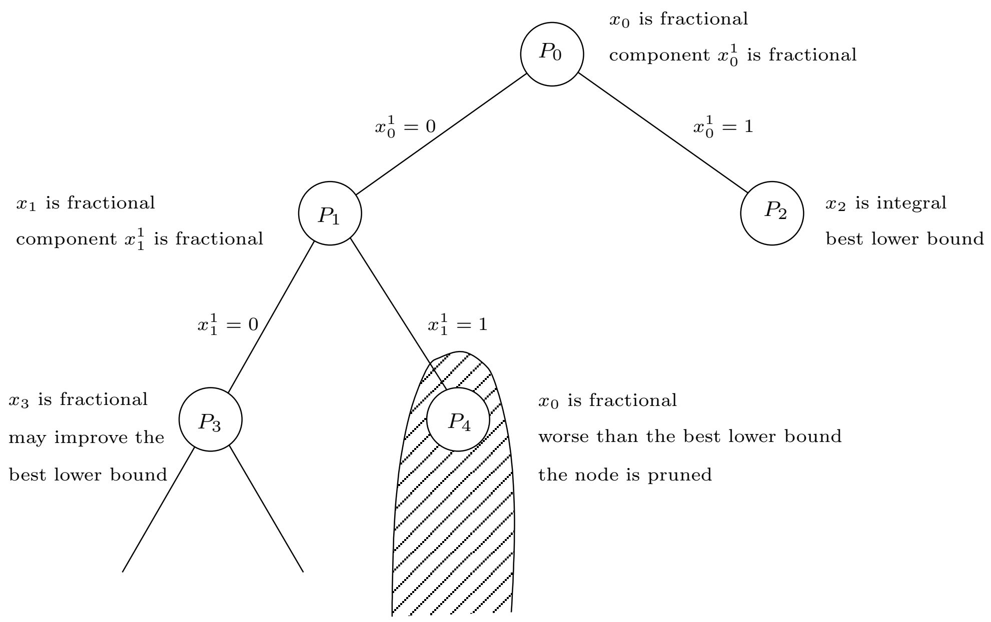


Figure 1.5: A Branch-and-Cut tree

1.4 Column generation and Branch-and-Price

There are several reasons that motivate the MIP formulations using a huge number of variables. Indeed, generally a compact formulation of a MIP have a weak LP relaxation, which can be tightened by a reformulation that involves a huge number of variables. Moreover, in some cases, a compact formulation of a MIP may have a symmetric structure which causes a poor performance of the branch-and-bound since the problem barely changes after branching. Consequently, a reformulation with a huge number of variables can eliminate this symmetry.

To solve such large-scale problems, the technique widely applied is the one called *column generation method.*

1.4.1 Column generation method

This method is used to efficiently solve linear programs with an exponential number of variables by considering only a small number of them. The method appears in the 1960's to solve problems related to huge data that could not be stored in the computers at this time. Dantzig and Wolfe [36] were the first to introduce this technique in their decomposition method. The *Dantzig-Wolfe decomposition* is a reformulation approach that is based on is a special form of variable redefinition. It can be represented using the concept of generating sets: for each sub-system on which the decomposition is based, one defines a finite set of generators from which each subproblem solution can be generated. The variables of the reformulation shall be the weights of the elements of these generating sets (more details concerning the Dantzig-Wolfe decomposition can be found in [132]). The column generation technique can be also used to deal with problems having a large number of variables. In this optic, Gilmore and Gomory [58, 59] used this approach for the cutting stock problem.

The general idea is that optimal solutions to large LP's can be obtained without explicitly including all columns in the constraint matrix. In fact, only a subset of all columns will be in the optimal solution and all other (non-basic) columns can be ignored. The idea of a column generation algorithm is to solve a sequence of linear programs having a reasonable number of variables (also called *columns*). The algorithm starts by solving a linear program having a small number of variables and which forms a feasible basis for the original program. At each iteration of the algorithm, we solve the so-called *pricing problem* whose objective is to determine the variables which must enter the current basis. These variables are those having a negative reduced cost (for a "minimize" objective function). The *reduced cost* associated with a variable is computed using the dual variables. We then solve the linear program obtained by adding these variables and repeat the procedure. The algorithm stops when the pricing algorithm does not generate new columns to add to the current basis. In this case, the solution obtained from the last restricted program is optimal for the original one. When the pricing problem is NP-hard, one can use some appropriate heuristic procedure to solve it. However, at the last iteration, it must be solved by an exact method to prove the optimality of the solution. Notice that the column generation method can be seen as the dual of the cutting plane method as it adds columns (variables) instead of rows (inequalities) in the linear program. For more details on the column generation method, the reader is referred to [94, 130, 38].

1.4.2 Branch-and-Price algorithm

In order to solve integer linear programs, the column generation method can be combined with a Branch-and-Bound algorithm. The obtained algorithm is called a *Branch-and-Price algorithm*. The principle of branch-and-price is similar to that of branch-and-cut except that the procedure focuses on column generation rather than row generation. Similarly to the Branch-and-Cut algorithm, the branching phase happens when no columns price out to enter the basis and the solution given by the linear program

Algorithm 2: A column generation algorithm

```
   Data: A linear program $MP$ (Master Problem) with a huge number of variables
   Result: Optimal solution $x^*$ of $MP$
1  Consider a linear program $RMP$ (Restricted Master Problem) including only a
   small subset of variables of the $MP$;
2  Solve $RMP$ and let $x^*$ be an optimal solution;
3  Solve the pricing problem associated with the dual variables obtained by the
   resolution of the RMP;
4  Denote $C = \{x|$ reduced cost $(x) < 0\}$, the set of variables (columns) with a
   negative reduced cost;
5  if $C \neq \emptyset$ then
6      Add the variables of $C$ to $RMP$;
7      Repeat step 2 ;
8  else
9      $x^*$ is optimal for $LP$;
10     return $x^*$;
```

is yet fractional.

In the literature, the Branch-and-Price algorithm has been extensively used to solve large scale problems such that the cutting stock problem [6], the integer multi-commodity flow problem [18], the crew scheduling problem [129], etc...

Moreover, it is possible to combine both column and row generation. This combination is very interesting since it can yield very strong LP relaxations. However, dealing with the two generation processes is not trivial. The main difficulty is, after additional rows are added, the structure of the pricing problem can be destroyed and its resolution becomes much harder.

Despite these difficulties, there have been some successful applications of combined row and column generation (see for example [17]).

1.4.3 Primal heuristics

The Branch-and-Price (resp. Branch-and-Cut) algorithm can be improved by deriving good primal feasible solutions to the combinatorial optimization problem. This can be achieved using the so-called *primal heuristics*, which compute good lower bounds that can be used to prune suboptimal branches of the Branch-and-Price tree (resp. Branch-and-Cut tree).

Primal heuristics can be used at the root to find early a first feasible solution. They also may be used at a given node of the tree mainly to round fractional solutions and try to get a better bound. As a consequence, they help reducing considerably the number of generated nodes of the tree as well as the CPU time. Moreover, this guarantees to have an approximation of the optimal solution of the problem for example when a CPU time limit has been reached.

1.5 Extended Formulations

To formulate a Combinatorial Optimization Problem $\mathcal{P}$, the natural way is to define an integer variable x_e for each $e \in E$ and find a suitable set of constraints to represent $\mathcal{F}$. Recall that E represents the basic set of $\mathcal{P}$ and $\mathcal{F}$ is a family of subsets of E. A formulation using only variables x_e is called *natural formulation*, and can be written as follows.

$$\begin{aligned} &\text{Min} \sum_{e \in E} c(e) x_e \\ &\text{s.t.} \quad Ax \geq b && (1.1) \\ &\quad x_e \text{ integer} \quad \text{for all } e \in E. && (1.2) \end{aligned}$$

Although a natural formulation is a direct formulation using generally a reasonable number of variables, it can present some drawbacks. In fact, without adding valid inequalities, natural formulations usually have a loose linear relaxation. As a consequence, to improve natural formulations one should resort to new classes of valid inequalities defining in preference facets. This may imply separation procedures that can be heavy to compute, mainly when the associated separation problem is NP-hard. Moreover, natural formulations often have a number of constraints that is exponential in the size of the problem.

An attractive alternative to overcome these drawbacks is to introduce additional variables, and thus work in a higher dimensional space. The resulting formulation is called *extended formulation*. In what follows, we give some preliminary notions about extended formulations. For more details, the reader is referred to [128, 29].

We distinguish *exact* and *compact* extended formulation. By exact, we mean that the projection onto the original space of variables gives the convex hull of the solutions in this space. This implies that the value of the dual bound is equivalent to that obtained when carrying out separation over the convex hull of the solutions. Also by compact, we mean that the number of variables and constraints of the extended formulation is polynomial in the size of the problem. Hence, theoretically, adding a priori such an extended formulation to tighten a mixed-integer model provides an alternative to the separation.

An extended formulation can be defined with respect to a given original formulation (a natural formulation or an already extended one) as follows. Suppose that there are n original variables x and p extended variables y. Suppose there are m constraints, that may involve only x variables, only y variables, or both sets of variables. We also assume that the cost function and the integrality requirements are only on the x variables, which is reasonable, because the original formulation could do that.

$$\begin{aligned} \text{Min} \quad & \sum_{e \in E} c(e) x_e \\ \text{s.t.} \;\; & Ax + By \geq b & (1.3) \\ & x \text{ integer.} & (1.4) \end{aligned}$$

An extended formulation can be compared with a formulation on the original variables by projecting its associated polyhedra onto the x space. Let $Q = \{(x, y) \in R^n R^p : Ax + By >= b\}$. The projection of Q onto x is defined as $proj_x(Q) = \{x \in R^n : \exists y \in R^p, Ax + By >= b\}$.

The use of extended formulations have many potential advantages. In fact, as mentioned above, extended formulations have the potential for providing high-quality bounds on some combinatorial optimization problems where the natural formulations perform poorly. In addition, some very complex families of facet-defining inequalities on the natural variables can be obtained by the projection of quite simple inequalities on the extended variables. Moreover, an inequality on the natural variables, even when facet-defining, can generally be lifted when expressed in the extended space of variables. Furthermore, when the separation of a certain family of inequalities in the original variables is NP-hard, a superfamily of valid inequalities obtained by a projection from an extended formulation can be separated in polynomial time.

However, extended formulations may present some drawbacks. First, because for some combinatorial optimization problems, stronger extended formulations are significantly huge as using a large number of variables. Moreover, sometimes the linear programs of large extended formulations can be highly degenerate, mainly when using the simplex algorithm to solve the linear relaxation.

1.6 Graph theory: definitions and notations

In this section, we present some basic definitions and notations of graph theory that will be necessary for the subsequent chapters. For more details, the reader is referred to [123].

The graphs we consider are either directed or undirected, finite, loopless and without multiple edge or arcs.

An undirected graph is denoted by $G = (V, E)$ where V is the set of nodes and E is the set of edges. If $e \in E$ is an edge with endnodes u and v, we also write uv to denote e. Given W and W', two disjoint subsets of V, $[W, W']$ denotes the set of edges of G having one endnode in W and the other in w'. If $W' = \overline{W}$, then $[W, \overline{W}]$ is called a *cut* of G denoted by $\delta(W)$ (see Figure 1.6). For a node subset $W \subseteq V$ and a node $v \in V$, we write $[v, W']$ for $[\{v\}, W']$ and we denote by $\overline{W}$ the node set $V \setminus W$.

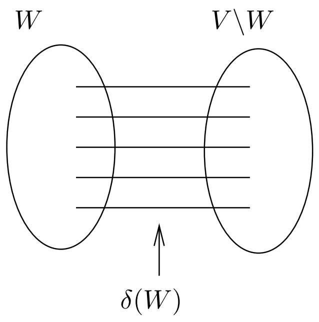


Figure 1.6: A cut $\delta(W)$

If $\Pi = (V_1, \dots, V_p)$, $p \geq 2$, is a partition of V, then we denote $\Delta = \delta(\Pi)$ the set of edges having their endnodes in different sets. We may also write $\delta(V_1, \dots, V_p)$ for $\delta(\Pi)$. Note that for $W \subset V$, $\delta(W) = \delta(W, \overline{W})$. If s and t are two nodes of G such that $s \in W$ and $t \in \overline{W}$, then $\delta(W)$ is called *st-cut* of G.

A directed graph is denoted by $D = (V, A)$ where V is the node set and A the arc set. An arc a with origin u and destination v is denotes by (u, v). Given two node subsets W and W' of V, $[W, W']$ denotes the set of arcs whose origins are in W and destinations in W'. As before, we write $[u, W']$ for $[\{u\}, W']$ and $\overline{W}$ denotes the node set $U \setminus W$. The set of arcs having their origins in W and destinations in $\overline{W}$ is called a *directed cut* or *dicut* of D. This arc set is denoted either by $\delta^+(W)$ or $\delta^-(\overline{W})$. We also write $\delta^+(u)$ for $\delta^+(\{u\})$ and $\delta^-(u)$ for $\delta^-(\{u\})$ with $u \in U$. If s and t are two nodes of D such that $s \in W$ and $t \in \overline{W}$, then $\delta^+(W)$ and $\delta^-(\overline{W})$ are called *st-dicuts* of H.

Let $G' = (V', E')$ (resp. $D' = (V', A')$) with $V' \subseteq V$ and $E' \subseteq E$ (resp. $V' \subseteq V$ and $A' \subseteq A$) be a subgraph of G (resp. D). If $c(.)$ is a weight function which associates with an edge (resp. arc) $e \in E$ (resp. $a \in A$) the weight $c(e)$ (resp. $c(a)$), then the total weight of G' (resp. D') is $c(E') = \sum\limits_{e \in E'} c(e)$ (resp. $c(A') = \sum\limits_{a \in A'} c(a)$).

In the notations, we will specify the graph as a subscript (that is, we will write $\delta_G(W)$, $\delta_G(\Pi)$, $\delta_G(V_1, \dots, V_p)$, $\delta^+_D(W)$, $\delta^-_D(W)$, $[W, W']_G$, $[W, W']_D$) whenever the considered graphs may not be clearly deduced from the context.

Given an undirected graph $G = (V, E)$, for all $F \subseteq E$, $V(F)$ will denote the set of nodes incident to the edges of F. For $W \subset V$, we denote by $E(W)$ the set of edges of G having both endnodes in W and $G[W]$ the subgraph induced by W, that is the graph $(W, E(W))$. Given an edge $e = uv \in E$, *contracting* e consists in deleting e, identifying the nodes u and v and in preserving all adjacencies. Contracting a node subset W consists in identifying all the nodes of W and preserving the adjacencies. Given a partition $\Pi = (V_1, \dots, V_p)$, $p \geq 2$, we will denote by G_Π the subgraph induced by Π, that is, the graph obtained from G by contracting the sets V_i, for $i = 1, \dots, p$. Note that the edge set of G_Π is the set $\delta(V_1, \dots, V_p)$.

A *path* P of an undirected graph G is an alternate sequence of nodes and edges $(u_1, e_1, u_2, \dots, u_{l-1}, e_l, u_l)$ where $e_i = [u_i, u_{i+1}]$ for $i = 1, \dots, l-1$. We will denote a path P either by its node sequence $(u_1, \dots, u_l)$ or its edge sequence $(e_1, \dots, e_{l-1})$. The nodes u_1 and u_l are called the *endnodes* of P, while its other nodes are said to be *internal*. A path is *simple* if it does not contain the same node twice. In the sequel, we will always consider that the paths are simple. By opposition, a non-simple path is called a *walk*. A path whose endnodes are s and t will be called an *st-path*. A *cycle* in G is a path whose endnodes coincide, that is $u_1 = u_l$. Also, a cycle is simple if it does not contain twice the same node, excepted u_1. We call a *chord* an edge between two non-adjacent nodes of a path.

Similarly, we call a *dipath* P a path in a directed graph, that is an alternate sequence of arcs $(u_1, a_1, u_2, \dots, u_{l-1}, a_l, u_l)$ with $a_i = [u_i, u_{i+1}]$ for $i = 1, \dots, l-1$. A dipath is denoted either by its node sequence $(u_1, \dots, u_l)$ or its arc sequence $(a_1, \dots, a_{l-1})$, and the nodes u_1, u_l are called the endnodes of that dipath. A dipath whose endnodes coincide ($u_1 = u_l$) is called a *circuit*. If $u_1 = s$ and $u_l = t$ then P is called an *st-dipath*. A dipath is simple if it does not contain twice the same node.

An undirected (resp. directed) graph is said *connected* if for every pair of nodes (u, v) there exists at least one path (resp. dipath) between u and v. A connected graph which have no cycle (resp. circuit) is called *tree*. A *connected component* of a graph G (resp.

D) which is maximal, that is adding a node or an edge (resp. arc) to that subgraph gives a non-connected graph.

Given an undirected (resp. directed) graph $G = (V, E)$ (resp. $D = (V, A)$), two st-paths (resp. st-dipaths) are said to be *edge-disjoint* (resp. *arc-disjoint*) if they have no edge (resp. arc) in common. They are *node-disjoint* if they have no internal node in common. A graph is said to be *k-edge-connected* (resp. *k-arc-connected*) if it contains at least k edge-disjoint (resp. arc-disjoint) st-paths (resp. st-dipaths) for all pair of node $\{s, t\} \in V \times V$ (resp. $\{s, t\} \in V \times V$).

Given an undirected graph $G = (V, E)$, a demand set $K \subseteq V \times V$ is a subset of pairs of nodes, called *demands*. For a demand $\{O, D\} \in K$, O is the *origin* and D is the *destination* of that demand. A node involved in a path P routing a demand k will be called a *terminal* for that demand, while the other nodes are rather *Steiner nodes* for the considered demand.

A *complete graph* is a graph in which there is an edge between each pair of nodes. A complete graph with n nodes is denoted by K_n.

An undirected graph is *outerplanar* when it can be drawn in the plane as a cycle with non crossing cords.

A graph is *series-parallel* if it can be obtained from a single edge by iterative application of the two operations: addition of a parallel edge, subdivision of an edge.

Observe that a graph if series-parallel if and only if it is not contractible to K_4. Similarly, a graph is outerplanar if and only if it is not contractible to K_4 and $K_{3,2}$. Therefore, an outerplanar graph is also series-parallel.

A graph G is said to be *Halin graph* if $G = (C \cup T, E)$ where the subgraph of G induced by T is a tree whose leaves forms the cycle C in G.

Chapter 2

Multilayer telecommunication networks

Contents

Nowadays, telecommunication networks provide a large variety of services enabling the transport of different kind of data such as vocal messages, photos, videos, etc. In addition to this variety, exchanged data have witnessed an explosive growth due mainly to the fast development of Internet. As a consequence, telecommunication networks have been evolving towards a multilayer architecture, which has proved to be the most powerful and the most adapted to all these changes and improvements. Moreover, as *these networks are involved in the transportation of important data for companies, administrations and governments, they ought to be sufficiently survivable, so that network services can be restored in the event of a catastrophic failure.*

In this chapter, we give a brief idea about the functionalities of multilayer networks. We first present the multilayer structure in telecommunication networks and focus especially on the IP-over-WDM model. More precisely, we present the particularities of the IP and WDM layers and discuss the technologies corresponding to each one. Then, we show the different ways of interaction between the two layers and present in particular the GMPLS technology. We further introduce some concepts related to survivability in general and to its application in the multilayer context. We finish the chapter by a quick survey on multilayer network design problems and in particular the ones dealing with survivability issues.

2.1 Telecommunication networks: toward a multilayer structure

2.1.1 Evolution of networks' architecture

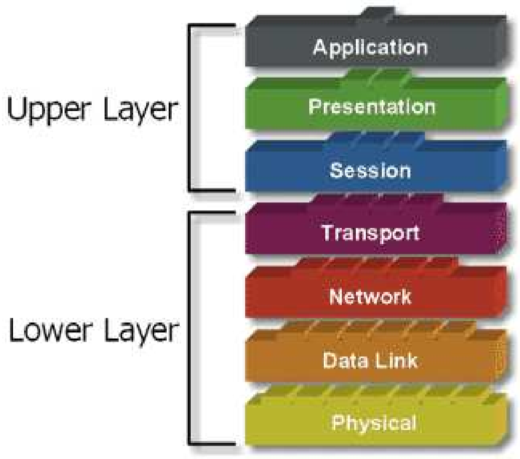


Figure 2.1: Reference Model OSI

For historical reasons, telecommunication networks have been represented as the superposition of several technological layers. Using the bottom technology, each layer has a specific functionality that provides a service for the layer above. Moreover, each layer is characterized by appropriate protocols. A protocol can be defined as a formal description of the conventions and rules that are used by a layer to manage data traffic and govern the interactions with the other layers. In order to classify the different protocols, the ISO (International Standardization Organization) proposed a seven-layer model known as the OSI (Open Systems Interconnection) model (see Figure 2.1).

Although the OSI model constitutes a reference that helped to well understand the multilayer network process, it remains significant only from a theoretical point of view. In reality, one should expect to have less than seven layers such that each layer can ensure several functions at a time.

In a further classification, the network has been represented using IP routers over ATM switches over SDH network components. The IP (Internet Protocol) layer is used as a platform for users' applications, ATM (Asynchronous Transfer Mode) for traffic engineering, flow control and QoS (Quality of Service) support, and SDH (Synchronous Digital Hierarchy) for the transportation of ATM flows over WDM optical network (Wavelength Division Multiplexing) [57] (see Figure 2.2).

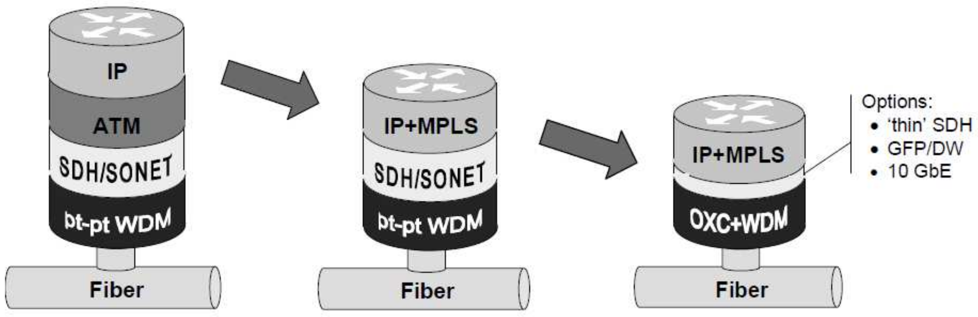


Figure 2.2: Evolution towards the IP-over-WDM model

Again, the resulting architecture creates more complications rather that simplifying the network. Due to the number of layers involved, this architecture stacks redundant functionalities and presents many operational complexities. In fact, when delivering data, each layer adds heavy control information (for encapsulation), yielding to an over-cost bandwidth and a very complex data processing over the nodes. Furthermore, this four-layer architecture shows a great lack of flexibility that impedes it to cope with the continuous increase of data traffic [93]. Moving towards a more flexible, dynamic and cost-effective architecture was consequently necessary.

The most straightforward alternative was to bypass all intermediate layer technologies and implement the IP directly over WDM, hence resulting in a simple two layer model, referred to as *IP-over-WDM* [93, 124].

The IP-over-WDM model presents many advantages. In fact, WDM technology can manage the continuous growth of traffic using the already existing infrastructure. This is thanks to the huge capacities of optical fibres which may ensure the transportation of many terabits per second. Moreover, since telecommunication operators choose to put all the different types of traffic (voice, data and video) on a single physical medium, the majority of traffic has become of type IP.

Nevertheless, the implementation of this new two layer model has been a bit challenging. In fact, the deleted intermediate layers provide important functionalities, such as traffic engineering in ATM or multiplexing and fast protection in SDH. Moving to the two layers IP-over-WDM model, these functionalities had to be continuously ensured by moving them either down to the optical layer, or up to the IP layer.

To overcome this difficulty, telecommunication operators had the idea of using the control protocols such as the MPLS (Multi-Protocol Label Switching) and the GMPLS (Generalized-MPLS) [115, 100]. These protocols permit the implementation of the traffic engineering in the IP layer (at the packet level) and in the optical layer (at the wavelength level), and hence the ATM can be removed from the network. Similarly, many functions of the SDH can be moved down to the optical layer, enhanced with optical switching capabilities and supported by advanced control mechanisms, now referred to as the OTN (Optical Transport Network) [134].

2.1.2 The IP layer

The role of the Internet layer is to ensure the interoperability and interconnection between the different AS (Autonomous System) subnetworks constituting it, allowing thus data to be routed through these networks until destination. To this end, it uses the IP protocol which is responsible of data routing.

2.1.2.1 IP Protocol

To be routed in the network, the data is divided into IP packets. An IP packet is composed of two parts: a header having information for various transport, in particular the destination IP address, and a data portion. The Internet Protocol (IP) [48, 81, 108] manages the transmission of these packets called datagrams through a set of interconnected networks from a source to a destination. The source and destination are host machines (terminals) identified by fixed length addresses called IP address. Many other protocols such as the TCP (Transmission Control Protocol) are used to complete the packet headers in order to ensure a correct packets' routing and a rapid detection of errors [109].

2.1.2.2 Traffic routing

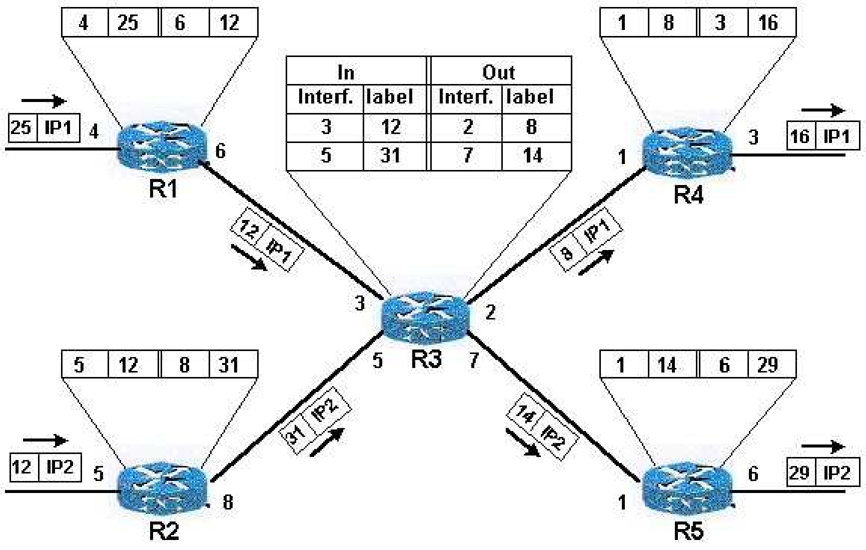
Traffic routing is one of the main functions of the Internet layer. The routing consists in determining a path in order to connect two distant terminals. Determining a path is a complex task that is performed in large networks using dedicated protocols. The role of these protocols is to discover the network topology and derive the best route. There are several routing protocols that differs in the criteria of choosing routes and the accuracy of the topology discovery. We distinguish for example, the RIP (Routing Information Protocol) that allows each router to know the distance separating the other routers from the IP network in term of hops. There is also the OSPF (Open Shortest Path First) protocol, which, being more efficient than RIP, has gradually replaced it. In fact, unlike RIP, the OSPF sends to all the adjacent routers the number of hops that separates them from the IP networks. Then, each router transmits to all the routers in the network the status of each of its links. A third protocol is the so-called IS-IS (Intermediate System to Intermediate System). The IS-IS is a link-state protocol that enables routers to have maintain a common view of network's topology. Each router makes its own topological database and then shared among the remaining routers. Packets are then transmitted via the shortest path. Finally, we can mention the BGP (Border Gateway Protocol) that is used to convey large volumes of data through the network.

The majority of routing protocols rely on traditional algorithms calculating shortest paths in the network. However, they do not generally take into account other criteria such as delays or congestion, which can degrade seriously the performance of the network. In this optic, telecommunication operators have been always seeking a good and optimal way to manage their networks. In particular, in order to ensure a better routing procedure, they thought of the introduction of the MPLS system, that will be presented in the next section.

2.1.2.3 MPLS technology

In traditional IP networks, packets' routing is performed using the destination address contained in the header of the IP layer's packet. To decide which is the next hop, each router of the IP layer consults its routing table and determines the outgoing interface to which the packet will be sent. Generally, the mechanism of research in the routing table is time consuming. Moreover, with the growth of the networks' size in the recent years, the routing tables' size has been steadily increasing, making it necessary to find a more efficient method to route packets. Telecommunication operators have then opted for the MPLS (Multi Protocol Label Switching) technology.

MPLS is a multilayer switching system that was inspired from Tag Switching technologies of Cisco and also from the ARIS (Aggregate Route-Based IP Switching) technology of IBM. MPLS was designed to improve the efficiency of routers in packet processing. Indeed, instead of being analysed at each router, the packets are analysed only once at the entrance of the network, and then routed along a path thanks to a system of labels. As its name indicates, MPLS is based on the technique of label switching. Figure 2.3 illustrates this mechanism. For example, an IP packet entering to the router $R3$ through the interface 5, with the label 31 will after be sent to the router $R5$ using interface 7 and the outgoing packet will be labeled 14 instead of 31. Note here that the forwarding table of labels has far fewer entries than the usual IP routing table. The switching procedure is hence done by routers that do not need to consult the IP address or the routing table. These routers are called LSR (Label Switch Routers).



In		Out	
Interf.	label	Interf.	label
3	12	2	8
5	31	7	14

Figure 2.3: An example of label switching

The LSR use the labels to switch packets inside the MPLS network (see Figure 2.4). MPLS routers at the periphery of the network are called Edge LSR. These routers have both traditional and IP interfaces connected to the MPLS network, and are responsible for imposing or removing labels on IP packets that pass through them. In particular, the Edge LSR can be divided into two classes: the Ingress LSR (ingress routers) responsible for imposing labels to IP packets that pass through to enter the MPLS backbone, and the Egress LSR (egress routers) that remove the added labels.

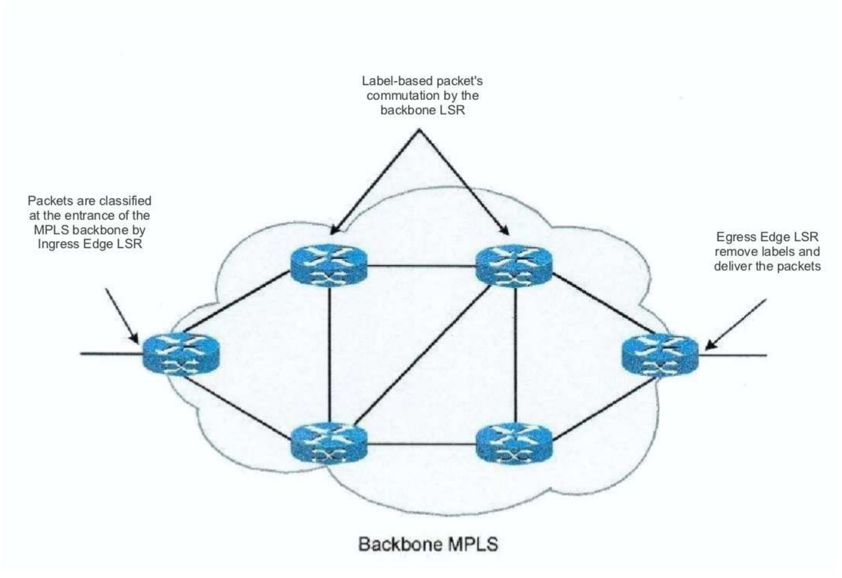

Label-based packet's commutation by the backbone LSR
Packets are classified at the entrance of the MPLS backbone by Ingress Edge LSR
Egress Edge LSR remove labels and deliver the packets
Backbone MPLS

Figure 2.4: LSR

When the packets enter the the MPLS network, Ingress LSR classify them into different classes called FEC (Forwarding Equivalent Classes). These classes can be formed according to several criteria such as the same prefix as the destination address for the IP routing, packets of the same application, packets from the same source address prefix, the quality of service required, etc. Hoewever, in general FEC are defined in terms of IP prefixes that are given by the IGP (Interior Gateway Protocol), and the information related to the quality of service insured by the so-called TE (Traffic Engineering). After the classification step, packets belonging to the same FEC will follow the same path and will be managed by the same method of forwarding. In fact, once inside the MPLS network, the packets can never be reclassified. The succession of LSR that have been used by an FEC constitutes the so-called LSP (Label Switched Path), also said the labels' switching path. This path is always unidirectional and unique for each FEC.

Originally, MPLS was developed for fast packet switching, but it has quickly allowed the implementation of higher level solutions such as VPN (Virtual Private Networks). This will be explained in details in the next section.

2.1.2.4 Virtual Private Network

A virtual private network (VPN) can be seen as an extension of an organisation private network in order to connect remote users over shared or public network. A private network is one where all data paths are secret to a certain extent, yet open to a limited group of persons. A VPN allows, for example, to interconnect networks of the same company in multiple locations.

VPNs are IP-based networks that use encryption and tunnelling to achieve the following goals:

- connect users securely to their own corporate network (remote access),
- link branch offices to an enterprise network (intranet),
- extend organizations' existing computing infrastructure to include partners, suppliers and customers (extranet).

VPNs often use the MPLS technology to connect sites belonging to one or more VPN since the MPLS LSP tunnels provide a good medium for the encapsulation of VPN traffic. These networks are called MPLS-VPN.

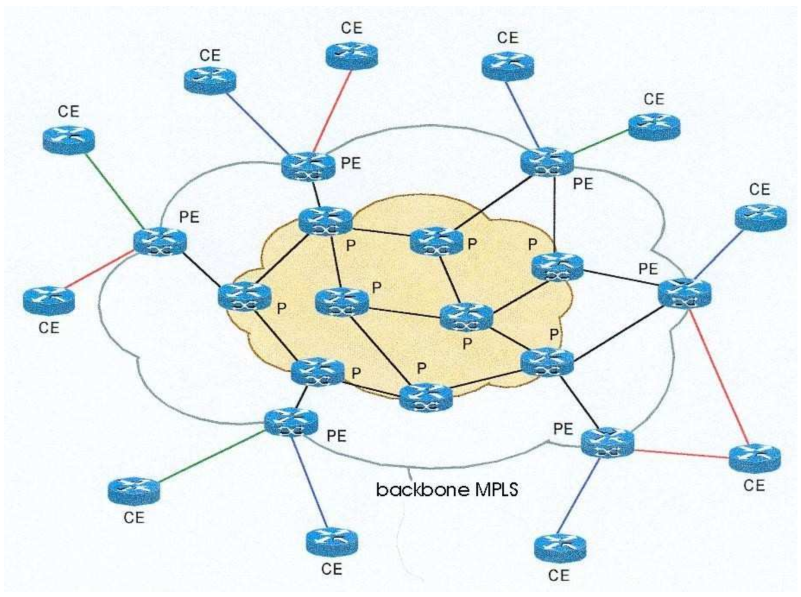


Figure 2.5: Different types of routers in an MPLS-VPN

In an MPLS-VPN, we distinguish several types of routers (see Figure 2.5):

- P (Provider): located Inside the MPLS network, they send the data by switching labels,
- PE (Provider Edge): located at the border of the MPLS network, they have interfaces that are connected to the costumers' routers,
- CE (Customer Edge): are traditional IP routers with no knowledge of MPLS.

The aim is to ensure the transmission of IP packets between CE routers belonging to the same VPN or to different VPNs.

Nowadays, companies are increasingly using private IP addressing, which allows them to use the same IP address ranges. MPLS-VPN help manage recovery plans addressing and can differentiate two identical addresses belonging to two different VPN. It hence enables traffic isolation between customer sites not belonging to the same VPN. The PEs have multiple interfaces each corresponding to a particular VPN. Each interface includes a VRF (VPN Routing and Forwarding) protocol that allows PE to manage multiple routing tables (one per interface). A VRF consists of a routing table, a FIB (Forwarding Information Base) and a CEF table, that is independent of other VRFs' tables and of the global routing table. When a PE router gets a packet on an interface IP, it checks the routing table of the VRF corresponding to this interface without even accessing to its global routing table. This ability to use multiple independent routing tables allows to manage an addressing plan for each site, even in case of recovery addresses between different VPN. To build their VRF tables, the PE must exchange routes corresponding to different VPN. To this end, they use the label distribution protocol MP-BGP (MultiProtocol BGP). The EC share their IP routes with their PE using standard routing protocols (for example eBGP, RIPv2 or OSPF).

2.1.3 The WDM layer

2.1.3.1 Optical fibres

An optical fibre is a flexible transparent fibre made of glass (silica) or plastic. It functions as a wave-guide, or light pipe, to transmit light between the two ends of the fibre. To this end, optical fibres exploit the properties of refracting light. Indeed, when a light beam strikes the surface at an angle between two environments that are more or less transparent, it splits in two. The first part is reflected while the other is refracted, that is to say, transmitted in the second medium when changing direction. Reflection is the principle used to guide the light into the optical fibre.

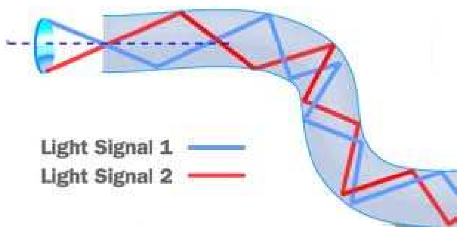


Figure 2.6: Signals' reflection

Optical fibres typically include a transparent core surrounded by a transparent cladding material with a lower index of refraction. Light is kept in the core by total internal reflection. This causes the fibre to act as a wave-guide. Fibres that support many propagation paths or transverse modes are called Multi-Mode Fibres (MMF), while those that only support a single mode are called Single-Mode Fibres (SMF).

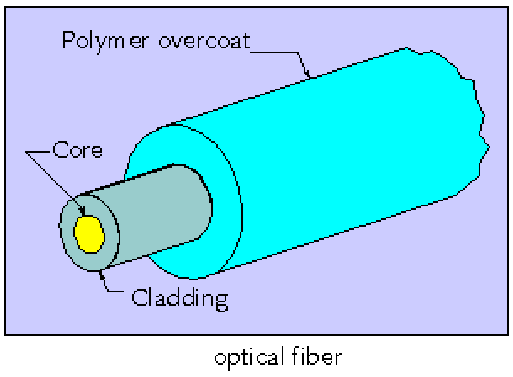


Figure 2.7: Optical fibre

Optical fibres are widely used in fibre-optic communications and this is due to many reasons. First, because they permit transmission over long distances with a low rate of attenuation. Also, compared to other medium of communication, they guarantee higher bandwidth, lower weight, smaller size and more flexibility. Moreover, from an electromagnetic point of view, optical fibre are quite immune to interference. As a consequence, they constitutes a very good choice for high-speed transmissions. Another advantage of the optical fibre is its safety aspect. Indeed, it is very difficult to connect a listening cable to an optical fibre and such an operation results in a significant drop of signal, whose cause can be easily localized. For all these advantages and in spite of a price remaining still high, the optical fibre is used in the field of telecommunications for the implementation of broadband networks. It is also used in a variety of other applications such as illumination, broadcasting, medical imaging, sensors, fibre lasers, etc.

2.1.3.2 Optical switches

The nodes of an optical network act like referral centers that allow the orientation of the different wavelengths carried on a fibre to their destination fibres, respectively. These nodes use a multiplexing technology that mix signals from different fibres. This technology allows also to extract or insert one or more signals before sending them to another optical fibre. In current networks, the number of transmitted wavelengths on a single fibre has been significantly increasing. Therefore, optical nodes have to be more efficient to be able to handle a large number of wavelengths. There are several kinds of optical nodes such as the OXCs (Optical Cross-Connect), which are divided into two types, opaque OXCs that use electronic switching matrices and transparent ones using the mirrors' reflection principle.

2.1.3.3 WDM technology

One of the crucial services that are provided by the transport layer is that of application's multiplexing and demultiplexing. This feature allows multiple applications to use simultaneously the physical network, and ensure that the transport layer can differentiate the data it receives according to the applications or processes to which the data belong. This is ensured thanks to the data packets' headers at the physical layer which is provided with a set of fields allowing to determine the process to which the data packets are to be delivered. At the receiver, these fields are examined to determine the process to which the data segment belongs, and the packet is then directed to that process. The receiver's functionality that consists in delivering the data it receives to the correct application process, is called *demultiplexing*. In parallel, the sender's functionality consists in incorporating the information it receives in the packet's header, and this is called *multiplexing*. There are several types of technologies ensuring multiplexing and demultiplexing such as PDH (Plesiochronous Digital Hierarchy), SDH (Synchronous Digital Hierarchy), WDM (Wavelength Division Multiplexing) and DWDM (Dense WDM). Here, we are interested to WDM technology.

2.1.4 Interactions between the IP and WDM layers

As it has been mentioned in the beginning of the chapter, a network consists of a number of interconnected layers such that each one is characterized by its own features. In particular, in the following we will be interested to IP-over-WDM networks consisted of a *transport layer* (or optical) that is responsible of the transfer of information between customers and a *client layer* (or IP) that manages the network command. In this section, we will be interested to the description of the strong relationship that may exist between these two layers. We introduce the three existing models of interconnection between the layers and then present the GMPLS control plane which is necessary for the implementation of two of these interconnection models.

2.1.4.1 Interconnection Model

The two-layer model which is currently advocated for tighter integration between the IP and the optical layers offers a series of advantages over an existing multilayer architecture model. We distinguish three models of interconnection between the two layers. For more details, the reader is referred to [7].

1) Overlay model

   Under the overlay model, the IP domain acts as a client to the optical domain. IP/MPLS routing and signaling protocols are independent of the routing and signaling protocols of the optical layer. The topology distribution, path computation, and signaling protocols would have to be defined for the optical domain. The client networks are provided with no knowledge of the optical network topology or resources.

2) Peer model

   Under the peer model, the IP/MPLS act as peers of the optical transport network (IP/MPLS routers and OXCs act as peers). A single routing protocol instance runs over both IP/MPLS and optical layers. A common IGP and OSPF or IS-IS may be used to exchange topology information. A consequence of the peer model is that all the optical switches and routers have a common addressing scheme.

3) Augmented model

   This model combines the best of the peer and overlay interconnection models. Under the augmented model, there are actually separate routing instances in the IP and optical domains, but the information from one routing instance is passed through the other routing instance. For example, external IP addresses could be carried within the optical routing protocols to allow reachability information to be passed to IP clients.

Over the peer and augmented models, to integrate structures between IP and optical layers, the control plane must use protocols that can well manage the interaction between the two layers, such as MPLS and GMPLS. In the following section, we discuss GMPLS.

2.1.4.2 GMPLS

There is now a general consensus on the fact that the control plane of optical networks should use the protocols and mechanisms developed for IP to automatically provide optical paths and ensure dynamic recovery. Mechanisms of signalling and routing applications developed for IP Traffic Engineering may indeed be reused in the WDM layer, taking into account the specificities of optical networks. One of these mechanisms is GMPLS. GMPLS (Generalized Multi-Protocol Label Switching) has additional features over MPLS:

- Neighbor Discovery: In order to manage the network, all devices must be known: switches, multiplexers and routers. GMPLS uses a new protocol called Link Management Protocol (LMP) to explore the equipment and negotiate functionality.
- Propagation of link states: The goal is not only to know the condition of equipment but also the link state. To propagate this information, a routing protocol should be used. For MPLS, OSPF or IS-IS have been modified to support these functions.
- Management of the link state: Routing protocols such as OSPF and IS-IS can be used to monitor and manage the topology of the link state.
- Control and management of roads: MPLS can use RSVP (Resource Reservation Protocol) to establish a link from end to end. However, if the data MPLS crossings telecom networks, other protocols must be implemented, such as UNI (User Network Interface) and PNNI (Private Network-to-Network Interface). Road management is a challenge because many standardization organization are concerned. The IETF (Internet Engineering Task Force) is currently working on extensions of LDP (Label Distribution Protocol) and RSVP protocols to enable the management and control of roads.

- Management of relationships: In MPLS, LSP (Label Switch Path) is used to establish, release and aggregating links. In GMPLS, the ability to establish and aggregating the optical channels is required. LMP (Link Management Protocol) extends MPLS functions in the optical plane to build relationships, improving scalability.

The introduction of new protocols in telecommunications like GMPLS gives a new trend for multilayer data networks. This new system provides a common signalling and routing framework between the different layers, and it does not restrict the way these layers work together. This evolution along with the explosive growth of traffic are yielding new survivability issues in multilayer networks, which we discuss in the next section.

2.2 Survivability concepts in multilayer networks

Network survivability implementation applies a mechanism for fault detection and localization together with a set of *recovery* techniques to reroute traffic around the failed point. Recovery techniques can be implemented based on two general approaches, *protection* and *restoration*.

2.2.1 Restoration

The restoration is a recovery strategy that allows to dynamically reroute connections when a failure occurs in the network. In a restoration approach, we must calculate, at the time of the crash, a new routing path using the available resources of the network. Here, the network must use the so-called online algorithms, since the failure is not a static problem neither a problem that is known in advance.

According to the failure nature and context, we can distinguish two types of restoration:

- Path restoration: In a path restoration, the source and destination nodes of each connection traversing the failed link participate in a distributed algorithm to dynamically discover a backup route.
- Link restoration: In a link restoration, the end nodes of the failed link participate in a distributed algorithm to dynamically discover a route around the link.

If in both cases no backup route is found, then the failed connection is blocked. For more details about restoration in WDM networks, the reader is referred to [112].

2.2.2 Protection

The principle of protection is to provide in advance all cases of failures that may occur in the network in order to set up emergency solutions that ensure the continuity of traffic. These solutions propose protection paths on which the traffic can be continuously routed when the main path is affected by a failure. The path used by default for routing is called *working path* or *primary path* . However, the path used to replace the working path in case of failure is called *protection path*. Several modes of protection can considered:

- Dedicated backup: the resources of the network are here dedicated for the backup path and are not shared with other backup paths.
- Shared backup: network's resources can be shared by many backup paths.

Both modes can be used either for end-to-end path protection or for link protection. For more details about protection in WDM networks, the reader is referred to [112]. Figure 2.8 taken from [113] summarizes the previously mentioned restoration and protection mechanisms. In particular, in this thesis, we will be interested by shared backup path protection (as it will be detailed in the presentation of the problem, Chapter 3).

Hence, in case of links (or nodes) breakdowns, it is important to avoid the interruption of the network traffic, using one of the mechanism mentioned above. This implies the importance of survivability in networks. Survivability issues become more and more important in the context of many-layer networks, mainly because of the strong interaction between the different layers. In fact, generally a simple link failure in one layer directly impacts the other layers.

2.2.3 Survivability in multilayer networks

Recently, multi-layer protection in IP-over-WDM networks has received much attention, mainly because several issues that did not exist for single layer networks arise in the context of multilayer ones. Many questions have hence appeared:

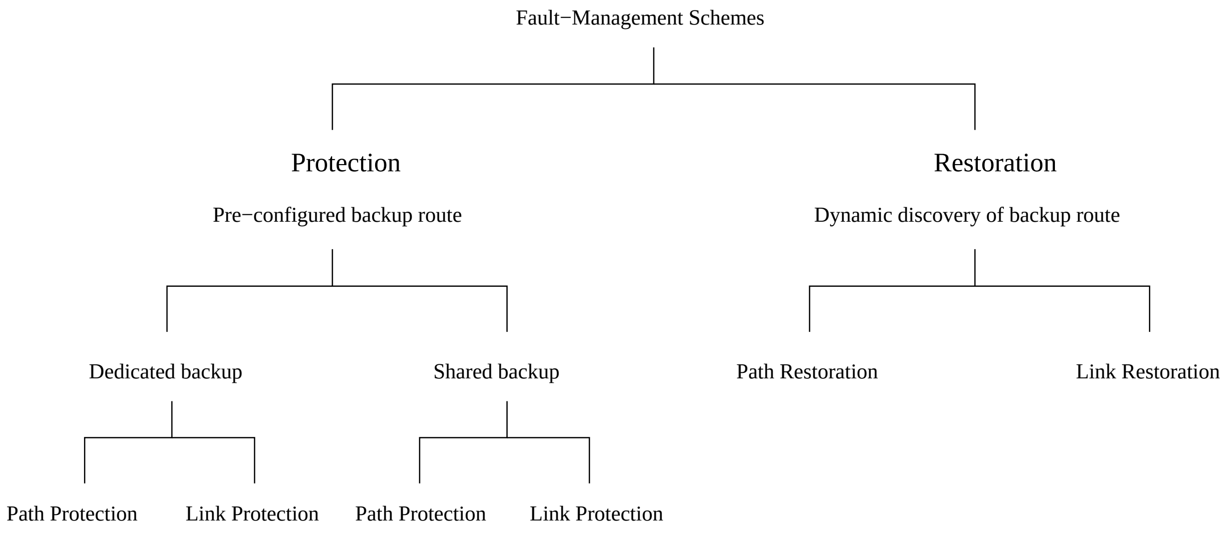


Figure 2.8: Different schemes for surviving link failures [113]

- How to identify the original failure ?
- How to handle protection responsibilities between optical and client layer ?
- How recovery strategies at the two layers can be coordinated to avoid confused actions against the same failure ?

As it will be detailed in Chapter 3, in the scope of this thesis, as a matter of coordination between the two layers, we will suppose that the IP layer is the one which first deals with failures. The impact is then determined in the optical layer.

2.3 Multilayer network design and survivability

2.3.1 The general survivable network design problem

In this section, we discuss several problems related to the conception of survivable networks, which in general originates from the telecommunications networks survivability issues. First, we present the general survivable network design problem and propose an integer linear programming formulation for it. Several variants of this model have been treated in the literature and will be presented in this section.

A network can be represented by a graph, directed or undirected, where each node of the network corresponds to a node of the graph and a link between two nodes of the network is represented by an edge or an arc of the graph. Consider a graph $G = (V, E)$ representing a telecommunication network. G is an undirected finite graph where V is the set of nodes (or telecommunication centres), and E the possible links between these nodes. With every edge $e \in E$ is associated a cost $c(e)$ corresponding to its cost of installation. Remind that the total cost of a subgraph $H = (U, F)$ of G, where $U \subseteq V$ and $F \subseteq E$, is $c(F) = \sum\limits_{e \in F} c(e)$.

With each node $v \in V$ is associated a positive integer $r(v)$ called the *connectivity type* of node v, which can be seen as the minimum number of edges connecting v to the other nodes of the network. The vector $(r(v)|v \in V)$ is the connectivity type vector associated with the nodes of G. The concept of *type of connectivity* has been introduced by Grötschel and Monma [68].

We say that a subgraph $H = (U, F)$ of G satisfy the *edge connectivity* (resp. *node connectivity*) *requirement*, if for every pair of nodes $u, v \in V$, there exists at least

$$r(u, v) = min\{r(u), r(v)\}$$

edge-disjoint (resp. node-disjoint) paths between u and v. These conditions ensure a certain survivability in the network. Remark that when at most $r(u, v) - 1$ links in case of edge-connectivity (resp. nodes in case of node-connectivity) break down in the path between u and v, the traffic between the two nodes continues always to be routed.

Consider an undirected graph $G = (V, E)$. Let $(r(v)|v \in V)$ be a connectivity type vector and $c : E \to \mathbb{R}$ a cost function. We suppose, without loss of generality, that there exist at least two nodes having a degree equal to k, where $k = max\{r(v)|v \in V\}$. Initiated by Winter [135] and after by Grötschel, Monma and Stoer [69], the *general survivable network design problem* consists in finding a minimum cost subgraph of G which satisfies the connectivity requirement. We will denote this problem by kECON (resp. kNCON) for edge connectivity (resp. node connectivity) requirement.

Remark that in the special case of $r(v) \in \{0, 1\}$ for all $v \in V$, the kECON (resp. kNCON) is nothing but the Steiner tree problem. Since the Steiner tree problem is NP-hard [56], we deduce that the kECON (resp. kNCON) is NP-hard as well. However, under certain conditions, the problem can be solved in polynomial time. In fact, when $r(v) = 1$ for all $v \in V$, the problem is equivalent to the minimum weight spanning tree problem. Therefore, the kECON (resp. kNCON) can be solved in polynomial time using Kruskal [89] or Prim [110] algorithms. In addition, when $r(s) = r(t) = 1$ for two nodes $s, t \in V$ and $r(v) = 0$ for all $v \in V \setminus \{s, t\}$, the problem coincides with the shortest st-path problem which is solvable in polynomial time using either the algorithm of Dijkstra [41] or the one of Bellman-Ford [19].

All the properties of connectivity mentioned above are in close relationship with the cuts. In [102], Menger state a fundamental relation between the number of edge-disjoint paths and the cardinality of cuts in the graph G. This relation is given in the following theorem.

Theorem 2.1 *[102] Let $G = (V, E)$ be an undirected graph and s and t two nodes of G. Then, there is at least k edge-disjoint paths between s and t if and only if every st-cut of G contains at least k edges.*

From Theorem 2.1, we can deduce an integer linear programming formulation of the kECON. To this end, let us first introduce some notations.

$$\begin{array}{rcll} r(W) & = & max\ \{r(u) — u \in W\} & \forall W \subseteq V, \\ con(W) & = & max\ \{r(u,v) — u \in W, v \in V\backslash W\} & \\ & = & min\ \{r(W), r(V\backslash W)\} & \forall W \subseteq V,\ \emptyset \neq W \neq V. \end{array}$$

Consequently, the kECON is equivalent to the following integer linear program.

$$\begin{array}{lll} min \sum_{e \in E} c(e)x(e) & & \\ x(\delta(W)) \geq con(W) & \text{for all } W \subseteq V, \emptyset \neq W \neq V, & (2.1) \\ x(e) \geq 0 & \text{for all } e \in E, & (2.2) \\ x(e) \leq 1 & \text{for all } e \in E, & (2.3) \\ x(e) \in \{0, 1\}, & & (2.4) \end{array}$$

where $x(\delta(W)) = \sum_{e \in \delta(W)} x(e)$.

Inequalities (2.1) are called the *cut inequalities*, inequalities (2.2) and (2.3) are the trivial inequalities and inequalities (2.4) are the integrality constraints. Inequalities (2.1) ensure that each solution of the problem intersect the cut $\delta(W)$ in at least $con(W)$ edges.

Consider now a node subset $Z \subseteq V$. If $W \subseteq V \setminus Z$, then we set $con_{G\setminus Z}(W) = min\{r(W), r(V \setminus (Z \cup W))\}$. It is not hard to see that inequalities

$$x(\delta_{G\setminus Z}(W)) \geq con_{G\setminus Z}(W) - |Z| \quad \text{for all } Z \subseteq V, \emptyset \neq Z \neq V \text{ and } W \subseteq V \setminus Z, \quad (2.5)$$

are valid for the polytope kNCON(G).

By adding inequalities (2.5) to the integer linear program given above, we have a formulation for the kNCON problem. Inequalities (2.5) will be also called *node-cut inequalities*.

The polytope of solutions of the kECON (resp. kNCON) is the polyhedron given by

$$k\text{ECON}(G) = conv\{x \in \mathbb{R}^E | x \text{ satisfies } (2.1) - (2.4)\}$$

$$(\text{ resp. } k\text{NCON}(G) = conv\{x \in \mathbb{R}^E | x \text{ satisfies } (2.1) - (2.5)\} \text{ }).$$

Notice that $k\text{NCON}(G) \subseteq k\text{ECON}(G)$.

In [68], Grötschel and Monma investigate the polyhedral aspect of the convex hull of the solutions to the model given by inequalities (2.1)- (2.4). They discuss the dimension of the associated polytope and study the facial aspect of the basic inequalities. In a further work, Grötschel et al [70] go deeper into the polyhedral properties of that model. They also devise cutting planes algorithms and give computational results.

One of the crucial problems related to survivable network design is the *k-edge connected spanning subgraph problem* abbreviated kECSP.

Definition 2.2 *A graph $G = (V, E)$ is said to be k-edge connected (resp. k-node connected) if between every pair of nodes in V, there exist at least k edge-disjoint paths (resp. node-disjoint paths).*

Consider an undirected graph G and assume the nodes of V have the same type of connectivity. In particular, if $r(v) = k$ for all $v \in V$, we have the so-called k-edge connected spanning subgraph problem (kECSP). The kECSP consist in finding a minimum cost k-edge connected subgraph that covers all the nodes of V. The kECSP has been widely studied in the literature. In particular, in the case of low connectivity requirement $k = 2$, the problem received in particular attention. In [96], Mahjoub give a complete characterization of the 2ECSP polytope when the graph G is series parallel. He also introduced a new class of valid inequalities called the *F-partition inequalities*. In [15], Barahona and Mahjoub prove that the F-partition inequalities together with the cut inequalities and the trivial ones are sufficient to give a complete description of the 2ECSP polytope for halin graphs. They also show that in this case, the F-partition are facet defining. Another variant of the 2ECSP problem, adding hop constraints, has been also studied. These constraints have been added to limit the length either of cycle or of paths [35, 52, 75]. Barahona and Mahjoub [15] also studied the node version of the problem, the 2-node connected spanning problem (TNCSP). The TNCSP consists in finding a minimum-cost 2-node connected subgraph that covers the nodes of V. The problem has been also studied in [98] and with a hop constraint in [39]. Moreover, as in practice it is often possible to impose stronger connectivity conditions, that is to look for types of connectivity greater than 2, many works in the literature have been interested to this problem [25, 40].

2.3.2 Multilayer survivable network design

The multilayer network design problem consists in choosing the suitable equipments and capacities to place respectively on the nodes and links of the network in order to route the requirements of the traffic matrix. In an IP-over WDM network, the design and dimensioning of the logical (IP) layer is obtained by the routing of the traffic matrix. The logical capacities of the IP layer define then a traffic matrix for the WDM layer. This new traffic matrix is, to its turn, routed in the optical layer. Decisions about equipments and capacities to install on the nodes and links are hence to be made for the optical layer. Moreover, if along with the dimensioning and routing, restoration or protection mechanisms are used to avoid failures, the multilayer network is said to be survivable.

Optimization over multilayer networks has interested many researchers and telecommunication operators. The problem of designing layered networks has been first introduced by Dahl et al in [34]. Many variants of multilayer routing, dimensioning and survivability problems have been after studied and several methods of resolution have been since devised.

On one hand, heuristic approaches have been proposed. In [63], Gouveia et al study the design of MPLS-over-WDM networks. They address the dimensioning subject to some path constraints in the WDM layer and hop constraints in the MPLS layer. They give an integer programming formulation and devise a heuristic technique based on that formulation. In [120], Ricciato et al. consider the problem of off-line configuration of MPLS-over-WDM networks under time-varying offered traffic. They present a mixed integer programming formulation for the problem and discuss heuristic approaches.

On the other hand, exact methods have been proved to be efficient for solving to optimality large-scale problems in the multilayer context. In [34], the authors studied the problem of designing virtual links (called pipes in the paper) over physical capacities. The problem is formulated as an integer linear program. The associated polytope is studied and different classes of facets are described resulting in a cutting-plane algorithm. In [32], Cruz et al have been interested in the design of multi-level networks including discrete facility location, topological network design and dimensioning. A Mixed Integer Program is proposed and solved using a branch-and-bound algorithm combined with a lagrangian relaxation. Belotti et al propose in [20], a path-based Mixed Integer Program for the design of two-layer networks taking into account the impact of statistical multiplexing in the MPLS layer. A column generation approach coordinated with a lagrangian relaxation has been introduced to solve the model. Gouveia et al [62] studied a hop-constrained node survivable network design problem in the context of MPLS-over-WDM networks. The problem is considered within two different survivability mechanisms: path diversity and path protection. An Integer Linear Programming model is proposed for both mechanisms and a cost analysis of the design solutions is handeled on the NSFNet and EON real world networks results. Fortz et Poss [53] gave a brief survey on some multi-layer network design works. They proposed a reformulation of the multi-layer network design using Benders decomposition. A branch-and-cut algorithm was devised and strengthened with metric inequalities improving hence the results obtained previously by Knippel and Lardeux in [87]. Orlowski et al [106] studied the problem of planning multilayer SDH/WDM networks. The goal was to find a minimum cost installation of link and node hardware for both layers, subject to many practical constraints as well as constraints related to survivability against failures. A mixed-integer programming formulation is proposed and solved using an efficient branch-and-cut algorithm. The algorithm is based on a problem-specific preprocessing, MIP-based heuristics, and cutting planes approach, which was useful to solve to optimality realistic two-layer instances. In [111], the authors studied the problem of designing logical and physical layers in multi-layer networks with non-bifurcated traffic flows. Unlike classical works dealing with the two layers independently or in a sequential fashion, the author proposed to design simultaneously the two layers of the graph. A branch-and-price procedure that simultaneously solves the logical topology design and the traffic routing in the optical layer is considered. In [105], Orlowski et Pióro proposed a survey of different mechanisms of survivability in telecommunication networks. The authors were particularly interested to non-compact formulations. Several path-based survivability mechanisms are considered and the complexity of the corresponding pricing problems is studied.

In this thesis, we propose to study a problem within the context of survivable multilayer networks. However, unlike the previously mentioned works, we will be interested only in the network topology design, that is to say we will not deal with dimensioning issues. Moreover, we will suppose that the protection strategy is accomplished in the IP layer and try to find the corresponding protection strategy in the lower layer. The problem that we study is detailed in the following chapter.

Chapter 3

MSOND Problem: context and complexity

Contents

As it was shown in Chapter 2, the problem of reliability in the multilayer telecommunication networks is of a crucial importance, mainly because of the strong interaction between the different layers. In this chapter, we consider a problem of survivability in IP-over-WDM networks that consists in determining a secure topology for the optical layer given a survivable one in the logical layer. First, we state the problem and model *it in terms of graph. We also give some notations and examples. In addition to its importance in telecommunications, our problem is very interesting from a theoretical point of view. We show indeed that it is in a close relationship with many interesting classical problems such that the Steiner Travelling Salesman Problem. Finally, we study the complexity of the problem and prove that it is NP-hard even in the simple cases.*

3.1 The MSOND problem

3.1.1 Problem presentation

Consider an IP-over-WDM network consisting of an IP layer over a WDM layer. The IP layer is called physical or also virtual layer. The WDM layer is called optical or physical layer as well. The IP layer is composed of IP routers which are interconnected by virtual links and the WDM layer consists of a number of Optical Cross Connects OXC (optical switches) interconnected by physical links. Since the two layers are communicating with an interface called UNI interface, to each router in the IP layer corresponds an optical switch in the WDM layer.

Figure 3.1 shows a bilayer network. The logical layer contains four routers denoted by R_1, R_2, R_3 and R_4. The optical layer holds seven optical switch denoted $S_1, S_2, \ldots, S_7$. The Optical switches S_1, S_2, S_3, S_4 correspond respectively to routers R_1, R_2, R_3, R_4. The edges (R_1, R_2), (R_2, R_3) and (R_1, R_3) represent virtual links in the logical layer. These links are in reality ensured by physical paths in the optical layer. In fact, (R_1, R_2) is physically routed by (S_1, S_4, S_5, S_2), (R_2, R_3) by (S_2, S_7, S_3) and (R_1, R_3) by (S_1, S_7, S_3).

Due to the multilayer aspect of the network, each failure in one layer may bring about a breakdown in the other layer. Remark that in the example of Figure 3.1, if node S_7 of the optical layer breaks down, the physical paths from S_1 to S_3 and from S_2 to S_3 are interrupted. Consequently, the virtual links (R_1, R_3) and (R_2, R_3) in the logical layer will break down.

In the sequel, we present a problem related to multilayer survivability in IP-over-WDM networks. Given a secure topology for the logical layer (protection by shared back-up paths, c.f. Chapter 2), we look for a survivable corresponding topology in the optical layer. Consequently, logical layer will constitute the data layer and optical layer is the layer to optimize.

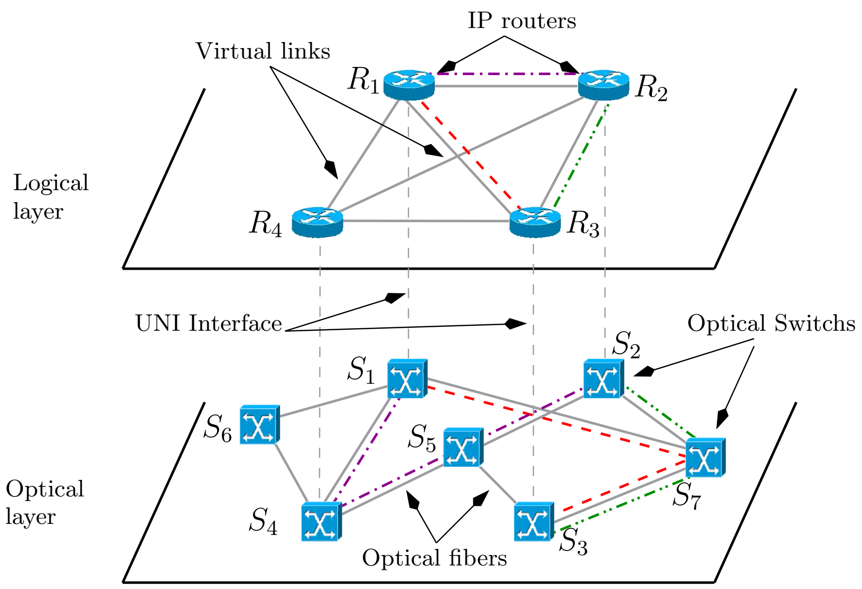


Figure 3.1: Example of multilayer network

Consider a bilayer IP-over-WDM network and a set of demands between routers such that for each demand two node-disjoint paths routing it in the virtual layer are given. We suppose that every installation in the optical layer is possible and with each physical link we associate a cost corresponding to its installation's cost.

The *Multilayer Survivable Optical Network Design (MSOND)* problem consists in finding, for each demand, two *node-disjoint* physical paths routing it in the optical layer. These paths must go in *order* through the optical switches corresponding to the routers visited in the logical paths and the total cost of installation must be minimum.

In the case of just one demand, the problem will be called *Single Commodity MSOND* problem or for short *SC-MSOND* problem. And in the case of more than one demand, the problem is called *Multi-Commodity MSOND* problem and is abbreviated MSOND problem.

3.1.2 Notations and examples

The problem can be modeled in terms of graph. We associate with the optical layer an undirected graph $G = (V, E)$ where the nodes of V correspond to the optical switches and the edges of E to the possible physical links between these optical switches. Similarly, we associate with the logical layer an undirected graph $G' = (V', E')$ where the nodes of V' correspond to routers and the edges of E' to possible links between these routers. To every router $v_i \in V'$ corresponds an optical switch $w_i \in V$. We assume that there exist traffic demands between nodes of G'. Let us denote by K the set of these demands. Denote by (O'_k, D'_k) the pair of routers origin-destination for demand $k \in K$ and by O_k et D_k the corresponding optical switches in the optical layer.

Let $L'_{k,1} = (v_1^{k,1}, ..., v_j^{k,1}, ..., v_{l_{k,1}}^{k,1})$ and $L'_{k,2} = (v_1^{k,2}, ..., v_j^{k,2}, ..., v_{l_{k,2}}^{k,2})$ be the two paths routing demand $k \in K$ in graph G' and denote by $L_{k,1}$ and $L_{k,2}$ the corresponding physical paths that we are looking for in G. $L'_{k,1}$ and $L'_{k,2}$ go through routers $v_j^{k,i}, k \in K, j = 1, ..., l_{k,i}, i \in \{1, 2\}$, k designates the demand, i the path and j the rank of the node in the corresponding path. To each of this router corresponds an optical switch $w_j^{k,i}$, $k \in K, j = 1, ..., l_{k,i}$ and $i \in \{1, 2\}$. These particular optical switches must be obligatory visited and they are hence called *terminal nodes* or *terminals* of the demand. We denote by T_k the set of terminals of demand $k \in K$. The remaining nodes, $V \backslash T_k$, are called *Steiner nodes* or *steiners* for demand $k \in K$ and are denoted S_k. Notice that terminals can be different from one demand to another and that a terminal node for one demand could be a Steiner node for another. For the SC-MSOND problem, we simply denote by T the set of terminals of the demand and by S the set of its Steiner nodes. The paths routing the different demands can be also seen as a sequence of sections. Each section is defined by two successive terminals of the demand. Let us denote by $\mathcal{T}_{k,1} = \{q_j^{k,1} = (w_j^{k,1}, w_{j+1}^{k,1}), j = 1, ..., l_{k,1} - 1\}$ the sections of the first path, and by $\mathcal{T}_{k,2} = \{q_j^{k,2} = (w_j^{k,2}, w_{j+1}^{k,2}), j = 1, ..., l_{k,2} - 1\}$ the sections of the second path, for $k \in K$. Let $\mathcal{T}_k = \mathcal{T}_{k,1} \cup \mathcal{T}_{k,2}$, $k \in K$. $\mathcal{T}_k$ represents the set of all sections of demand $k \in K$ and can also be defined as follows $\mathcal{T}_k = \{q_j^k, j = 1, ..., |T_k|, k \in K\}$, where $q_j^k = (w_j^k, w_{j+1}^k)$, $j = 1, ..., |T_k|, k \in K\}$. For convenience, a section q_j^k, $j = 1, ..., |T_k|, k \in K\}$ may also be denoted q for short. For each section $q \in \mathcal{T}_k$, we denote by $\mathcal{L}_q^k \subset (L_{k,1} \cup L_{k,2})$ the sub-path routing the section q in G. Consider a section $q = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k$. We denote by $G^{k,j}$ the reduced graph obtained from G by deleting all terminals T_k of the demand k excepted the terminals extremities of section q, namely w_j^k and w_{j+1}^k. Moreover, as a matter of simplification, in Chapter 6, we opt for $G^{k,q}$ to designate the same reduced graph. Finally, Graph G is assumed to have infinite capacities on its edges. The installation of a physical edge e in the optical layer has a cost denoted by $c(e) > 0$.

In order to understand better the problem, in what follows we propose some examples of illustration.

3.1.2.1 Single commodity MSOND problem

When $k = 1$, we deal with the SC-MSOND problem. Figure 3.2 depicts an example of a solution for the SC-MSOND problem. It reproduces the graphs G' and G corresponding to the multilayer network of Figure 3.1. G' corresponds to the logical layer and its nodes are $V' = \{v_1, v_2, v_3, v_4\}$. G represents the optical layer and its nodes are $V = \{w_i, i = 1, ..., 7\}$. To each node $v_i \in V', i = 1, ..., 4$, corresponds a node $w_i \in V, i = 1, ..., 4$. Nodes w_5, w_6 and w_7 of G do not have corresponding nodes in G'.

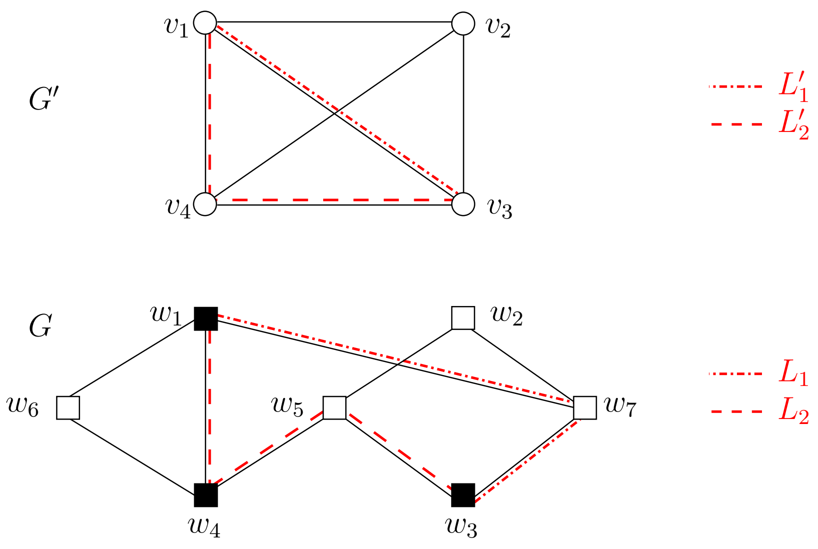


Figure 3.2: Feasible solution for the SC-MSOND problem

The graph G' represents data for the problem. G' shows a demand between nodes v_1 and v_3. This demand is routed by the two node-disjoint paths $L'_1 = (v_1, v_3)$ and $L'_2 = (v_1, v_4, v_3)$. In G, a feasible solution for the problem is given. The first path is routed by $L_1 = (w_1, w_7, w_3)$. The second path is routed by $L_2 = (w_1, w_4, w_5, w_3)$. Note that, unlike the other nodes, nodes w_1, w_4 and w_3 are represented by filled squares. These nodes corresponds to v_1, v_4 and v_3 that represents the nodes visited in L'_1 and L'_2 in G'. Nodes w_1, w_4 and w_3, called also terminals, must be visited in the same order of visiting the corresponding nodes in G'. All the other nodes in $V \setminus \{w_1, w_3, w_4\}$ are nodes that may be used to route the demand. They are called Steiner nodes or steiners. First, note that paths L_1 and L_2 of G are node-disjoint. In fact, apart from the origin w_1 and the destination w_3, L_1 and L_2 do not have any other node in common. Remark also that L_1 and L_2 are respecting the order of passing through the nodes in L'_1 and L'_2. In fact, L_2 is visiting node w_4 corresponding to v_4 before arriving to w_3 corresponding to v_3. Also note that, since the graph is undirected, looking for the two paths L_1 and L_2 for the demand is no more than seeking an elementary cycle going in a specific order through the terminals w_1, w_4 and w_3.

Consider now a second example for the SC-MSOND problem (see Figure 3.3). Consider the same demand between nodes v_1 and v_3. The demand is now routed by $L'_1 = (v_1, v_5, v_2, v_3)$ and $L'_2 = (v_1, v_4, v_3)$. Figure 3.3 shows two solutions suggested for the given paths. The first solution is represented by (a) and the second by (b). Note that solution (a) is infeasible because the order constraint is violated. In fact, in the path $L_1 = (w_1, w_2, w_5, w_3)$, the terminal node w_2 is visited before the terminal node w_5 and this does not correspond to the order in the path $L'_1 = (v_1, v_5, v_2, v_3)$, where v_5 is visited before v_2. The solution given by (b) is not feasible as well. In fact, this solution gives two node-disjoint paths respecting the order constraint. However, the first path $L_1 = (w_1, w_5, w_8, w_7, w_2, w_8, w_3)$ defines rather a walk as it goes twice through the Steiner node w_8.

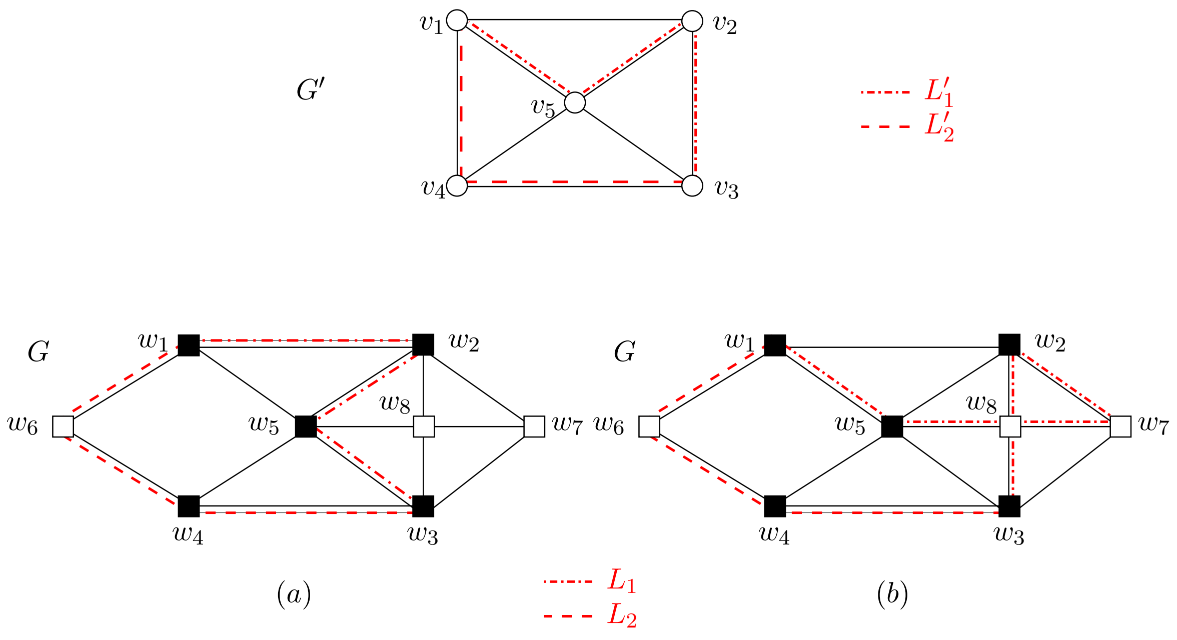


Figure 3.3: Infeasible solutions for the SC-MSOND problem

3.1.2.2 Multi-commodity MSOND

Suppose now that $k > 1$, which means that we are given at least two demands to route. Figure 3.4 shows an instance composed of two demands. The first demand is between v_1 and v_3 routed on paths (v_1, v_4, v_3) and (v_1, v_6, v_3). The second demand is between v_5 and v_4 routed on paths (v_5, v_4) and (v_5, v_1, v_2, v_4). Note that the two demands have different sets of terminals and steiners. In fact for the first demand, the terminals' set is $T_1 = \{w_1, w_3, w_4, w_6\}$ and the steiners' set is $S_1 = \{w_2, w_5, w_7\}$. However, for the second demand the terminals' set is $T_2 = \{w_1, w_2, w_4, w_5\}$ and the steiners' set is $S_2 = \{w_3, w_6, w_7\}$. Here, node w_7 does not have a corresponding node in the graph G and it is thus a Steiner node for both demands. Figure 3.4 illustrates two feasible solutions shown in (a) and (b). Solution (a) shows that the first demand is routed on (w_1, w_4, w_3) and (w_1, w_6, w_3) and that the second demand is routed on (w_5, w_4) and (w_5, w_1, w_2, w_4). However, in solution (b), the first demand is routed on (w_1, w_5, w_4, w_3) and (w_1, w_2, w_6, w_3) and the second demand is routed on (w_5, w_4) and $(w_5, w_1, w_2, w_6, w_3, w_4)$. Note that both solutions are feasible for the MSOND problem. Assuming that the costs correspond to euclidean distances, solution (b) seems to be more interesting since it is less expensive than solution (a). In fact, in solution (b) we make profit from the same edges to route the two demands.

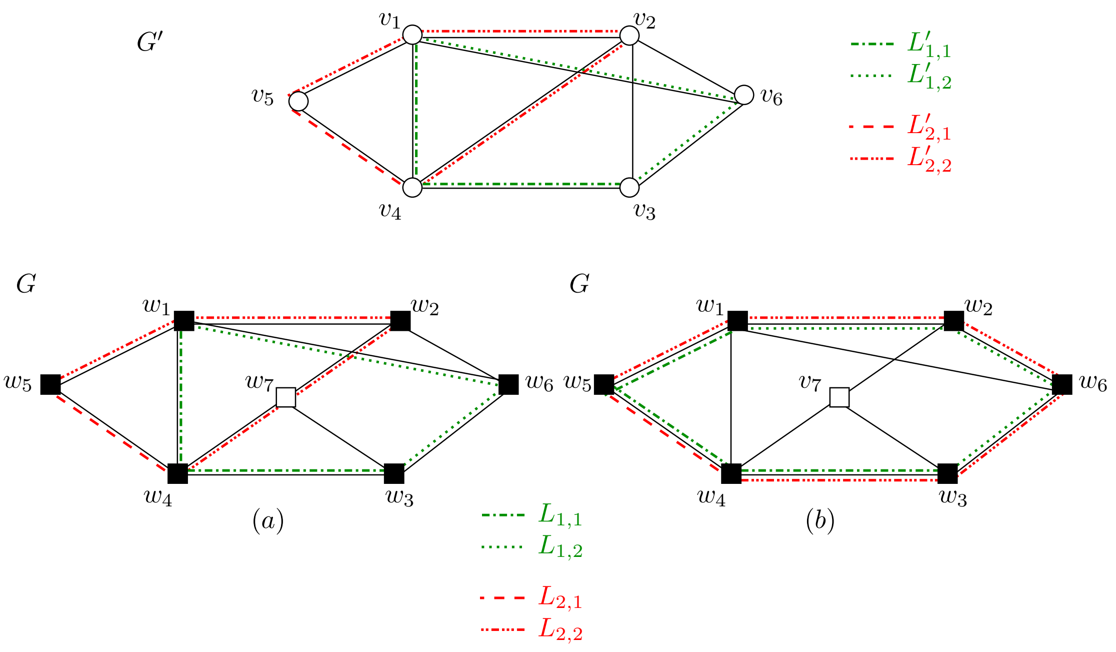


Figure 3.4: Feasible solutions for the MSOND problem

3.1.3 Sections' disjunction

First, we give the definition of disjunction between sections.

Definition 3.1 *Consider a demand $k \in K$ and two sections of the demand $q_i, q_j \in \mathcal{T}_k$. q_i and q_j are said to be node-disjoint if the sub-paths $\mathcal{L}^k_{q_i}$ and $\mathcal{L}^k_{q_j}$ routing respectively q_i and q_j are node-disjoint.*

We have the following result.

Proposition 3.2 *Consider a demand $k \in K$. Paths $L_{k,1}$ and $L_{k,2}$ are node-disjoint if and only if the sections of demand k are pairwise node-disjoint.*

Proof Consider a demand $k \in K$ and two node-disjoint paths $L_{k,1}$ and $L_{k,2}$ routing it in G. Suppose that there exist two sections $q_i \in \mathcal{T}_k$ and $q_j \in \mathcal{T}_k$ that are not node-disjoint. This means that the sub-paths $\mathcal{L}^k_{q_i}$ and $\mathcal{L}^k_{q_j}$ routing q_i and q_j have at least one node in common. Let us denote by w this node. Here, two cases have to been distinguished.

1) q_i and q_j are sections of the first path $L'_{k,1}$ (respectively second path $L'_{k,2}$). w is a common node between $\mathcal{L}^k_{q_i}$ and $\mathcal{L}^k_{q_j}$ means that the path $L_{k,1}$ (respectively $L_{k,2}$) passes at least twice by w which implies that $L_{k,1}$ (respectively $L_{k,2}$) is a walk (not elementary), contradiction.

2) q_i and q_j do not belong to the same path. Suppose that $q_i \in L'_{k,1}$ and $q_j \in L'_{k,2}$. q_i and q_j are not node-disjoint means that $\mathcal{L}^k_{q_i}$ and $\mathcal{L}_{q_j,k}$ have a common node w. Consequently w is a common node between $L_{k,1}$ and $L_{k,2}$ and hence the two paths are not node-disjoint, contradiction.

Conversely, suppose that all the sections of demand k are pairwise node-disjoint, and that the two paths $L_{k,1}$ and $L_{k,2}$ are not node-disjoint. This implies that $L_{k,1}$ and $L_{k,2}$ share at least one node, say w. Thus there exist at least two sections q_i and q_j sharing w and so they are not node-disjoint, contradiction. □

The MSOND problem is of a crucial importance in telecommunication since it provides a secure topology for the optical layer in IP-over-optical networks. The problem is also quite interesting from a theoretical point of view. In fact, it is in a close relationship with many important known problems as it will be detailed in the next session.

3.2 Theoretical context

Apart from its importance in the telecommunication context, the MSOND problem is very interesting from a theoretical point of view and raised from challenging classical problems. In fact, for a single demand, the SC-MSOND problem can be seen as an elementary cycle that must go in order through some terminal nodes. The cycle can also visits other nodes called Steiner nodes. This is in a close relationship with some classical problems and in particular some variants of the Travelling Salesman Problem (TSP) [8]. In particular, the SC-MSOND problem can be seen as a Steiner TSP with an order constraint on its terminals.

In the following, we describe some classical problems known in the literature that are closely related to the MSOND problem.

3.2.1 Shortest Path Problem with Specified Nodes

The Shortest Path Problem with Specified Nodes (*SPPSN*) consists in finding a shortest path between an origin and a destination passing necessarily through a set of *specified nodes* called also *terminals*. Consider a weighted graph with N nodes and distances associated with its edges. The SPPSN is to find the shortest path between nodes 1 and N that goes through k terminals (different from nodes 1 and N). This problem was first introduced by Kalaba when describing some communication network problems [82]. A polynomial algorithm was given by Saksena and Kumar in [118] but was later proved to be erroneous by Dreyfus [42]. Different variants of the problem was then studied in [13],[77], [90] and [133].

The SPPSN is close to the SC-MSOND problem since the later can be seen as a shortest path problem between the origin of the demand and itself and going through some specific nodes that are the terminals of the demand. However, another constraint must be taken into account for the SC-MSOND problem, which is the order between the different terminals.

3.2.2 Travelling Salesman Problem and its variants

Given a set of cities along with the cost of travel between each pair of them, the Travelling Salesman Problem, or TSP for short, is to find the cheapest way of visiting all the cities and returning to the starting point [8]. More formally, this can be modelled as weighted graph $G = (V, E)$ in which we look for a minimum-cost tour. The TSP is a very well-known problem in combinatorial optimization and operations research in general. Since its appearance in the 1930's, it has interested many researchers and has been one of the most intensively studied problems in optimization. Moreover, as it has several applications in various fields, many variants of the problem have since appeared. In the following, we describe two variants of the TSP that are closely related to the SC-MSOND problem.

3.2.2.1 Steiner TSP

Consider a graph $G = (V, E)$ with a cost $c_e > 0$ associated with each $e \in E$, and a set of *terminal nodes* W. The Steiner TSP consists in finding a minimum-cost tour going through the terminals W. The problem was first introduced by Cornuéjols et al. [31] in 1985. The authors study the problem in the graphical case and investigate its polyhedron in series-parallel graphs. Later, in 2002, Mahjoub and Baïou give a complete polyhedral description of the STSP in series-parallel graphs [12]. In 2003, Salazar-González proposes a formulation in terms of Integer Linear Programs for the problem [119]. He gives some polyhedral results and shows a lifting method to used to deduce valid inequalities from the TSP's inequalities. In 2009, Steinová proposes some approximation results for the problem [125]. In 2012, Letchford et al. introduce in [91] compact formulations deduced from the ones known for the TSP.

The Steiner TSP is very near to the SC-MSOND problem. The only difference is that in the SC-MSOND problem, we have one more constraint which is the order between the diffrent terminals. Which means that the SC-MSOND problem is no more than a Steiner TSP with an order constraint between the different terminals.

3.2.2.2 TSP with precedence constraints

Given a graph $G = (V, E)$ and a set of nodes $W \subset V$, the Travelling Salesman Problem with Precedence Constraints can be defined as a TSP subject to some precedence constraints between the nodes W. This problem was widely studied in the litterature. In [14] and [9], the authors study the polytope of the Asymmetric TSP with precedence constraints. In [9], Ascheuer et al. propose a Branch-and-Cut algorithm for the problem. In [64], Gouveia and Pesneau give extended formulations to the Asymmetric TSP with precedence constraints and described also a Branch-and-Cut algorithm for the problem.

Some of the works viewed the precedence constraint as a pick-up/delivery constraint. In fact, when a precedence constraint exists between two nodes v_1 and v_2, the problem can be treated as if v_1 was a pick-up node and v_2 a delivery node. In this context, in [116] and [43], the authors propose different formulations to the Pick-up/Delivery TSP. Given a set of pick-up nodes and a set of delivery nodes, the pick-up/delivery TSP consists in calculating a Hamiltonian tour such that each pick-up node precedes the corresponding delivery node. Several classes of valid inequalities are proposed and Branch-and-Cut algorithms are developed to solve the problem for both works.

The TSP with precedence constraint has close links with the SC-MSOND problem. However, there are some particularities for the SC-MSOND problem. First, for the SC-MSOND problem, not all the nodes must be visited, only terminals are mandatory. Also, unlike the TSP with precedence constraint, the precedence constraint for the SC-MSOND problem is defined between all the terminals which leads to a total order between them.

3.2.3 The k-Vertex Disjoint Paths Problem

Given a graph $G = (V, E)$ and a collection of vertex pairs $\{(s_1, t_1), ..., (s_k, t_k)\}$, the *$k$ Vertex Disjoint Paths Problem (k-VDPP)* is to find k pairwise node-disjoint paths $P_1, ..., P_k$ where P_i is a path from s_i to t_i for each $i = 1, ..., k$.

In the corresponding weighted problem, the paths P_i, $i = 1, ..., k$ have to be chosen such that a given objective function is minimized. Here, two possible objectives can be considered: minimizing the total path length (min-sum) and minimizing the length of the longest path (min-max). Another optimization version is to maximize the number of paths calculated in the graph such that they are vertex-disjoint.

The k vertex disjoint paths problem has been widely studied mainly in terms of complexity. In Table 3.1, some important complexity results for different variants of the problem are given.

The relationship between the k-VDPP and the SC-MSOND will be treated in details in the next sections.

Theorem 3.3 *The k-VDPP is :*

(i) NP-hard when k is input [84].

"k" Vertex Disjoint Paths Problem			
Unweighted Version		**Weighted Versions**	
Conditions	**Complexity**	**Conditions**	**Complexity**
undirected, fixed k	P [114, 85]	directed/undirected, k input, min-sum	NP-hard [46]
undirected, k input	NP-hard [84]	undirected, $k = 2$, min-sum	pol.alg [46]
directed, $k \geq 2$	NP-hard [51]	directed, $k = 2$, min-sum	NP-hard [88]
graph special cases	pol.alg [72]-[127]	directed/undirected, fixed $k \geq 3$, min-sum	NP-hard [136]
		directed planar graph, min-sum	pol.alg, $k = 2, 3$ [37, 88]
		directed, $k = 2$, min-max	NP-hard [92]

Table 3.1: Complexity of variants of the k-VDPP

(ii) Polynomial when k is fixed (algorithms in $\mathcal{O}(n^3)$ [114] and $\mathcal{O}(n^2)$ [85]).

In the next section, we will show that the SC-MSOND problem is NP-hard, except for some cases, using a reduction from the k-VDPP.

3.3 Complexity results

In this section, we propose to study the complexity of the MSOND problem. To achieve this, we will distinguish the case of single commodity SC-MSOND and the case of multi-commodity MSOND. For both SC-MSOND and MSOND, we distinguish different cases depending on the number of Steiner nodes $|S|$ (resp. terminal nodes $|T|$), whether it is fixed or a part of the input.

3.3.1 Single commodity MSOND problem

Consider the case of the SC-MSOND problem. We first study the case of a fixed number of Steiner nodes.

3.3.1.1 Case of fixed number of Steiner nodes ($|S|$ fixed)

Let us denote by $T = \{t_1, t_2, ..., t_q\}$ the set of terminal nodes and $S = \{s_1, s_2, ..., s_p\}$ the set of Steiner nodes for the demand.

When the number of Steiner nodes is fixed, it is possible to enumerate in a polynomial time the solutions of SC-MSOND. Algorithm 3 illustrates this enumeration. Let us denote by SOL the possible solutions for SC-MSOND and by $S_0(T)$ the solution given by all the edges between successive terminals of T.

Algorithm 3: SC-MSOND: solutions' enumeration

```
1 $SOL \leftarrow S_0(T)$;
2 forall $s \in S$ do
3     forall $Sol \in SOL$ do
4         forall $(t_j, t_{j+1})$ successive terminals of $Sol$ do
5             $T' = \{t_1, ..., t_i, s, t_j, ..., t_q\}$;
              /* $s$ will be after considered as a terminal */
6             $SOL' \leftarrow SOL' \cup S_0(T')$;
7     $SOL \leftarrow SOL \cup SOL'$;
8     $SOL' \leftarrow \emptyset$;
9 return $SOL$;
```

The idea of Algorithm 3 is the following. We start from the trivial initial solution $S_0(T)$. Then, we consider a Steiner node denoted in the algorithm by s. In the first iteration, this coincides with s_1, the first Steiner node to be inserted in the solution. Since inserting Steiner nodes between two successive terminals instead of the edge between these terminals gives always feasible solutions, the Steiner node s_1 will be inserted between all the sections of the demand. Recall that a section is defined by two successive terminals of a demand. This means that the Steiner node s_1 will be considered like a "terminal" with different possible positions between the initial terminals T of the demand. Indeed, since the terminals are considered in a predefined order, the Steiner "terminal" node s_1 can be placed between the initial terminals t_1 and t_2 or t_2 and t_3, etc... yielding to $|T|$ feasible solutions. At the second iteration, the same reasoning will be applied considering all the previous solutions and inserting this time the Steiner node s_2. And so on, until considering all the possible solutions inserting the different Steiner nodes.

From a complexity point of view, the algorithm can be inspected as follows. In the first level, we have only one solution using 0 Steiner nodes. In the second level, we consider the Steiner node s_1 and depending on its position, we get $|T|$ more solutions. We then get in total $|T|+1$ feasible solutions. In the third level, we will apply the same process inserting s_2 for the $|T|+1$ previous solutions, yielding hence to $(|T|+1)*(|T|+2)$ feasible solutions. Repeating the same procedure for all the Steiner nodes, at the last level, we will have $1{*}(|T|{+}1){*}(|T|{+}2){*}(|T|{+}3){*}....{*}(|T|{+}|S|)$ possible solutions which is dominated by $1{*}(|T|{+}|S|){*}(|T|{+}|S|){*}(|T|{+}|S|){*}....{*}(|T|{+}|S|) = (|T|{+}|S|)^{|S|} = n^{|S|}$ and this is polynomial since we suppose that $|S|$ is fixed.

Consequently, we conclude that Algorithm 3 is polynomial and it has a complexity $\mathcal{O}(n^{|S|})$.

Remark 3.4 *When the number of terminals $|T|$ is fixed, $n = |T| + |S|$ is also fixed and hence $\mathcal{O}(n^{|S|})$ is constant.*

Based on the developments given above, we have the following result.

Theorem 3.5 *When the number of Steiner nodes is fixed, it is possible to enumerate in $\mathcal{O}(n^{|S|})$ all the solutions of the SC-MSOND problem.*

In the next section, we prove that if $|S|$ is input the SC-MSOND problem become NP-hard.

3.3.1.2 Case of non-fixed number of Steiner nodes ($|S|$ input)

In this section, we study the complexity of the SC-MSOND problem when the number of Steiner $|S|$ is a part of the input.

The SC-MSOND problem can be defined as follows:
Input: an undirected graph $G' = (V', E')$, a cost $w'_e \geq 0$ associated to each $e' \in E'$ and $T' = \{v_1, ..., v_l\}$ terminals.
Output: A simple cycle going in order through the terminals T' and such that the total cost is minimum.

The corresponding decision problem is to find if there exists an elementary cycle going in order through the terminals T' and that the total cost is at most equal to a positive integer U. Recall that T' constitutes the terminals corresponding actually to the source, the destination and intermediary nodes used in the two paths between the source and the destination. Since the two given paths are vertex disjoint, we have always $|T'| \geq 3$.

Now, consider the problem called *Weighted Min-Sum Vertex Disjoint Paths Problem* (WMSVDPP). This problem can be defined as follows:

Input: an undirected graph $G = (V, E)$, a cost $w_e \geq 0$ associated to each $e \in E$, a parameter $k \geq 3$ and $T = \{(s_i, t_i) \in V, i = 1, ..., k\}$ pairs of origin-destination.
Output: k vertex disjoint paths $P_1, ..., P_k$, where P_i is a path from s_i to t_i, $i = 1, ..., k$ such that the total cost is minimum.

Theorem 3.6 *SC-MSOND problem is equivalent to the minimum sum WMSVDPP.*

Proof We prove in the following that the decision problem associated with the SC-MSOND problem is equivalent to the decision problem associated with the WMSVDPP.

The decision problem associated with WMSVDPP is to find if there exists k vertex disjoint paths $P_1, ..., P_k$, where P_i is a path from s_i to t_i, $i = 1, ..., k$ such that the total cost is at most equal to a positive integer U.

We first propose a polynomial reduction from the WMSVDPP to the SC-MSOND problem.

Consider an instance I of the WMSVDPP formed by an undirected graph $G = (V, E)$, a cost $w_e \geq 0$ for each $e \in E$ and $T = \{(s_i, t_i) \in V^2, i = 1, \ldots, k\}$. We construct an instance I' of the SC-MSOND problem formed by an undirected graph $G' = (V', E')$, $w'_e \geq 0$ for each $e \in E'$ and T' as follows (see Figure 3.5). We add to a copy of the graph G, k vertices $u_1, ..., u_k$ and $2k$ edges $t_i u_i, u_i$ and s_{i+1}, $i = 1, ..., k$ with $s_{k+1} = s_1$. Denote E_u the added edges. Let $w'_e = w_e$ if $e \in E$ and 0 otherwise. Finally we set $T' = \{s_1, t_1, u_1, s_2, ..., s_j, t_j, u_j..., s_k, t_k, u_k, s_1\}$ the terminals.

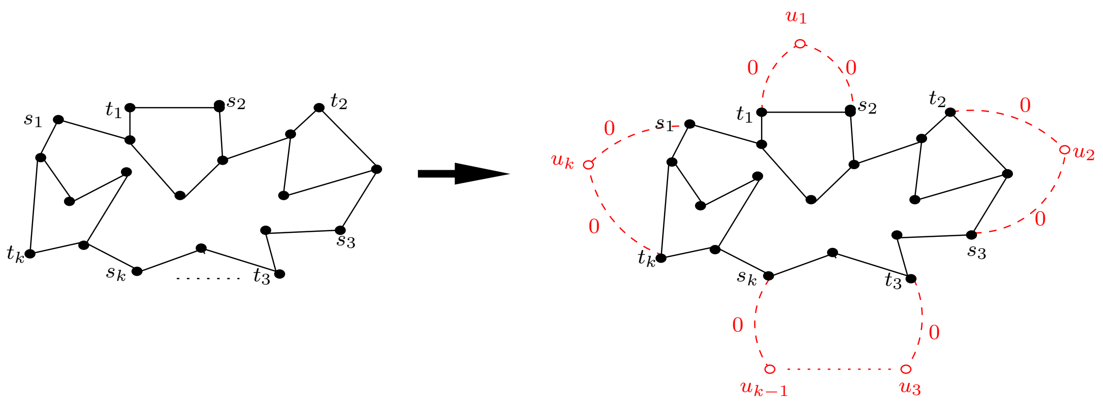


Figure 3.5: Reduction from the WMSVDP to the SC-MSOND problem

In the following we show that there exist k vertex disjoint paths between the pairs of T in G such that the total cost is at most equal to U if and only if there exists in

G' an elementary cycle going in order through the terminals T' and such that the total cost is at most equal to U.
Consider first a solution of the WMSVDPP in G with a total cost $C \leq U$. The solution consists of k vertex disjoint paths between the pairs $(s_i, t_i), i = 1, ..., k$. These paths plus the set of edges E_u constitute by construction an elementary cycle in G' going in order through the terminals of T'. And since the weights of all edges in E_u is equal to 0, the cost of the cycle is equal to C which is at most equal to U.

Consider now an elementary cycle in G' going in order through the terminals T' with a total cost $C' \leq U$. Consider the sections between the terminals $(s_i, t_i), i = 1, ..., k$. Since the cycle is elementary, these sections are vertex disjoint. Moreover, as the weights of all edges in E_u are 0, the total weight of the sub-paths between $(s_i, t_i), i = 1, ..., k$ is exactly equal to C' which at most equal to U.

Now, we present a polynomial reduction from the SC-MSOND problem to the WMSVDPP.

In a similar way, consider an instance I' of the SC-MSOND problem formed by a graph $G' = (V', E')$, a cost $w'_e \geq$ for each $e \in E'$ and $T' = \{t_1, t_2, \ldots, t_k\}$ We construct an instance of WMSVDPP formed by a graph $G = (V, E)$, a cost $w_e \geq$ for each $e \in E$ and $T = \{(s_i, t_i) \in V, i = 1, ..., k\}$ as follows (see Figure 3.6). We duplicate all the terminals $T' = \{t_1, t_2, ..., t_k\}$ by creating $S = \{s_2, s_3, ..., s_k, s_1\}$ such that with terminal t_j we associate a copy s_{j+1} for $j = 1, ..., k-1$ and with t_k is associated s_1. We also make a copy of all the incident edges to terminals $t_j, j = 1, ..., k$ as follows. For each $v \in \delta(t_j)$ we create an edge $(s_{j+1}, v) \in E'$, $j = 1, ..., k-1$ and with each $v \in \delta(t_k)$ we associate an edge $(s_1, v) \in E'$. We also add edges between the terminals of T' and their corresponding vertices in S. Denote E_s the added edges. Let $w_e = w'_e$ if $e \in E'$. If $e = (t_j, s_{j+1}), j = 1, ..., k-1$ or $e = (t_k, s_1)$, that is e is an edge between a terminal of T' and its corresponding in S, then $w_e = 0$. Else, for each $e = (s_{j+1}, v)$ with $v \in V'$ and $j = 1, ..., k-1$, $w_e = w'_{t_j v}$ and $w_{s_1 v} = w'_{t_k v}$ for each $v \in \delta(t_k)$. We set $T = T' \cup S$.

In the following we show that there exists in G' an elementary cycle going in order through the terminals T' such that the total cost is at most equal to U if and only if there exist k vertex disjoint paths between the pairs $(s_i, t_i) \in T, i = 1, ..., k$ such that the total cost is at most equal to U.

Consider first a solution of the SC-MSOND problem in G' with a total cost $C' \leq U$. The solution consists of an elementary cycle in G' going in order through the terminals of T'. By construction, we can deduce an equivalent cycle going in order through the terminals $T = \{t_1, s_2, t_2, s_3, ..., t_{k-1}, s_k, t_k, s_1\}$. This guarantees paths between the different pairs (s_i, t_i), $i = 1, ..., k$. These paths are disjoint since the cycle joining the nodes in T is elementary. Moreover, the total weight of these paths is exactly equal to

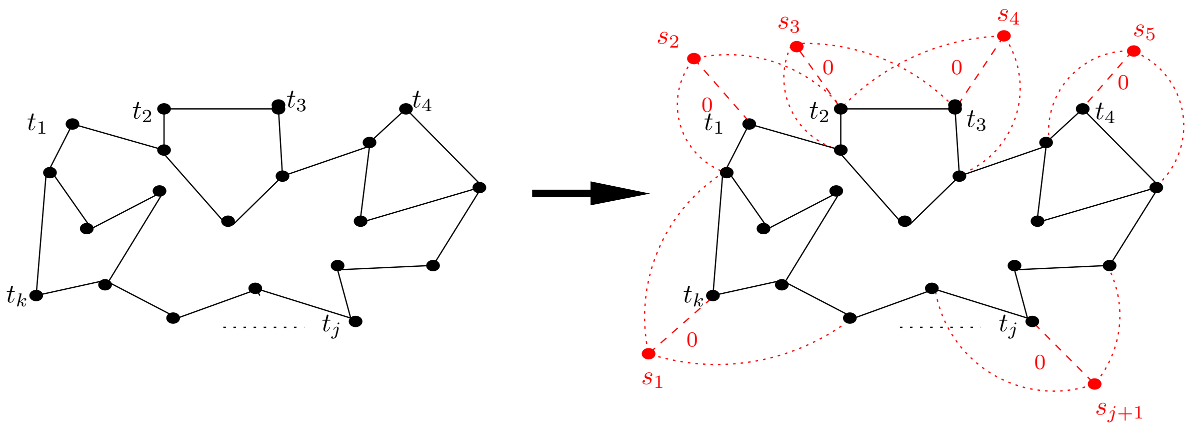


Figure 3.6: Reduction from the MSOND problem to the WMSVDP

the cost of the elementary cycle C' which is at most equal to U. □

Using the reduction developed above, we can state the following result.

Theorem 3.7 *When $|S|$ is input, the SC-MSOND is :*

(i) NP hard if $|T|$ is part of the input.

(ii) polynomial if $|T|$ is fixed.

Proof Deduced from Theorem 3.3 and Theorem 3.6. □

In the next section, we propose to study the case of many demands.

3.3.2 Multi-commodity MSOND problem

Consider now the case of multi-commodity MSOND problem ($k > 1$). Two cases are to be discerned, the case when the number of demands $|K|$ is fixed and the case when $|K|$ is input.

3.3.2.1 Case of fixed $|K|$

In this case, we have the following results

Theorem 3.8 *When the number of demands $|K|$ is fixed, the MSOND problem is NP-hard if and only if for each demand $k \in K$, the number of Steiner nodes $|S_k|$ is input.*

Proof First, suppose that $|S_k|$ is fixed for each $k \in K$. From Theorem 3.5, we know that it is possible to enumerate all the solutions for a single demand $k \in K$ in $\mathcal{O}(n^{|S_k|})$. Having $|K|$ fixed, this implies that we can enumerate all the possible solutions of the MSOND problem in $\mathcal{O}(n^{|S| \times |K|})$ (where $S = \max\limits_{k \in K} |S_k|$). But this means that in this case, the MSOND problem can be solved in polynomial time. Thus if the MSOND problem is NP-hard then $|S_k|$, $k \in K$ is input.

Conversely, suppose that $|S_k|$ is input for each demand $k \in K$. In what follows, we will prove that the MSOND problem is in this case NP-hard. To this end, we distinguish two sub-cases.

- First, suppose that $|T_k|$ is input. As it has been shown in Theorem **??**, we know that the SC-MSOND problem is NP-hard. Since the SC-MSOND problem is a particular case of the MSOND problem, we deduce that the MSOND problem is also NP-hard.
- Now, we will deal with the case of $|T_k|$ fixed. Clearly, if we take the simple case where for each $k \in K$, $|T_k| = 3$, the problem equivalent to the Steiner TSP (since there is no order constraint between the different terminal). As the Steiner TSP is NP-hard and constitutes a particular case of our problem, we then deduce that the MSOND problem is NP-hard as well.

Thus if $|S_k|$, $k \in K$ is input then the MSOND problem is NP-hard. $\square$

3.3.2.2 Case of $|K|$ input

In this case, we have the following result

Theorem 3.9 *The MSOND problem is NP-hard when the number of demands $|K|$ is inputand the number of Steiner node $|S_k|$, $k \in K$ is input, .*

Proof The result follows from Theorem 3.7 as the latter is a particular case of the one described in this theorem. □

Remark 3.10 1) When $|S_k|$ and $|T_k|$ for $k \in K$ are fixed, and $|K|$ is input, it is clear that we can apply Algorithm 3 and get a complexity of $\mathcal{O}(n^{|S||K|})$. This is polynomial since $|S_k|$ and $|T_k|$ are fixed for $k \in K$, and $|K|$ can not exceed $n(n-1)/2$ (which corresponds to all the possible demands in the graph).

2) When $|S|$ is fixed and $|T|$ and $|K|$ are input, we conjecture that the MSOND problem is NP-hard.

3.3.3 Summary table

In this section, we propose a recapitulation of the different cases of complexity of the MSOND problem in the case of single and multiple demands. Table 3.3.3 reports the different complexity results for the MSOND problem depending on the cardinality of Steiner nodes $|S|$, terminal nodes $|T|$ and demands $|K|$, whether they are fixed or input.

3.4 Concluding remarks

In this chapter, we have presented a problem related to survivability multilayer IP-over-Optical networks, called the Multilayer Survivable Optical Network Design Problem (MSOND problem). We have shown that the problem is very close to the classical Travelling Salesman Problem with its variants, Steiner TSP and TSP with precedence constraints. Our problem is also closed to the well known k-Vertex Disjoint Paths Problem. By a reduction of this problem, we have proved that the SC-MSOND problem is NP-hard when the number of terminals and Steiner nodes of the demand are input. For a fixed number of Steiner nodes, we have given a polynomial time algorithm to enumerate all the possible solutions of the problem. In the following chapters, we propose several integer linear programming formulations for the MSOND problem and develop efficient exact algorithms to solve random and realistic instances. We also carry on a polyhedral study of the polytope of one of these formulations, called the cut formulation, that will be present in the next Chapter.

	|T| fixed		**|T| input**	
	|S| fixed	**|S| input**	**|S| fixed**	**|S| input**
Single Commodity	Constant Algorithm 3 $\mathcal{O}(n^{\|S\|})$ Section 3.3.1.1	Polynomial Robertson and Seymour [114] Kawarabayashi [85] Section 3.3.1.2	Polynomial Algorithm 3 $\mathcal{O}(n^{\|S\|})$ Section 3.3.1.1	NP-hard Karp [84] Section 3.3.1.2
Many Commodities **|K| fixed**	Constant Algorithm 3 $\mathcal{O}(n^{\|S\|\|K\|})$ Section 3.3.2.1	NP-hard Steiner TSP Section 3.3.2.1	Polynomial Algorithm 3 $\mathcal{O}(n^{\|S\|\|K\|})$ Section 3.3.2.1	NP-hard generalization Section 3.3.2.1
Many Commodities **|K| input**	Polynomial Algorithm 3 $\mathcal{O}(n^{\|S\|\|K\|})$ Section 3.3.2.2	NP-hard generalization Section 3.3.2.2	NP-hard Section 3.3.2.2	NP-hard generalization Section 3.3.2.2

Table 3.2: Summary table of the complexity results for the SC-MSOND and MSOND problems

Chapter 4

Cut formulation and polyhedra

Contents

In this chapter, we consider the MSOND problem from a polyhedral point of view. We first propose an integer programming formulation for the problem. This uses two types of decision variables, design variables and demand variables. It is based on an exponential number of cut inequalities and is thus called cut formulation. In a second step, we study the polytope associated to this formulation. We characterize its dimension and study the facial aspect of its basic constraints. We then introduce further classes of valid inequalities and describe necessary and sufficient conditions for these inequalities to be facet defining. This investigation will give a good base for the algorithmic study that will be presented in chapter 5.

4.1 Cut formulation

In this section, we present an integer programming formulation for the MSOND problem using two sets of variables. The first set of variables, called *demand variables*, represent the edges used to route the two paths of each demand. The second set represents the edges that are considered in the solution and that can be used to route one or more demands. These are called *design variables*.
Let $x \in \mathbb{R}^{m|K|}$ such that for each demand $k \in K$ and each arc $e \in E$

$$x_e^k = \begin{cases} 1 & \text{if demand}\, k\, \text{is routed using edge}\, e, \\ 0 & \text{otherwise,} \end{cases}$$

and let $y \in \mathbb{R}^m$ such that for each $e \in E$,

$$y_e = \begin{cases} 1 & \text{if edge}\, e\, \text{is installed,} \\ 0 & \text{otherwise.} \end{cases}$$

An instance of the MSOND problem corresponds to the triplet (G, K, T). Let $\mathcal{S}(G, K, T)$ define the set of the feasible solutions of the MSOND problem associated with the graph G, the demand set K and the set of the terminals of the different demands $T = \bigcup_{k \in K} T_k$. A vector (x, y) induced by a solution of $\mathcal{S}(G, K, T)$ satisfies the following constraints:

$$\sum_{e \in \delta_{G^{k,j}}(W)} x_e^k \geq 1 \qquad \begin{array}{l} \text{for all}\, k \in K,\ q_j^k = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k, \\ W \subset V^{k,j} :\ w_j^k \in W \; and \; w_{j+1}^k \in \overline{W}, \end{array} \tag{4.1}$$

$$\sum_{e \in \delta(w)} x_e^k \leq 2 \qquad \text{for all}\, w \in V,\ k \in K, \tag{4.2}$$

$$x_e^k \leq y_e \qquad \text{for all}\, e \in E,\ k \in K, \tag{4.3}$$

$$0 \leq x_e^k \qquad \text{for all}\, e \in E,\ k \in K, \tag{4.4}$$

$$y_e \leq 1 \qquad \text{for all}\, e \in E. \tag{4.5}$$

Inequalities (4.1) are called *section cut inequalities*. They ensure for each section $q_j^k = (w_j^k, w_{j+1}^k)$ of a demand k a path in the reduced graph $G^{k,j}$. Recall that the reduced graph $G^{k,j}$ is obtained by deleting all the terminals T_k of the demand except the extremities w_j^k and w_{j+1}^k of section q_j^k. Hence, this guarantees for each demand two paths routing it and going in order through its terminals. Inequalities (4.2) are called *disjunction inequalities*. They ensure that the different sections of a demand k are disjoint and hence that the associated paths are node-disjoint. Inequalities (4.3) are the *linking inequalities* which express the fact that if an edge $e \in E$ is not installed, that is $y_e = 0$, then it can not be used to route any demand $k \in K$. Finally, inequalities (4.4) and (4.5) are the *trivial inequalities*.

Theorem 4.1 *The set* $\{(x, y) \in \{0, 1\}^{m(|K|+1)} : (x, y)$ *satisfies* $(4.1) - (4.5)\}$ *corresponds to the set* $\mathcal{S}(G, K, T)$ *of the feasible solutions of the MSOND problem.*

Proof It is clear that the incidence vector of any solution of the MSOND problem satisfies inequalities (4.1)-(4.5).
Let (x, y) be a vector in $\{0, 1\}^{m(|K|+1)}$ that does not induce a feasible solution of the MSOND problem. Suppose that (x, y) satisfies inequalities (4.1) and (4.3). In what follows, we show that there is at least one inequality of type (4.2) that is violated by (x, y). Consider a demand $k \in K$. By inequalities (4.1), we know that for each section $q_j^k = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k$, there exists a sub-path between its extremities w_j^k and w_{j+1}^k. This means that, for demand k, there are two paths going through its terminals. Moreover, since the sub-paths of each section of demand k are considered in the reduced graphs, the two paths routing demand k satisfy the order of the terminals of T_k. AS (x, y) is not feasible for the MSOND problem, there are at least two sections of demand k that are not node-disjoint, say q_j^k and q_i^k. Hence, q_j^k and q_i^k share at least one Steiner node of S^k. As q_j^k and q_i^k have different terminals, one of the shared nodes, say s, must be of degree at least three in the subgraph induced by q_j^k and q_i^k. Consequently, inequality (4.2), corresponding to s, is violated and the result follows. □

An immediate consequence of Theorem 4.1 is the following.

Corollary 4.2 *The MSOND problem is equivalent to the following integer program*

$$\min\{cy \,|\, (x, y) \in \{0, 1\}^{m(|K|+1)} : (x, y) \text{ satisfies} (4.1) - (4.5)\} \tag{4.6}$$

Problem 4.6 will be called the *cut formulation*.

Theorem 4.3 *The linear relaxation of* (4.6) *can be solved in polynomial time.*

Proof Since there is a polynomial number of constraints (4.2)-(4.5), the complexity of the linear relaxation of (4.6) depends only on the one of the separation problem of inequalities (4.1). Let us denote by $(\bar{x}, \bar{y})$ the solution to be separated. The separation of constraints (4.1) can be reduced to the calculation of $\sum_{k \in K} |\mathcal{T}_k|$ minimum cuts in graphs with weights $\bar{x}^k \geq 0, k \in K$. This can be done in polynomial time. □

4.2 Associated polytope

In this section, we study the convex hull of the solutions of the MSOND problem. We characterize its dimension and discuss necessary and sufficient conditions for inequalities (4.1)- (4.5) to be facet defining.

Recall that an instance of the MSOND problem corresponds to the triplet (G, K, T), where G is a graph, K a set of demands, each demand has a set T_k of terminals and $T = \bigcup_{k \in K} T_k$. Each solution in $\mathcal{S}(G, K, T)$ is represented by the pair (U, I) where $I \subseteq E$ is the set of *installed edges* and $U = (U_1, U_2, ..., U_{|K|}) \subset E^{|K|}$ such that $U_j, j = 1, ..., |K|$, is the set of *used edges* for demand j. Given a solution $(U, I) \in \mathcal{S}(G, K, T)$, we define (x^U, y^I) its incidence vector. Note that $x^U = (x^{U_1}, x^{U_2}, ..., x^{U_K})$.

Now, we present a solution for the MSOND problem that will be used throughout this chapter.

Remark 4.4 Assume that G is a complete graph. Let $U^0 = (U_1^0, U_2^0, ..., U_{|K|}^0)$ be the set of edges between the successive terminals of all the demands, that is $U_j^0 = \{w_i^j w_{i+1}^j, j \in K, i = 1, ..., |T_j|\}$. And denote $I^0 = \bigcup_{j \in K} U_j^0$. The pair (U^0, I^0) is a solution of the MSOND problem.

We denote by MSOND(G, K, T) the polytope associated with the MSOND problem, that is the convex hull of the solutions of formulation (4.6) corresponding to K and T in G, i.e.

$$\text{MSOND}(G, K, T) = conv\{c^T y \,|\, (x, y) \in \{0, 1\}^{m(|K|+1)} : (x, y) \,\text{satisfies}\, (4.1) - (4.5)\}.$$

In the following sections, we study the dimension of MSOND(G, K, T) and the facial aspect of the constraint (4.1)-(4.5).

4.2.1 Dimension

In this section, we characterize the dimension of the polytope MSOND(G, K, T). We first begin by giving some definitions and setting some assumptions which are necessary for the sequel of the section.

Consider a demand $k \in K$ and suppose that k does not have Steiner nodes. This means that $T_k = V$ and $S_k = \emptyset$. Note that in this case, the only alternative to route demand k is to route each section $q_j^k = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k$ using the edge $w_j^k w_{j+1}^k$. Here, every solution of the MSOND problem will use the edges $w_j^k w_{j+1}^k$, $j = 1, ..., |T_k|$, which implies that these edges are *essential* for demand k.

More generally, consider an edge $e \in E$ and a section $q_j^k = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k$ of demand k. Assume that every $w_j^k w_{j+1}^k$-cut of $G^{k,j}$, separating the terminals w_j^k and w_{j+1}^k reduces to edge e. That is to say the $w_j^k w_{j+1}^k$-cut is of a cardinality 1. Hence, edge e is an *essential* edge for demand k.

In the following, we summarize these cases and give a general characterization of essential edges for a demand $k \in K$.

Definition 4.5 *An edge $e \in E$ is said to be essential if there exists a section $q_j^k = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k$, $j = 1, ..., |T_k|$ of a demand $k \in K$, and a $w_j^k w_{j+1}^k$-cut of $G^{k,j}$ consisting of e.*

We will denote by E^* the set of essential edges in graph G.
Note that if $G = (V, E)$ is complete and for each demand $k \in K$ $T_k \neq V$, then $E^* = \emptyset$. Throughout the manuscript, we will restrict ourselves to that case. These assumptions are not restrictive. First, because if the graph is not complete, one can consider a complete graph by associating very hight costs to the non-existent edges. Moreover, if there exist a demand $k \in K$ such that $S_k = \emptyset$, then the solution is unique for this demand. In this case, the problem reduces to solving the MSOND problem for the $K \setminus \{k\}$ remaining demands. From a practical point of view, the terminals of a demand $k \in K$ could never coincide with all the vertices V of the graph. Furthermore, for convenience, we will suppose that each demand $k \in K$ has at least 2 Steiner nodes in S_k. This will enable to considerably simplify some facets proofs.

Under these hypotheses, we give in the following a characterization of the dimension of MSOND(G, K, T). To this end, we first identify the system of equation of the MSOND(G, K, T) polytope. We then prove that every equation of MSOND(G, K, T) is a linear combination of this system.

Proposition 4.6 *Consider a demand $k \in K$ and let $w_j^k \in T_k$ be a terminal of demand k. Then w_j^k has a degree equal to 2 in any solution.*

$$\sum_{e \in \delta(w_j^k)} x_e^k = 2 \qquad \text{for all } w_j^k \in T_k,\ k = 1, ..., K. \tag{4.7}$$

Proof Consider a solution (x, y) of the MSOND problem. Consider a demand k and let $w_j^k \in T_k$ be one of its terminals. By constraints (4.1), there is a path between w_{j-1}^k and w_j^k and a path between w_j^k and w_{j+1}^k. Moreover, by constraints (4.2), these paths are node-disjoint. Hence, $\sum_{e\in\delta(w_j^k)} x_e^k \geq 2$ The result follows by constraints (4.2).

□

Remark 4.7 Since equalities (4.7) are written for each terminal $w_j^k \in T_k$, there are $|T_k|$ equations for each demand $k \in K$.

Proposition 4.8 *Consider a demand $k \in K$ and let $w_i^k, w_j^k \in T_k$ be two non-successive terminals of k. Then edge $e = w_i^k w_j^k$ does not belong to any solution of the MSOND problem. That is*

$$x_e^k = 0 \qquad \text{for all } e = w_i^k w_j^k,\ w_i^k, w_j^k \in T_k : j > i,\ k = 1, ..., K. \tag{4.8}$$

Proof By constraints (4.1), there is a path between w_{i-1}^k and w_i^k and a path between w_i^k and w_{i+1}^k. Moreover, by equation (4.6), the terminal w_i^k has a degree equal exactly to 2. This implies that edge e could not be used in any solution of the problem. □

Remark 4.9 Let $G(T_k)$ be the graph induced by the terminals T_k of demand k. Notice that, since G is complete, $G(T_k)$ is complete and contains $\frac{|T_k|(|T_k|-1)}{2}$ edges. In this graph, there are exactly $|T_k|$ edges between the successive terminals. All the remaining edges are between non-successive terminals and in consequence their number is exactly equal to $\frac{|T_k|(|T_k|-1)}{2} - |T_k| = \frac{|T_k|(|T_k|-3)}{2}$. Consequently, constraints (4.8) imply $\frac{|T_k|(|T_k|-3)}{2}$ equations for demand k. In the sequel, we shall denote by p_k the number of edges between non-successive terminals for demand k, that is $p_k = \frac{|T_k|(|T_k|-3)}{2}$.

Proposition 4.10 *Consider an equation $ax + by = \alpha$ of MSOND(G, K, T). Then:*

1) $b = 0$,

2) $ax = \alpha$ is a linear combination of equations (4.7) and (4.8).

Proof Let $ax + by = \alpha$ be an equation of MSOND(G, K, T) where (x, y) is a solution of the MSOND problem such that $x = (x^1, x^2, .., x^{|K|}) \in \{0, 1\}^{mK}$, $y \in \{0, 1\}^m$ and (a, b) are vectors such that $a = (a^1, a^2, ..., a^{|K|}) \in \mathbb{R}^{mK}$ and $b \in \mathbb{R}^m$.

We first show that $b = 0$, which means that all the possible equations of the polytope MSOND(G, K, T) are expressed only in terms of the x variables.

Consider the feasible solution (U^0, I^0) given by Remark ??. Recall that U_j^0, $j = 1, ..., K$ gives the paths routed on the edges linking the successive terminals of T_j of demand $j \in K$; $I^0 = \bigcup_{j \in K} U_j^0$. Let $e \in E \setminus I^0$ be an arbitrary edge. The solution (U^1, I^1) such that $I^1 = I^0 \cup \{e\}$ and $U^1 = U^0$ also induces a feasible solution of MSOND(G, K, T). This implies that the incidence vectors of these solutions satisfy equation $ax + by = \alpha$. Consequently, $ax^{U^0} + by^{I^0} = ax^{U^1} + by^{I^1} = ax^{U^0} + by^{I^0} + b_e$, implying that $b_e = 0$. Since e is arbitrarily chosen in $E \setminus I^0$, we have

$$b_e = 0 \qquad \text{for all } e \in E \setminus I^0. \tag{4.9}$$

In a similar way, let us consider the solution (U^2, I^2) obtained as follows. Consider first a demand $k \in K$ and let $e = t_i t_{i+1}$ be an edge between two successive terminals t_i and t_{i+1} of T_k. Let $U_k^2 = (U_k^0 \setminus \{e\}) \cup \{t_i s, s t_{i+1}\}$ where $s \in S_k$ is a Steiner node of demand k. In addition, let $U_j^2 = U_j^0$, $j = 1, ..., K, j \neq k$, and $I^2 = \bigcup_{j \in K} U_j^2$. Now, let us define the solution (U^3, I^3) given by $U^3 = U^2$ and $I^3 = I^2 \cup \{e\}$. Both are feasible solutions for the MSOND problem and hence satisfy equation $ax + by = \alpha$. As a consequence, we have $ax^{U^2} + by^{I^2} = ax^{U^3} + by^{I^3} = ax^{U^2} + by^{I^2} + b_e$, which implies that $b_e = 0$. Recall that I^0 is the set of edges between the consecutive terminals of all the demands. As e is arbitrary in I^0, this yields

$$b_e = 0 \qquad \text{for all } e \in I^0. \tag{4.10}$$

By (4.9) and (4.10) we then have

$$b_e = 0 \qquad \text{for all } e \in E. \tag{4.11}$$

Therefore all the equations of MSOND(G, K, T) are given only in terms of the demand variables x.

Let a^k be the restriction of a on demand $k \in K$. Let $M^k = \begin{pmatrix} M_1^k \\ M_2^k \end{pmatrix}$ be the sub-matrix of equations (4.7) and (4.8) involving variables x_e^k,$e \in E$, such that M_1^k is the sub-matrix corresponding to equation (4.7) and M_2^k the one corresponding to (4.8). To prove that $ax = \alpha$ is a linear combination of equations (4.7) and (4.8), it is sufficient to prove that for each $k \in K$, there exit λ_1^k and λ_2^k such that $a^k = \lambda_1^k M_1^k + \lambda_2^k M_2^k$ where, $\lambda_1^k \in \mathbb{R}^{|T_k|}$ and $\lambda_2^k \in \mathbb{R}^{p_k}$ (recall that p_k is the number of edges between non-successive terminals of T_k).

First, we show that for every edge ss' between two Steiner nodes $s, s' \in S_k$, $a_{ss'}^k = 0$. Consider again the solution (U^0, I^0) and let (U^4, I^4) be the solution given by $U_k^4 =$

$U_k^0 \cup \{ss'\}$, $U_j^4 = U_j^0$ for all $j \in \{1, ..., K\} \setminus k$ and $I^4 = \bigcup_{j \in K} U_j^4$. Obviously, (U^4, I^4) is a solution of the MSOND problem. Since solutions (U^0, I^0) and (U^4, I^4) are both feasible for the MSOND problem and by (4.11) $b = 0$, we have $ax^{U^0} = ax^{U^4} = ax^{U^0} + a_e^k x_e^k$. Hence $a_e^k = 0$, implying that

$$a_{ss'}^k = 0 \qquad \text{for all } s, s' \in S_k, k \in K. \tag{4.12}$$

Consider now an edge $f = t_i t_{i+1}$ between two successive terminals t_i and t_{i+1} of T_k. Let s be a Steiner node of S_k. Consider the solution (U^5, I^5) given by $U_k^5 = (U_k^0 \setminus \{f\}) \cup \{t_i s, s t_{i+1}\}$, $U_j^5 = U_j^0$, for each $j \in \{1, ..., K\} \setminus k$, and $I^5 = \bigcup_{j \in K} U_j^5$. As (U^5, I^5) is feasible for the MSOND problem, it follows that $ax^{U^0} = ax^{U^5} = ax^{U^0} - a_{t_i t_{i+1}}^k + a_{t_i s}^k + a_{s t_{i+1}}^k$, implying that $a_{t_i t_{i+1}}^k = a_{t_i s}^k + a_{s t_{i+1}}^k$. Therefore, as nodes s, t_i and t_{i+1} are all arbitrary, we have

$$a_{t_i t_{i+1}}^k = a_{t_i s}^k + a_{s t_{i+1}}^k \qquad \text{for all } t_i, t_{i+1} \in T_k,\ s \in S_k, k \in K. \tag{4.13}$$

Now, we will prove that all the edges linking a terminal to Steiner nodes of S_k have the same coefficient a_e^k. Consider (U^6, I^6) obtained from (U^5, I^5) as follows: $U_k^6 = (U_k^5 \setminus \{s t_i\}) \cup \{ss', s' t_i\}$, $U_j^6 = U_j^5$ for each $j \in \{1, ..., K\} \setminus k$ and $I^6 = \bigcup_{j \in K} U_j^6$, where s and s' are Steiner nodes of S_k. Since (U^5, I^5) and (U^6, I^6) satisfy equation $ax + by = \alpha$, this implies $ax^{U^5} = ax^{U^6} = ax^{U^5} - a_{st_i}^k + a_{ss'}^k + a_{s't_i^k}^k$, and hence $a_{st_i}^k = a_{ss'}^k + a_{s't_i}^k$. By (4.12), it follows that

$$a_{st_i}^k = a_{s't_i}^k = \lambda_1^k(t_i) \qquad \begin{array}{l} \text{for all}\, t_i \in T_k,\ s, s' \in S_k, k \in K, \\ \text{for some } \lambda_1^k(t_i) \in \mathbb{R}. \end{array} \tag{4.14}$$

Now, let $\lambda^k = (\lambda_1^k, \lambda_2^k)$, $k \in K$ such that $\lambda_1^k = (\lambda_1^k(t_i), t_i \in T_k)$ where $\lambda_1^k(t_i)$ is as given by (4.14) and $\lambda_2^k = (\lambda_2^k(uv),\ u, v \in T_k, uv \notin U_k^0)$ such that $\lambda_2^k(uv) = a_{uv}^k - \lambda_1^k(u) - \lambda_1^k(v)$, $k \in K$. By (4.13), we have $a_{t_i t_{i+1}}^k = \lambda_1^k(t_i) + \lambda_1^k(t_{i+1})$ for each two successive terminals t_i and t_{i+1} of T_k.

The vectors λ_1^k and λ_2^k can be then given so that:

$$a_{uv}^k = \begin{cases} \lambda_1^k(u) + \lambda_1^k(v) & \text{if } uv = t_i t_{i+1},\ t_i, t_{i+1} \in T_k, \\ \lambda_1^k(u) & \text{if } u \in T_k,\ v \in S_k, \\ \lambda_2^k(uv) + \lambda_1^k(u) + \lambda_1^k(v) & \text{if } uv = t_i t_j,\ t_i, t_j \in T_k : j > i, \\ 0 & \text{if } uv = s_i s_j,\ s_i, s_j \in S_k : j \neq i, \end{cases}$$

yielding

$$a^k = \lambda_1^k M_1^k + \lambda_2^k M_2^k \qquad \text{for all } k \in K,$$

as desired. □

By Proposition 4.10, we know that the only equations of MSOND(G, K, T) are equations (4.7) and (4.8). The matrix M of equations of MSOND(G, K, T) can hence be written as

$$M = \begin{pmatrix} M_1 & & & \\ & M_2 & & \\ & & \ddots & \\ & & & M_K \end{pmatrix}$$

where M_k is the matrix of the equations system (4.7) and (4.8) involving variables x_e^k for demand $k \in K$. Each matrix $M_k, k = 1, ..., K$, consists of two sub-matrices. The first is an incidence matrix terminal-edge and the second is a restriction of the identity matrix for the edges between non-successive terminals. More precisely, each matrix M_k can be organized as in Figure 4.1. The first m_1 columns correspond to the edges between successive terminals ($m_1 = |T_k|$). The next m_2 columns correspond to the edges between non-successive terminals ($m_2 = p_k$). The last m_3 columns correspond to the edges between a terminal of T_k and a Steiner node of S_k. Here, the first columns are related to the Steiner node s_1, the second to s_2 and so on. The first $|T_k|$ rows of M_k are associated with the terminals of T_k, and the last ones are associated with the edges between non-successive terminals. Note that the first $|T_k|$ rows of M_k correspond to equations (4.7) and the last p_k ones correspond to equations (4.8). The sub-matrices $I_1, ..., I_{|S_k|}$ are identity matrices and are related to the steiner nodes $s_1, ..., s_{|S_k|}$ of S_k, respectively.

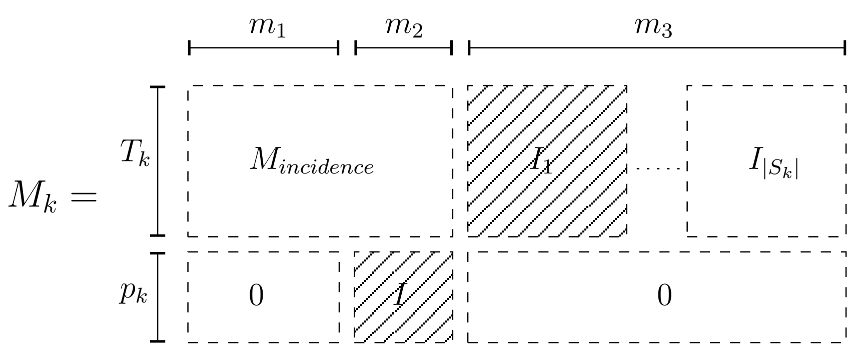


Figure 4.1: Matrix of equations for demand k

Observe that M_k contains two blocks consisting of two idendity matrices, namely I_1 and I (hatched matrices). Moreover, these matrices cover the rows of M_k. Hence M_k is of full rank equal to $|T_k| + p_k = \frac{|T_k|(|T_k|-1)}{2}$.

As M is a diagonal matrix where each block is a matrix M_k, it follows that $rank(M) = \sum\limits_{k \in K} rank(M_k) = \sum\limits_{k \in K} \frac{|T_k|(|T_k|-1)}{2}$.

As a consequence, we have the following result.

Theorem 4.11

$$dim(MSOND(G,K,T)) = (|K|+1)|E| - \sum_{k\in K} \frac{|T_k|(|T_k|-1)}{2}.$$

4.2.2 Facial investigation

In this section, we study the facial structure of the polytope MSOND(G,K,T). In particular, we give necessary and sufficient conditions for inequalities of formulation (4.6) to be facet defining.

4.2.2.1 Trivial inequalities

First, we will be interested in the non-negativity inequality $x_e^k \geq 0$. For convenience, we will suppose that each demand has at least 3 Steiner nodes. The following theorem gives a necessary and sufficient condition under which inequality $x_e^k \geq 0$ defines a facet of MSOND(G,K,T).

Theorem 4.12 *Inequality $x_e^k \geq 0$ defines a facet of MSOND(G,K,T) if and only if e is not between non-successive terminals of T_k.*

Proof Denote by F_e^k the face induced by inequality $x_e^k \geq 0$, that is

$$F_e^k = \{(x,y) \in \text{MSOND}(G,K,T) : x_e^k = 0\}.$$

The necessity condition is a consequence of Proposition 4.8.

In what follows, we suppose that e is not an edge between non-successive terminals of T^k.

Denote inequality $x_e^k \geq 0$ by $ax+by \leq \alpha$ and let $rx+qy \leq \beta$ be a valid inequality defining a facet F of MSOND(G,K,T). Assume that $F_e^k \subseteq F$. To prove that F_e^k is a facet of MSOND(G,K,T), it suffices to show that there exist $\rho \in \mathbb{R}$ and $\lambda = (\lambda^j, j \in K)$, $\lambda^j \in \mathbb{R}^{|T_j|+p_j}$ for $j \in K$, such that $q = \rho b$ and $r = \rho a + \lambda M$ (where $r = (r^1, r^2, ..., r^{|K|})$ with $r^j \in \mathbb{R}^m$, $j = 1, ..., |K|$ and M is the matrix of equations defined above). Note here that $a = (a^1, a^2, ..., a^{|K|})$ is such that $a^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ with $a^i = 0$ for $i \in \{1, ..., |K|\} \setminus \{k\}$, $a^k_e = 1$ and $a^k_{e'} = 0$ for $e \neq e'$. Note also that $b = 0$.

Case 1. $e = s_1 s_2$, where s_1 and s_2 are Steiner nodes of S_k.

First, we prove that $q = 0$.

Consider the solution (U^0, I^0) given in Remark 4.4 and let $e \in E \setminus I^0$. The solution (U^1, I^1) such that $I^1 = I^0 \cup \{e\}$ and $U^1 = U^0$ also induces a feasible solution of MSOND(G, K, T). Moreover, the incidence vectors of these solutions are in F^k_e and hence in F. This implies that these vectors satisfy equation $rx + qy = \beta$. Consequently, $rx^{U^0} + qy^{I^0} = rx^{U^1} + qy^{I^1} = qx^{U^0} + qy^{I^0} + q_e$, implying that $q_e = 0$. Since e is arbitrarily chosen in $E \setminus I^0$, we have

$$q_e = 0 \qquad \text{for all } e \in E \setminus I^0. \tag{4.15}$$

Now, consider the solution (U^2, I^2) obtained as follows. Consider a demand $l \in K$ and let $e = t_i t_{i+1}$ be an edge between two successive terminals t_i and t_{i+1} of S_l. Let $U^2_l = (U^0_l \setminus \{e\}) \cup \{t_i s, s t_{i+1}\}$ where $s \in S_l$ is a Steiner node of demand l. In addition, let $U^2_j = U^0_j$, $j = 1, ..., K, j \neq l$, and $I^2 = \bigcup_{j \in K} U^2_j$. Now, let us define the solution (U^3, I^3) given by $U^3 = U^2$ and $I^3 = I^2 \cup \{e\}$. Both are feasible solutions for the MSOND problem and are in F^k_e. Hence, they satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^2} + qy^{I^2} = rx^{U^3} + qy^{I^3} = rx^{U^2} + qy^{I^2} + q_e$, which implies that $q_e = 0$. As e is arbitrary in I^0, this yields

$$q_e = 0 \qquad \text{for all } e \in I^0. \tag{4.16}$$

By (4.15) and (4.16), we then have

$$q_e = 0 \qquad \text{for all } e \in E. \tag{4.17}$$

We now show that $a^j_{ss'}$ for all $s, s' \in S_j$ such that either $j = k$ and $ss' \neq e$ or $j \in K \setminus \{k\}$.

Consider an edge ss' between two Steiner nodes s and s' of S_k different from e. Let (U^4, I^4) be the solution defined as follows: $U^4_k = U^0_k \cup \{ss'\}$, $U^4_j = U^0_j$ for all $j \in \{1, ..., K\} \setminus \{k\}$ and $I^4 = \bigcup_{j \in K} U^4_j$. As $ss' \notin U^0_k$ and $ss' \notin U^4_k$, $(x^{U^0}, y^{I^0}), (x^{U^4}, y^{I^4}) \in F^k_e$ and consequently $(x^{U^0}, y^{I^0}), (x^{U^4}, y^{I^4}) \in F$. Hence, by (4.17), it follows that $rx^{U^0} = rx^{U^4}$. This implies that $rx^{U^0} = rx^{U^0} + r^k_{ss'}$, yielding to $r^k_{ss'} = 0$. As s and s' are arbitrary in S_k, we then obtain that

$$r^k_{ss'} = 0 \qquad \text{for all}\, ss' \in S_k, ss' \neq e. \tag{4.18}$$

Consider now a demand $l \in K \setminus \{k\}$ and suppose that s and s' are here Steiner nodes of S_l. In a similar way, let (U^5, I^5) be the solution obtained from (U^0, I^0) as follows, $U_l^5 = U_l^0 \cup \{ss'\}$, $U_j^5 = U_j^0$ for all $j \in \{1, ..., K\} \setminus \{l\}$ and $I^5 = \bigcup_{j \in K} U_j^5$. It is clear that $(x^{U^5}, y^{I^5}) \in F_e^k$ and hence $(x^{U^5}, y^{I^5}) \in F$. We then have $rx^{U^0} = rx^{U^5} = rx^{U^0} + r_{ss'}^l$ implying that $r_{ss'}^l = 0$. As s and s' are arbitrary in S_l, this yields

$$r_{ss'}^l = 0 \qquad \text{for all}\, ss' \in S_l, l \in K \setminus \{k\}. \tag{4.19}$$

Now, we will consider edges between successive terminals.

Consider a demand $l \in K$ and let $t_i t_{i+1}$ be an edge between two successive terminals t_i and t_{i+1} of T_l. Consider a Steiner node s of S_l and let (U^6, I^6) be the solution given by $U_l^6 = (U_l^0 \setminus \{t_i t_{i+1}\}) \cup \{t_i s, s t_{i+1}\}$, $U_j^6 = U_j^0$ for each $j \in \{1, ..., K\} \setminus \{l\}$ and $I^6 = \bigcup_{j \in K} U_j^6$. As (x^{U^6}, y^{I^6}) is in F_e^k and hence in F, by (4.17) we have $rx^{U^0} = rx^{U^0} - r_{t_i t_{i+1}}^l + r_{t_i s}^l + r_{s t_{i+1}}^l$, and consequently, $r_{t_i t_{i+1}}^l = r_{t_i s}^l + r_{s t_{i+1}}^l$. As l, t_i, t_{i+1} and s are chosen arbitrarily, we have

$$r_{t_i t_{i+1}}^l = r_{t_i s}^l + r_{s t_{i+1}}^l \qquad \text{for all}\, t_i, t_{i+1} \in T_l,\ s \in S_l, l \in K. \tag{4.20}$$

Now, we will prove that all the edges linking a terminal to Steiner nodes have the same coefficient.

Consider a demand $l \in K \setminus \{k\}$ and two Steiner nodes s and s' in S_l. Denote (U^7, I^7) the solution given by $U_l^7 = (U_l^6 \setminus \{st_i\}) \cup \{ss', s't_i\}$, $U_j^7 = U_j^6$ for each $j \in \{1, ..., K\} \setminus \{l\}$ and $I^6 = \bigcup_{j \in K} U_j^6$. Clearly, (x^{U^7}, y^{I^7}) is in F_e^k and thus in F. The incidence vectors of (U^6, I^6) and (U^7, I^7) satisfy equation $rx + qy = \beta$. As $q = 0$, this implies $rx^{U^6} = rx^{U^7} = rx^{U^6} - r_{st_i}^l + r_{ss'}^l + r_{s't_i}^l$. This leads to $r_{st_i}^l = r_{ss'}^l + r_{s't_i}^l$. By (4.19), we obtain $r_{st_{i+1}}^l = r_{s't_{i+1}}^l$. As demand l, and nodes s, s' and t_{i+1} are arbitrary, we then have

$$r_{st_i}^l = r_{s't_i}^l \qquad \text{for all}\, t_i \in T_l,\ s, s' \in S_l, l \in K \setminus \{k\}.$$

In a similar way, we can show that

$$r_{st_i}^k = r_{s't_i}^k \qquad \text{for all}\, t_i \in T_k,\ s, s' \in S_k,\ ss' \neq e.$$

The previous relation can be generalized even for $\{s, s'\} = \{s_1, s_2\}$. In fact, if we suppose that there is $\overline{s} \in S_k \setminus \{s_1, s_2\}$, we know that for every terminal t_i of T_k, $r_{t_i s_1}^k = r_{t_i \overline{s}}^k$ and $r_{t_i s_2}^k = r_{t_i \overline{s}}^k$. As a consequence, we have $r_{t_i s_1}^k = r_{t_i s_2}^k$.

Overall, we obtain that

$$r_{st_i}^l = r_{s't_i}^l = \lambda_1^l(t_i) \qquad \begin{array}{l} \text{for all}\, t_i \in T_l,\ s, s' \in S_l, l \in K, \\ \text{for some}\, \lambda_1^l(t_i) \in \mathbb{R}. \end{array} \tag{4.21}$$

Now, let $\rho = r^k_{s_1s_2}$ and $\lambda^l = (\lambda^l_1, \lambda^l_2)$, $l \in K$ such that $\lambda^l_1 = (\lambda^l_1(t_i), t_i \in T_k)$ where $\lambda^l_1(t_i)$ is as given by (4.21) and $\lambda^l_2 = (\lambda^l_2(uv),\ u, v \in T_k, uv \notin U^0_l)$ such that $\lambda^l_2(uv) = r^l_{uv} - \lambda^l_1(u) - \lambda^l_1(v)$, $l \in K$.

The coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can then be expressed in terms of ρ, λ^l_1 and λ^l_2 as follows

$$r^l_{uv} = \begin{cases} \lambda^l_1(u) + \lambda^l_1(v) & \text{if } uv = t_it_{i+1},\ t_i, t_{i+1} \in T_l, \\ \lambda^l_1(u) & \text{if } u \in T_l,\ v \in S_l, \\ \lambda^l_2(uv) + \lambda^l_1(u) + \lambda^l_1(v) & \text{if } uv = t_it_j,\ t_i, t_j \in T_l,\ j > i, \\ 0 & \text{if } l \neq k \text{ and } uv = s_is_j,\ s_i, s_j \in S_l,\ j \neq i, \\ 0 & \text{if } l = k \text{ and } uv = s_is_j,\ s_i, s_j \in S_k,\ j \neq i,\ s_is_j \neq s_1s_2, \\ \rho & \text{if } l = k \text{ and } uv = s_1s_2. \end{cases}$$

Clearly, $r^l = \rho a^l + \lambda^l_1 M^l_1 + \lambda^l_2 M^l_2$, for all $l \in K$. As a consequence, $r = \rho a + \lambda M$.

Case 2. $e = s_1t_1$ where s_1 is a Steiner node of S^k and t_1 a terminal of T^k.

First, similarly to Case 1., we can prove that

$$q_e = 0 \qquad \text{for all } e \in E. \tag{4.22}$$

Now, we will find some relations linking the components of vector r.

First, we will consider the edges between Steiner nodes.

Consider a demand $l \in K$ and let ss' be an edge between two Steiner nodes s and s' of S^l. Consider the solutions (U^0, I^0) and (U^5, I^5) introduced above. Clearly, (x^{U^0}, y^{I^0}) and (x^{U^5}, y^{I^5}) are in F^k_e and thus in F. As a consequence, we have $rx^{U^0} = rx^{U^5} = rx^{U^0} + r^l_{ss'}$ and hence $r^l_{ss'} = 0$. As s and s' are arbitrarily in S_l, we have

$$r^l_{ss'} = 0 \qquad \text{for all}\, s, s' \in S_l, l \in K. \tag{4.23}$$

Now, consider the edges between successive terminals.

Suppose now that demand $l \neq k$. Consider an edge t_it_{i+1} between two successive terminals t_i and t_{i+1} of T_l and let s be a Steiner node of S^l. Consider again the solution (U^6, I^6) defined as follows: $U^6_l = (U^0_l \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U^6_j = U^0_j$ for each $j \in \{1, ..., K\} \setminus l$ and $I^6 = \bigcup_{j \in K} U^6_j$. As (x^{U^0}, y^{I^0}) and (x^{U^6}, y^{I^6}) are in F^k_e and hence in F, we have $rx^{U^0} = rx^{U^6} = rx^{U^0} - r^l_{t_it_{i+1}} + r^l_{t_is} + r^l_{st_{i+1}}$. This yields

$r^l_{t_it_{i+1}} = r^l_{t_is} + r^l_{st_{i+1}}$. Remind that demand l and nodes t_i, t_{i+1} and s are all arbitrary. Therefore we have

$$r^l_{t_it_{i+1}} = r^l_{t_is} + r^l_{st_{i+1}} \qquad \text{for all}\, t_i, t_{i+1} \in T_l,\ s \in S_l, l \in K \setminus \{k\}. \tag{4.24}$$

Consider now demand k. Consider an edge t_it_{i+1} between two successive terminals t_i and t_{i+1} of T_k and let s be a Steiner node of S_k such that $t_is \neq e$ and $st_{i+1} \neq e$. Similarly, let (U^8, I^8) be the solution defined as follows $U^8_k = (U^0_k \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U^8_j = U^0_j$ for each $j \in \{1, ..., K\} \setminus \{k\}$ and $I^8 = \bigcup_{j\in K} U^8_j$. As (x^{U^0}, y^{I^0}) and (x^{U^8}, y^{I^8}) are in F^k_e and hence in F, we have we have $rx^{U^0} = rx^{U^8} = rx^{U^0} - r^k_{t_it_{i+1}} + r^k_{t_is} + r^k_{st_{i+1}}$, implying that $r^k_{t_it_{i+1}} = r^k_{t_is} + r^k_{st_{i+1}}$. As nodes t_i, t_{i+1} and s are all arbitrary for demand k, we obtain

$$r^k_{t_it_{i+1}} = r^k_{t_is} + r^k_{st_{i+1}} \qquad \text{for all}\, t_i, t_{i+1} \in T_k,\ s \in S_k,\ t_is \neq e \neq st_{i+1}. \tag{4.25}$$

In what follows, we will prove that all the edges linking a terminal to Steiner nodes have the same coefficient.

Consider again an arbitrary demand $l \in K \setminus \{k\}$. Consider two Steiner nodes s and s' of S_l and two successive terminals $t_i, t_{i+1} \in T_l$. Let (U^7, I^7) be the solution defined above. Obvisouly, (x^{U^6}, y^{I^6}) and (x^{U^7}, y^{I^7}) are in F^k_e and consequently in F. This, together with (4.23), allow us to write $r^l_{st_i} = r^l_{s't_i}$. As demand l and nodes s, s' and t_i are arbitrary, it follows that

$$r^l_{st_i} = r^l_{s't_i} = \lambda^l_1(t_i) \qquad \begin{array}{l}\text{for all}\, t_i \in T_l,\ s, s' \in S_l,\ l \in K \setminus \{k\},\\ \text{for some}\, \lambda^l_1(t_i) \in \mathbb{R}.\end{array} \tag{4.26}$$

Consider now demand k, the Steiner nodes $s, s' \in S_k$ and two successive terminals $t_i, t_{i+1} \in T_k$ such that $t_is \neq e$ and $s't_{i+1} \neq e$. Along the same line, we can show that

$$r^k_{st_i} = r^k_{s't_i} = \lambda^k_1(t_i) \qquad \begin{array}{l}\text{for all}\, t_i \in T_k,\ s, s' \in S_k,\ t_is \neq e \neq s't_i,\\ \text{for some}\, \lambda^k_1(t_i) \in \mathbb{R}.\end{array} \tag{4.27}$$

Now, let $\rho = r^k_{s_1t_1} - \lambda^l_1(t_1)$ and $\lambda^l = (\lambda^l_1, \lambda^l_2)$, $l \in K$ such that $\lambda^l_1 = (\lambda^l_1(t_i), t_i \in T_l)$ where $\lambda^l_1(t_i)$ is as given by (4.26) and (4.27) and $\lambda^l_2 = (\lambda^l_2(uv),\, u, v \in T_k, uv \notin U^0_l)$ such that $\lambda^l_2(uv) = r^l_{uv} - \lambda^l_1(u) - \lambda^l_1(v)$, $l \in K$.

Overall, the coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can be expressed in terms of ρ,

λ_1^l and λ_2^l as follows

$$r_{uv}^l = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_i t_{i+1},\ t_i, t_{i+1} \in T_l, \\ \lambda_1^l(u) & \text{if } l \neq k \text{ and } u \in T_l,\ v \in S_l, \\ \lambda_1^l(u) & \text{if } l = k \text{ and } u \in T_k,\ v \in S_k,\ uv \neq s_1 t_1, \\ \rho + \lambda_1^l(u) & \text{if } l = k \text{ and } u \in T_k,\ v \in S_k,\ uv = s_1 t_1, \\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_i t_j,\ t_i, t_j \in T_l,\ j > i, \\ 0 & \text{if } uv = s_i s_j,\ s_i, s_j \in T_l,\ j \neq i. \end{cases}$$

Clearly, we have $r = \rho a + \lambda M$.

Case 3. $e = t_1 t_2$, where s_1 and s_2 are terminals of T_k.

Along the same line of the previous cases, we can show that

$$q_e = 0 \qquad \text{for all } e \in E. \tag{4.28}$$

By constraints (4.1), there must exist a path between the terminals t_1 and t_2. Hence any solution of F_e^k must necessarily use a subset of Steiner nodes of S_k. For the solutions that will be used in the sequel of the proof, we suppose that section (t_1, t_2) is routed by the path $(t_1 s_1, s_1 t_2)$, where s_1 is a Steiner node of S_k.

Now, we establish some relations linking the components of vector r.

First, we consider the edges between Steiner nodes.

Consider a demand $l \in K \setminus \{k\}$ and let ss' be an edge between two Steiner nodes s and s' of S_l. Let (U^9, I^9) be the solution defined by: $U_k^9 = (U_k^0 \setminus \{t_1 t_2\}) \cup \{t_1 s_1, s_1 t_2\}$, $U_j^9 = U_j^0$ for each $j \in \{1, ..., K\} \setminus k$ and $I^9 = \bigcup_{j \in K} U_j^9$. Now,let us define the solution (U^{10}, I^{10}) as follows: $U_l^{10} = U_l^9 \cup \{ss'\}$, $U_j^{10} = U_j^9$ for each $j \in \{1, ..., K\} \setminus l$ and $I^{10} = \bigcup_{j \in K} U_j^{10}$. As (x^{U^9}, y^{I^9}) and $(x^{U^{10}}, y^{I^{10}})$ are in F_e^k and hence in F, they satisfy equation $rx + qy = \beta$. Since $q = 0$, it follows that $rx^{U^9} = rx^{U^{10}} = rx^{U^9} + r_{ss'}^l$, yielding $r_{ss'}^l = 0$. As demand l and nodes s and s' are all aribtrary, we have

$$r_{ss'}^l = 0 \qquad \text{for all}\, s, s' \in S_l, l \in K \setminus \{k\}. \tag{4.29}$$

Now, we suppose that s and s' are Steiner nodes of $S_k \setminus \{s_1\}$. In a similar way, we prove that

$$r_{ss'}^k = 0 \qquad \text{for all}\, s, s' \in S_k \setminus s_1.$$

Since s_1 is arbitrarily in S_k, we can construct further solutions by inserting other Steiner nodes between t_1 and t_2. This implies that

$$r^k_{ss'} = 0 \qquad \text{for all } s, s' \in S_k. \tag{4.30}$$

Now, we will consider edges between terminals.

Consider an edge $t_i t_{i+1}$ between two successive terminals t_i and t_{i+1} of T_k such that $t_i t_{i+1} \neq t_1 t_2$ and let s be a Steiner node in $S_k \backslash \{s_1\}$. Consider again the solution (U^9, I^9) and let (U^{11}, I^{11}) be the solution defined as follows $U^{11}_k = (U^{11}_k \setminus \{t_i t_{i+1}\}) \cup \{t_i s, s t_{i+1}\}$, $U^{11}_j = U^9_j$ for each $j \in \{1, ..., K\} \backslash \{k\}$ and $I^{11} = \bigcup_{j \in K} U^{11}_j$. As (x^{U^9}, y^{I^9}) and $(x^{U^{11}}, y^{I^{11}})$ are in F^k_e and hence in F, and s_1 is chosen arbitrarily in S_k, we have $rx^{U^9} = rx^{U^{11}} = rx^{U^9} - r^k_{t_i t_{i+1}} + r^k_{t_i s} + r^k_{s t_{i+1}}$. This implies that $r^k_{t_i t_{i+1}} = r^k_{t_i s} + r^k_{s t_{i+1}}$. And since nodes t_i, t_{i+1} and s are all arbitrary, we obtain

$$r^k_{t_i t_{i+1}} = r^k_{t_i s} + r^k_{s t_{i+1}} \qquad \begin{array}{l} \text{for all } s \in S_k, \\ \text{for all } t_i, t_{i+1} \in T_k, t_i t_{i+1} \neq t_1 t_2. \end{array} \tag{4.31}$$

Along the same line, we can prove that

$$r^l_{t_i t_{i+1}} = r^l_{t_i s} + r^l_{s t_{i+1}} \quad \text{for all } t_i, t_{i+1} \in T_l,\ s \in S_l, l \in K \setminus \{k\}. \tag{4.32}$$

In the sequel, we consider edges between terminals and Steiner nodes.

Consider two Steiner nodes s and s' in $S_k \backslash \{s_1\}$. Let (U^{12}, I^{12}) be the solution given as follows $U^{12}_k = (U^{11}_k \setminus \{s t_{i+1}\}) \cup \{ss', s' t_{i+1}\}$, $U^{12}_j = U^{11}_j$ for each $j \in \{1, ..., K\} \setminus \{k\}$ and $I^{12} = \bigcup_{j \in K} U^{12}_j$. $(x^{U^{11}}, y^{I^{11}})$ and $(x^{U^{12}}, y^{I^{12}})$ are in F^k_e, they are consequently in F. This implies that $(x^{U^{11}}, y^{I^{11}})$ and $(x^{U^{12}}, y^{I^{12}})$ satisfy equation $rx + qy = \beta$. Consequently, we can write $rx^{U^{11}} = rx^{U^{12}} = rx^{U^{11}} - r^k_{s t_i} + r^k_{ss'} + r^k_{s' t_i}$ yielding $r^k_{s t_i} = r^k_{ss'} + r^k_{s' t_i}$. By (4.30) and as s_1 is chosen arbitrarily in S_k, we have

$$r^k_{s t_i} = r^k_{s' t_i} \qquad \text{for all } t_i \in T_k,\ s, s' \in S_k.$$

Similarly, we can show that

$$r^l_{s t_i} = r^l_{s' t_i} \qquad \text{for all } t_i \in T_l,\ s, s' \in S_l, l \in K \setminus \{k\}.$$

Overall, and as nodes t_i, s and s' as well as demand l are all arbitrary, we obtain that

$$r^l_{s t_i} = r^l_{s' t_i} = \lambda^l_1(t_i) \qquad \begin{array}{l} \text{for all } t_i \in T_l,\ s, s' \in S_l, l \in K, \\ \text{for some } \lambda^l_1(t_i) \in \mathbb{R}. \end{array} \tag{4.33}$$

Now, let $\rho = r^k_{t_1t_2} - \lambda^l_1(t_1) - \lambda^l_1(t_2)$ and $\lambda^l = (\lambda^l_1, \lambda^l_2)$, $l \in K$ such that $\lambda^l_1 = (\lambda^l_1(t_i), t_i \in T_k)$ where $\lambda^l_1(t_i)$ is as given by (4.33) and $\lambda^l_2 = (\lambda^l_2(uv),\ u, v \in T_k, uv \notin U^0_l)$ such that $\lambda^l_2(uv) = r^l_{uv} - \lambda^l_1(u) - \lambda^l_1(v)$, $l \in K$.

The coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can be expressed in terms of ρ, λ^l_1 and λ^l_2 as follows

$$r^l_{uv} = \begin{cases} \lambda^l_1(u) + \lambda^l_1(v) & \text{if } l \neq k \text{ and } uv = t_it_{i+1},\ t_i, t_{i+1} \in T_l, \\ \lambda^l_1(u) + \lambda^l_1(v) & \text{if } l = k \text{ and } uv = t_it_{i+1},\ t_i, t_{i+1} \in T_k, uv \neq t_1t_2 \\ \rho + \lambda^l_1(u) + \lambda^l_1(v) & \text{if } l = k \text{ and } uv = t_1t_2, \\ \lambda^l_1(u) & \text{if } u \in T_l,\ v \in S_l, \\ \lambda^l_2(uv) + \lambda^l_1(u) + \lambda^l_1(v) & \text{if } uv = t_it_j,\ t_i, t_j \in T_l,\ j > i, \\ 0 & \text{if } uv = s_is_j,\ s_i, s_j \in T_l,\ j \neq i. \end{cases}$$

Clearly, we have $r = \rho a + \lambda M$, which ends the proof.

□

Now, we study the facial structure of the trivial constraints $y_e \leq 1$, $e \in E$.

Theorem 4.13 *Inequality $y_e \leq 1$ defines a facet of MSOND(G, K, T).*

Proof Let F_e be the corresponding induced face, that is

$$F_e = \{(x, y) \in \text{MSOND}(G, K, T) : y_e = 1\}.$$

Denote inequality $y_e \leq 1$ by $ax + by \leq \alpha$. Let $rx + qy \leq \beta$ be a valid inequality defining a facet F of MSOND(G, K, T). Assume that $F_e \subseteq F$. We prove that there exist $\rho \in \mathbb{R}$ and $\lambda = (\lambda^j, j \in K)$, $\lambda^j \in \mathbb{R}^{|T_j|+p_j}$ for $j \in K$, such that $q = \rho b$ and $r = \rho a + \lambda M$ (where $r = (r^1, r^2, ..., r^{|K|})$ with $r^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ and M is the matrix of equations defined above). Note here that $b_{e'} = 0$ for all $e' \in E \setminus \{e\}$. Moreover, $a = (a^1, a^2, ..., a^{|K|})$ is such that $a^i \in \mathbb{R}^m$ with $a^i = 0$, $i = 1, ..., |K|$.

In the sequel, we will distinguish two cases.

Case 1. $e \in I^0$.

Suppose $e = t_1t_2$ is an edge between two successive terminals t_1 and t_2 of a demand $k \in K$. We first show that every edge f in $E \setminus I^0$ has a coefficient q_f equal to 0.

Consider the solution (U^0, I^0) and let (U^1, I^1) be the solution defined by $U^1 = U^0$ and $I^1 = I^0 \cup \{f\}$. It is clear that (x^{U^0}, y^{I^0}) and (x^{U^1}, y^{I^1}) are in F_e and hence in F. Therefore, $rx^{U^0} + qy^{U^0} = rx^{U^1} + qy^{U^1} = rx^{U^0} + qy^{U^0} + q_f$ and thus $q_f = 0$. As f is arbitrary in $E \setminus I^0$, this implies that

$$q_f = 0 \qquad \text{for all } f \in E \setminus I^0. \tag{4.34}$$

Now, consider a demand $l \in K$, and let $t_it_{i+1} \in I^0$ be an edge of I^0 such that $t_it_{i+1} \neq e$. Consider a Steiner node s of S_l and let (U^2, I^2) be the solution given by $U_l^2 = (U_l^0 \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U_j^2 = U_j^0$, $j \in \{1, ..., |K|\} \setminus \{l\}$ and $I^2 = \bigcup_{j \in K} U_j^2$. And let (U^3, I^3) be the solution defined by $U^3 = U^2$ and $I^3 = I^2 \cup \{t_it_{i+1}\}$. As (x^{U^2}, y^{I^2}) and (x^{U^3}, y^{I^3}) are in F_e and thus in F, we have $rx^{U^2} + qy^{I^2} = rx^{U^3} + qy^{I^3} = rx^{U^2} + qy^{I^2} + q_{t_it_{i+1}}$, which implies that $q_{t_it_{i+1}} = 0$. As t_i and t_{i+1} are arbitrary in T_l, and l is arbitrary in K, we obtain

$$q_f = 0 \qquad \text{for all } f \in I^0 \setminus \{e\}. \tag{4.35}$$

Next, we will establish some relations between the components of vector r.

Consider a demand $l \in K$ and let s and s' be two Steiner nodes of S_l. Consider again the solution (U^0, I^0) and let (U^4, I^4) be the solution defined as follows: $U_l^4 = U_l^0 \cup \{ss'\}$, $U_j^4 = U_j^0$ for $j \in \{1, ..., |K|\} \setminus \{l\}$ and $I^4 = \bigcup_{j \in K} U_j^4$. It is clear that (x^{U^0}, y^{I^0}) and (x^{U^4}, y^{I^4}) are both in F_e and hence in F. Consequently, $rx^{U^0} + qy^{U^0} = rx^{U^4} + qy^{U^4} = rx^{U^0} + qy^{U^0} + r_{ss'}^l + q_{ss'}$. By (4.34), we have $q_{ss'} = 0$, implying that $r_{ss'}^l$. And therefore,

$$r_{ss'}^l = 0 \qquad \text{for all } s, s' \in S_l. \tag{4.36}$$

Now, consider $l \in K$, $l \neq k$ and let t_i and t_{i+1} be two successive terminals of T_l and s a Steiner node of S_l. Let (U^5, I^5) be given as follows $U_l^5 = (U_l^0 \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U_j^5 = U_j^0$ for $j \in \{1, ..., |K|\} \setminus \{l\}$ and $I^5 = \bigcup_{j \in K} U_j^5$. Obviously (x^{U^0}, y^{I^0}) and (x^{U^5}, y^{I^5}) are in F_e and then in F. This means that $rx^{U^0} + qy^{I^0} = rx^{U^5} + qy^{I^5} = rx^{U^0} + qy^{I^0} - r_{t_it_{i+1}}^l + r_{t_is}^l + r_{st_{i+1}}^l - q_{t_it_{i+1}} + q_{t_is} + q_{st_{i+1}}$. By (4.34), $q_{t_is} = q_{st_{i+1}} = 0$. Moreover, as $l \neq k$, by (4.35), $q_{t_it_{i+1}} = 0$. This yields $r_{t_it_{i+1}}^l = r_{t_is}^l + r_{st_{i+1}}^l$. As demand l and nodes s, t_i, t_{i+1} are arbitrary, we obtain that

$$r_{t_it_{i+1}}^l = r_{t_is}^l + r_{st_{i+1}}^l \qquad \text{for all}\, t_i, t_{i+1} \in T_l,\ s \in S_l, l \in K \setminus \{k\}. \tag{4.37}$$

In a similar way, we also obtain that

$$r_{t_it_{i+1}}^k = r_{t_is}^k + r_{st_{i+1}}^k \quad \text{for all}\, t_i, t_{i+1} \in T_k,\ s \in S_k,\ t_it_{i+1} \neq e. \tag{4.38}$$

Now, we will prove that the previous relation remains valid even when $t_it_{i+1} = e$. Consider a Steiner node s of S_k and let (U^6, I^6) be the solution defined by $U_k^6 = (U_k^0 \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U_j^6 = U_j^0$ for $j \in \{1, ..., |K|\} \setminus \{k\}$ and $I^6 = (\bigcup_{j \in K} U_j^6) \cup \{e\}$. Clearly (x^{U^0}, y^{I^0}) and (x^{U^6}, y^{I^6}) are in F_e and thus in F. This implies that $rx^{U^0} + qy^{I^0} = rx^{U^6} + qy^{I^6} = rx^{U^0} + qy^{I^0} - r^k_{t_it_{i+1}} + r^k_{t_is} + r^k_{st_{i+1}} - q_{t_it_{i+1}} + q_{t_is} + q_{st_{i+1}} + q_{t_it_{i+1}}$ (recall that $e = t_it_{i+1}$). By equation (4.34), it follows that $q_{t_is} = q_{st_{i+1}} = 0$, implying that $r^k_{t_it_{i+1}} = r^k_{t_is} + r^k_{st_{i+1}}$. And hence,

$$r^k_{t_it_{i+1}} = r^k_{t_is} + r^k_{st_{i+1}} \quad \text{for all}\, s \in S_k,\ t_it_{i+1} = e. \tag{4.39}$$

From (4.38) and (4.39), we get

$$r^k_{t_it_{i+1}} = r^k_{t_is} + r^k_{st_{i+1}} \quad \text{for all}\, t_i, t_{i+1} \in T_k,\ s \in S_k. \tag{4.40}$$

In what follows, we will prove that all the edges linking a terminal to Steiner nodes of a demand l have the same coefficient in r^l.

First suppose that $l \neq k$ and consider two Steiner nodes s and s' in S_l. consider again the solution (U^5, I^5) given previously and denote by (U^7, I^7) the solution given as follows $U_l^7 = (U_l^5 \setminus \{st_i\}) \cup \{ss', s't_i\}$, $U_j^7 = U_j^5$ for each $j \in \{1, ..., K\} \setminus l$ and $I^7 = \bigcup_{j \in K} U_j^7$. It is clear (x^{U^7}, y^{I^7}) is in F_e and thus in F. The incidence vectors of (U^5, I^5) and (U^7, I^7) satisfy equation $rx + qy = \beta$. This, together with equations (4.34), allow to write $rx^{U^5} = rx^{U^7} = rx^{U^5} - r^l_{st_i} + r^l_{ss'} + r^l_{s't_i}$. This leads to $r^l_{st_i} = r^l_{ss'} + r^l_{s't_i}$. By (4.36), we obtain that $r^l_{st_{i+1}} = r^l_{s't_i}$

Similarly, we can show that, if $l = k$, $r^k_{st_i} = r^k_{s't_i}$.

Thus, we get

$$r^l_{st_i} = r^l_{s't_i} = \lambda_1^l(t_i) \quad \begin{array}{l} \text{for all}\, t_i \in T_l,\ s, s' \in S_l, l \in K, \\ \text{for some}\, \lambda_1^l(t_i) \in \mathbb{R}. \end{array} \tag{4.41}$$

Now, let $\rho = q_e$ and $\lambda^l = (\lambda_1^l, \lambda_2^l)$, $l \in K$ such that $\lambda_1^l = (\lambda_1^l(t_i), t_i \in T_k)$ where $\lambda_1^l(t_i)$ is as given by (4.41) and $\lambda_2^l = (\lambda_2^l(uv),\ u, v \in T_k, uv \notin U_l^0)$ such that $\lambda_2^l(uv) = r^l_{uv} - \lambda_1^l(u) - \lambda_1^l(v)$, $l \in K$.

The coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can then be expressed in terms of λ_1^l and λ_2^l as follows

$$r^l_{uv} = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_it_{i+1},\ t_i, t_{i+1} \in T_l, \\ \lambda_1^l(u) & \text{if } u \in T_l,\ v \in S_l, \\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_it_j,\ t_i, t_j \in T_l : j > i, \\ 0 & \text{if } uv = s_is_j,\ s_i, s_j \in S_l : j \neq i, \end{cases}$$

yielding $r^l = \lambda_1^l M_1^l + \lambda_2^l M_2^l$ for all $l \in K$ as desired. Consequently, we have $q = \rho b$ and $r = \rho a + \lambda M$.

Case 2. $e \in E \setminus I^0$. Here, we distinguish 3 subcases.

Case 2.1. e is an edge between a terminal and a Steiner node.
Suppose that $e = s_1t_1$ where $t_1 \in T_k$ and $s_1 \in S_k$ for some $k \in K$.

We prove first that all the coefficient q_f, $f \neq e$ are equal to 0.

First, we suppose that $f \notin I^0$. Consider demand $k \in K$ and let t_1 and t_2 be two successive terminals of T_k. Consider the solution (U^8, I^8) defined by $U_k^8 = (U_k^0 \setminus \{t_1t_2\}) \cup \{t_1s_1, s_1t_2\}$, $U_j^8 = U_j^0$ for $j \in \{1, ..., |K|\} \setminus \{k\}$ and $I^8 = \bigcup_{j\in K} U_j^8$. Suppose that $f \in E \setminus \{s_1t_1, s_1t_2\}$, and let (U^9, I^9) be the the solution given by $U^9 = U^8$ and $I^9 = I^8 \cup \{f\}$. As (x^{U^8}, y^{I^8}) and (x^{U^9}, y^{I^9}) are in F_e and hence in F, we have $rx^{U^8} + qy^{I^8} = rx^{U^9} + qy^{I^9} = rx^{U^8} + qy^{I^8} + q_f$, implying that $q_f = 0$.

Now, let $t_0 \in T_k$ be the terminal predecessor of t_1 in demand k. Let (U^{10}, I^{10}) be the solution given by $U_k^{10} = (U_k^0 \setminus \{t_1t_2\}) \cup \{t_0s_1, s_1t_1\}$, $U_j^{10} = U_j^0$ for $j \in \{1, ..., |K|\} \setminus \{k\}$ and $I^{10} = (\bigcup_{j\in K} U_j^{10})$. Consider also the solution $(U^{11}, I^{11}) = (U^{10}, I^{10} \cup \{s_1t_2\})$. As $(x^{U^{10}}, y^{I^{10}})$ and $(x^{U^{11}}, y^{I^{11}})$ are in F_e and hence in F, we have $rx^{U^{10}} + qy^{I^{10}} = rx^{U^{11}} + qy^{I^{11}} = rx^{U^{10}} + qy^{I^{10}} + q_{s_1t_2}$, which implies that $q_{s_1t_2} = 0$. Consequently, we have

$$q_f = 0 \qquad \text{for all}\, f \in E \setminus (I^0 \cup \{e\}). \tag{4.42}$$

Now, we suppose that $f \in I^0$. Suppose first that $f = t_1t_2$. Consider the solution (U^8, I^8) previously defined and let (U^{12}, I^{12}) be the solution given by $U^{12} = U^8$ and $I^{12} = I^8 \cup \{t_1t_2\}$. As (x^{U^8}, y^{I^8}) and $(x^{U^{12}}, y^{I^{12}})$ are in F_e and hence in F, we have $rx^{U^8} + qy^{I^8} = rx^{U^{12}} + qy^{I^{12}} = rx^{U^8} + qy^{I^8} + q_{t_1t_2}$ yielding $q_{t_1t_2} = 0$.

Now, consider a demand $l \in K$ and suppose that $f = t_it_{i+1} \neq t_1t_2$, where t_i and t_{i+1} are two successive terminals of T_l. Consider s a Steiner node of S_l and let (U^{13}, I^{13}) be the solution given by $U_k^{13} = (U_k^8 \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U_j^{13} = U_j^8$ for $j \in \{1, ..., |K|\} \setminus \{l\}$ and $I^{13} = \bigcup_{j\in K} U_j^{13}$. In addition, we define the solution (U^{14}, I^{14}) given by $U^{14} = U^{13}$ and $I^{14} = I^{13} \cup \{t_it_{i+1}\}$. It is obvious that $(x^{U^{13}}, y^{I^{13}})$ and $(x^{U^{14}}, y^{I^{14}})$ are in F_e and hence in F. As a consequence, $rx^{U^{13}} + qy^{I^{13}} = rx^{U^{14}} + qy^{I^{14}} = rx^{U^{13}} + qy^{I^{13}} + q_{t_it_{i+1}}$, implying that $q_{t_it_{i+1}} = 0$.

Consequently, we have

$$q_f = 0 \qquad \text{for all}\, f \in I^0. \tag{4.43}$$

In what follows, we will show that vector $r = \rho a + \lambda M$.

First we show that for every edge ss' between two Steiner nodes $s, s' \in S_l$, $r^l_{ss'} = 0$ where $l \in K$. Consider again the solution (U^0, I^0) and define the solution (U^{15}, I^{15}) given as follows $U^{15} = U^0$ and $I^{15} = I^0 \cup \{e\}$. In addition, let (U^{16}, I^{16}) be the solution defined by $U^{16}_l = U^0_l \cup \{ss'\}$, $U^{16}_j = U^0_j$ for all $j \in \{1, ..., K\} \setminus \{l\}$ and $I^{16} = (\bigcup_{j \in K} U^{16}_j) \cup \{e\}$. As $(x^{U^{15}}, y^{I^{15}})$ and $(x^{U^{16}}, y^{I^{16}})$ are in F_e and hence in F, we have $rx^{U^{15}} + qy^{I^{15}} = rx^{U^{16}} + qy^{I^{16}} = rx^{U^{15}} + qy^{I^{15}} + r^l_{ss'} + q_{ss'}$. This, together with (4.42), implies that $r^l_{ss'} = 0$. Since l and ss' are chosen arbitrarily, we obtain

$$r^l_{ss'} = 0 \qquad \text{for all } s, s' \in S_l, l \in K. \tag{4.44}$$

Consider now an edge $t_i t_{i+1}$ between two successive terminals t_i and t_{i+1} of T_l, where $l \in K$. Let s be a Steiner node of S_l. Consider the solution (U^{17}, I^{17}) given by $U^{17}_l = (U^0_l \setminus \{t_i t_{i+1}\}) \cup \{t_i s, s t_{i+1}\}$, $U^{17}_j = U^0_j$, for each $j \in \{1, ..., K\} \setminus l$ and let $I^{17} = (\bigcup_{j \in K} U^{17}_j) \cup \{e\}$. As $(x^{U^{15}}, y^{I^{15}})$ and $(x^{U^{17}}, y^{I^{17}})$ are in F_e and hence in F, they satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^{15}} = rx^{U^{17}} = rx^{U^{15}} - r^l_{t_i t_{i+1}} + r^l_{t_i s} + r^l_{s t_{i+1}}$, implying that $r^l_{t_i t_{i+1}} = r^l_{t_i s} + r^l_{s t_{i+1}}$. Therefore

$$r^l_{t_i t_{i+1}} = r^l_{t_i s} + r^l_{s t_{i+1}} \qquad \text{for all } t_i, t_{i+1} \in T_l, \text{ for all } s \in S_l, l \in K. \tag{4.45}$$

Now, we will prove that all the edges linking a terminal of T_l to Steiner nodes of S_l have the same coefficient r^l, where $l \in K$. Consider s and s' two Steiner nodes of S_l. Consider (U^{18}, I^{18}) obtained from (U^{17}, I^{17}) as follows $U^{18}_l = (U^{17}_l \setminus \{s t_{i+1}\}) \cup \{ss', s' t_{i+1}\}$, $U^{18}_j = U^{17}_j$ for each $j \in \{1, ..., K\} \setminus l$ and $I^{18} = (\bigcup_{j \in K} U^{18}_j) \cup \{e\}$. Since $(x^{U^{15}}, y^{I^{15}})$ and $(x^{U^{18}}, y^{I^{18}})$ are in F_e and hence in F, they satisfy equation $rx + qy = \beta$, this implies $rx^{U^{15}} = rx^{U^{18}} = rx^{U^{15}} - r^l_{s t_{i+1}} + r^l_{ss'} + r^l_{s' t_{i+1}}$, By (4.44), it follows that $r^l_{s t_{i+1}} = r^l_{s' t_{i+1}}$. Therefore,

$$r^l_{s t_i} = r^l_{s' t_i} = \lambda^l_1(t_i) \qquad \begin{array}{l} \text{for all } t_i \in T_l,\ s, s' \in S_l,\ l \in K, \\ \text{for some } \lambda^l_1(t_i) \in \mathbb{R}. \end{array} \tag{4.46}$$

Now, let $\rho = q_e$ and $\lambda^l = (\lambda^l_1, \lambda^l_2)$, $l \in K$ such that $\lambda^l_1 = (\lambda^l_1(t_i), t_i \in T_k)$ where $\lambda^l_1(t_i)$ is as given by (4.46) and $\lambda^l_2 = (\lambda^l_2(uv),\ u, v \in T_k, uv \notin U^0_l)$ such that $\lambda^l_2(uv) = r^l_{uv} - \lambda^l_1(u) - \lambda^l_1(v)$, $l \in K$.

The coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can then be expressed in terms of λ^l_1

and λ_2^l as follows

$$r_{uv}^l = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_it_{i+1},\ t_i, t_{i+1} \in T_l \\ \lambda_1^l(u) & \text{if } u \in T_l,\ v \in S_l \\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_it_j,\ t_i, t_j \in T_l : j > i \\ 0 & \text{if } uv = s_is_j,\ s_i, s_j \in S_l : j \neq i, \end{cases}$$

implying that $r^l = \lambda_1^l M_1^l + \lambda_2^l M_2^l \qquad$ for all $l \in K$.

We then obtain $q = \rho b$ and $r = \rho a + \lambda M$ and the result follows.

Case 2.2 e is an edge between two Steiner nodes of a demand $k \in K$.

Suppose that $e = s_1s_2$ where s_1 and s_2 are Steiner nodes of S_k.

First, we prove that all the coefficient q_f, $f \neq e$ are equal to 0. In a first step, we suppose that $f \notin (I^0 \cup \{e\})$. Consider the following solutions: (U^{19}, I^{19}) given by $U^{19} = U^0$ and $I^{19} = I^0 \cup \{e\}$ and (U^{20}, I^{20}) such that $U^{20} = U^{19}$ and $I^{20} = I^{19} \cup \{f\}$. As $(x^{U^{19}}, y^{I^{19}})$ and $(x^{U^{20}}, y^{I^{20}})$ are in F_e and hence in F, we have $rx^{U^{19}} + qy^{I^{19}} = rx^{U^{20}} + qy^{I^{20}} = rx^{U^{19}} + qy^{I^{19}} + q_f$, which yields $q_f = 0$. Since f is arbitrarily in $E \setminus (I^0 \cup \{e\})$, this implies

$$q_f = 0 \qquad \text{for all}\, f \in E \setminus (I^0 \cup \{e\}). \tag{4.47}$$

Now, consider a demand $l \in K$ and let t_i and t_{i+1} be two successive terminals of T_l. Consider a Steiner node s of S_l and let (U^{21}, I^{21}) be the solution obtained as follows $U_l^{21} = (U_l^0 \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U_j^{21} = U_j^0$, $j \in \{1, ..., |K|\} \setminus \{l\}$ and $I^{21} = (\bigcup_{j \in K} U_j^{21}) \cup \{e\}$. Let us also define the solution (U^{22}, I^{22}) such that $U^{22} = U^{21}$ and $I^{22} = I^{21} \cup \{t_it_{i+1}\}$. As $(x^{U^{21}}, y^{I^{21}})$ and $(x^{U^{22}}, y^{I^{22}})$ are in F_e and thus in F, we have $rx^{U^{21}} + qy^{I^{21}} = rx^{U^{22}} + qy^{I^{22}} = rx^{U^{21}} + qy^{I^{21}} + q_{t_it_{i+1}}$, which implies $q_{t_it_{i+1}} = 0$. As t_i and t_{i+1} are arbitrary in T_l and l is arbitrary in K, we obtain

$$q_f = 0 \qquad \forall f \in I^0. \tag{4.48}$$

Next, we will establish some relations linking the components of vector r.

Consider a demand $l \in K$, $l \neq k$ and let s and s' be two Steiner nodes of S_l. Consider the solution $(x^{U^{23}}, y^{I^{23}})$ given by $x^{U^{23}} = x^0$ and $y^{I^{23}} = y^0 \cup \{s_1s_2\}$. Let also $(x^{U^{24}}, y^{I^{24}})$ be the solution defined as follows $x^{U^{24}} = x^{U^{23}}$ and $y^{I^{24}} = y^{I^{24}} \cup \{ss'\}$. As $(x^{U^{23}}, y^{I^{23}})$ and $(x^{U^{24}}, y^{I^{24}})$ are in F_e and thus in F, we have $rx^{U^{23}} + qy^{I^{23}} = rx^{U^{24}} + qy^{I^{24}} =$

$rx^{U^{23}} + qy^{I^{23}} + r^l_{ss'} + q_{ss'}$. This, together with (4.47), implies that $r^l_{ss'} = 0$. Since demand l and nodes s and s' are arbitrarily, we then have

$$r^l_{ss'} = 0 \qquad \text{for all } s, s' \in S_l, l \in K \setminus \{k\}. \tag{4.49}$$

Suppose now that $l = k$ and that s and s' are two Steiner nodes of S_k such that $ss' \neq s_1s_2$. In a similar way, we can show that $r^k_{ss'} = 0$. As s and s' are arbitrary in S_k, we have

$$r^k_{ss'} = 0 \qquad \text{for all } s, s' \in S_k, ss' \neq s_1s_2. \tag{4.50}$$

Now, consider a demand $l \in K$ and let t_i and t_{i+1} be two successive terminals of T_l. Consider the solution $(x^{U^{25}}, y^{I^{25}})$ given by $U^{25}_l = (U^0_l \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U^{25}_j = U^0_j$, for each $j \in \{1, ..., K\} \setminus l$ and let $I^{25} = (\bigcup_{j \in K} U^{25}_j) \cup \{e\}$. As $(x^{U^{23}}, y^{I^{23}})$ and $(x^{U^{25}}, y^{I^{25}})$ are in F_e and hence in F, they satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^{23}} = rx^{U^{25}} = rx^{U^{23}} - r^l_{t_it_{i+1}} + r^l_{t_is} + r^l_{st_{i+1}}$, implying that $r^l_{t_it_{i+1}} = r^l_{t_is} + r^l_{st_{i+1}}$. As demand l is arbitray and nodes s, t_i, t_{i+1} are arbitrary as well, we have

$$r^l_{t_it_{i+1}} = r^l_{t_is} + r^l_{st_{i+1}} \qquad \text{for all } t_i, t_{i+1} \in T_l, \text{ for all } s \in S_l, l \in K. \tag{4.51}$$

Now, we will prove that all the edges linking a terminal of T_l to Steiner nodes of S_l have the same coefficient r^l, where $l \in K$. Consider s and s' two Steiner nodes of S_l. Consider (U^{26}, I^{26}) obtained from (U^{25}, I^{25}) as follows $U^{26}_l = (U^{25}_l \setminus \{st_{i+1}\}) \cup \{ss', s't_{i+1}\}$, $U^{26}_j = U^{25}_j$ for each $j \in \{1, ..., K\} \setminus l$ and $I^{26} = (\bigcup_{j \in K} U^{26}_j) \cup \{e\}$. Since $(x^{U^{23}}, y^{I^{23}})$ and $(x^{U^{26}}, y^{I^{26}})$ are in F_e and hence in F, they satisfy equation $rx + qy = \beta$, this implies $rx^{U^{23}} = rx^{U^{26}} = rx^{U^{26}} - r^l_{st_{i+1}} + r^l_{ss'} + r^l_{s't_{i+1}}$, By (4.50), it follows that $r^l_{st_{i+1}} = r^l_{s't_{i+1}}$. As a consequence,

$$r^l_{st_i} = r^l_{s't_i} = \lambda^l_1(t_i) \qquad \begin{array}{l} \text{for all } t_i \in T_l,\ s, s' \in S_l,\ l \in K, \\ \text{for some } \lambda^l_1(t_i) \in \mathbb{R}. \end{array} \tag{4.52}$$

Now, let $\rho = q_e$ and $\lambda^l = (\lambda^l_1, \lambda^l_2)$, $l \in K$ such that $\lambda^l_1 = (\lambda^l_1(t_i), t_i \in T_k)$ where $\lambda^l_1(t_i)$ is as given by (4.52) and $\lambda^l_2 = (\lambda^l_2(uv),\ u, v \in T_k, uv \notin U^0_l)$ such that $\lambda^l_2(uv) = r^l_{uv} - \lambda^l_1(u) - \lambda^l_1(v)$, $l \in K$.

The coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can then be expressed in terms of λ^l_1 and λ^l_2 as follows

$$r^l_{uv} = \begin{cases} \lambda^l_1(u) + \lambda^l_1(v) & \text{if } uv = t_it_{i+1},\ t_i, t_{i+1} \in T_l \\ \lambda^l_1(u) & \text{if } u \in T_l,\ v \in S_l \\ \lambda^l_2(uv) + \lambda^l_1(u) + \lambda^l_1(v) & \text{if } uv = t_it_j,\ t_i, t_j \in T_l : j > i \\ 0 & \text{if } uv = s_is_j,\ s_i, s_j \in S_l : j \neq i. \end{cases}$$

This implies that

$$r^l = \lambda_1^l M_1^l + \lambda_2^l M_2^l \qquad \text{for all } l \in K.$$

As a consequence, $q = \rho b$ and $r = \rho a + \lambda M$ and the result follows.

Case 2.3. e is an edge between two non-successive terminals of a demand $k \in K$. Suppose that $e = t_m t_n$ where t_m and t_n are two non-successive terminals of T_k.

First, we prove that all the coefficient q_f, $f \neq e$ are equal to 0. First, assume that $f \notin (I^0 \cup \{e\})$. Consider the following solutions: (U^{27}, I^{27}) given by $U^{27} = U^0$ and $I^{27} = I^0 \cup \{e\}$ and (U^{28}, I^{28}) such that $U^{28} = U^{17}$ and $I^{28} = I^{17} \cup \{f\}$. As $(x^{U^{27}}, y^{I^{27}})$ and $(x^{U^{20}}, y^{I^{20}})$ are in F_e and hence in F, we have $rx^{U^{27}} + qy^{I^{27}} = rx^{U^{28}} + qy^{I^{28}} = rx^{U^{27}} + qy^{I^{27}} + q_f$, which yields $q_f = 0$. Since f is arbitrarily in $E \setminus (I^0 \cup \{e\})$, this implies

$$q_f = 0 \qquad \text{for all } f \in E \setminus (I^0 \cup \{e\}). \tag{4.53}$$

Now, consider a demand $l \in K$ and let t_i and t_{i+1} be two successive terminals of T_l. Consider a Steiner node s of S_l and let (U^{29}, I^{29}) be the solution obtained as follows $U_l^{29} = (U_l^0 \setminus \{t_i t_{i+1}\}) \cup \{t_i s, s t_{i+1}\}$, $U_j^{29} = U_j^0$, $j \in \{1, ..., |K|\} \setminus \{l\}$ and $I^{29} = (\bigcup_{j \in K} U_j^{29}) \cup \{e\}$. Let us also define the solution (U^{30}, I^{30}) such that $U^{30} = U^{29}$ and $I^{30} = I^{29} \cup \{t_i t_{i+1}\}$. As $(x^{U^{29}}, y^{I^{29}})$ and $(x^{U^{30}}, y^{I^{30}})$ are in F_e and thus in F, we have $rx^{U^{29}} + qy^{I^{29}} = rx^{U^{30}} + qy^{I^{30}} = rx^{U^{29}} + qy^{I^{29}} + q_{t_i t_{i+1}}$, which implies $q_{t_i t_{i+1}} = 0$. As t_i and t_{i+1} are arbitrary in T_l and l is arbitrary in K, we obtain

$$q_f = 0 \qquad \forall f \in I^0. \tag{4.54}$$

Next, we will establish some relations linking the components of vector r.

Consider a demand $l \in K$ and let s and s' be two Steiner nodes of S_l. Consider the solution $(x^{U^{31}}, y^{I^{31}})$ given by $x^{U^{31}} = x^0$ and $y^{I^{31}} = y^0 \cup \{e\}$. Let also $(x^{U^{32}}, y^{I^{32}})$ be the solution defined as follows $x^{U^{32}} = x^{U^{31}}$ and $y^{I^{32}} = y^{I^{31}} \cup \{ss'\}$. As $(x^{U^{31}}, y^{I^{31}})$ and $(x^{U^{32}}, y^{I^{32}})$ are in F_e and thus in F, we have $rx^{U^{31}} + qy^{I^{31}} = rx^{U^{32}} + qy^{I^{32}} = rx^{U^{31}} + qy^{I^{31}} + r_{ss'}^l + q_{ss'}$. This, together with (4.53), implies that $r_{ss'}^l = 0$. Since demand l and nodes s and s' are arbitrarily, we then have

$$r_{ss'}^l = 0 \qquad \text{for all } s, s' \in S_l, l \in K. \tag{4.55}$$

Consider now t_i and t_{i+1} two successive terminals of T_l. Consider the solution $(x^{U^{33}}, y^{I^{33}})$ given by $U_l^{33} = (U_l^0 \setminus \{t_i t_{i+1}\}) \cup \{t_i s, s t_{i+1}\}$, $U_j^{33} = U_j^0$, for each $j \in$

$\{1, ..., K\} \setminus l$ and let $I^{33} = (\bigcup_{j\in K} U_j^{25}) \cup \{e\}$. As $(x^{U^{31}}, y^{I^{31}})$ and $(x^{U^{33}}, y^{I^{33}})$ are in F_e and hence in F, they satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^{31}} = rx^{U^{33}} = rx^{U^{31}} - r^l_{t_it_{i+1}} + r^l_{t_is} + r^l_{st_{i+1}}$, yielding $r^l_{t_it_{i+1}} = r^l_{t_is} + r^l_{st_{i+1}}$. As demand l is arbitray and nodes s, t_i, t_{i+1} are arbitrary, we have

$$r^l_{t_it_{i+1}} = r^l_{t_is} + r^l_{st_{i+1}} \qquad \text{for all } t_i, t_{i+1} \in T_l, \text{ for all } s \in S_l, l \in K. \tag{4.56}$$

Now, we will prove that all the edges linking a terminal of T_l to Steiner nodes of S_l have the same coefficient r^l, where $l \in K$. Consider s and s' two Steiner nodes of S_l. Consider (U^{34}, I^{34}) obtained from (U^{33}, I^{33}) as follows $U_l^{34} = (U_l^{33} \setminus \{st_{i+1}\}) \cup \{ss', s't_{i+1}\}$, $U_j^{34} = U_j^{33}$ for each $j \in \{1, ..., K\} \setminus l$ and $I^{34} = (\bigcup_{j\in K} U_j^{34}) \cup \{e\}$. Since $(x^{U^{31}}, y^{I^{31}})$ and $(x^{U^{34}}, y^{I^{34}})$ are in F_e and hence in F, they satisfy equation $rx+qy = \beta$, this implies $rx^{U^{31}} = rx^{U^{34}} = rx^{U^{31}} - r^l_{st_{i+1}} + r^l_{ss'} + r^l_{s't_{i+1}}$, By (4.55), it follows that $r^l_{st_{i+1}} = r^l_{s't_{i+1}}$. As a consequence,

$$r^l_{st_i} = r^l_{s't_i} = \lambda_1^l(t_i) \qquad \begin{array}{l}\text{for all}\, t_i \in T_l,\ s, s' \in S_l,\ l \in K, \\ \text{for some}\, \lambda_1^l(t_i) \in \mathbb{R}.\end{array} \tag{4.57}$$

Now, let $\rho = q_e$ and $\lambda^l = (\lambda_1^l, \lambda_2^l)$, $l \in K$ such that $\lambda_1^l = (\lambda_1^l(t_i), t_i \in T_k)$ where $\lambda_1^l(t_i)$ is as given by (4.57) and $\lambda_2^l = (\lambda_2^l(uv),\ u, v \in T_k, uv \notin U_l^0)$ such that $\lambda_2^l(uv) = r^l_{uv} - \lambda_1^l(u) - \lambda_1^l(v)$, $l \in K$.

The coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can then be expressed in terms of λ_1^l and λ_2^l as follows

$$r^l_{uv} = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_it_{i+1},\ t_i, t_{i+1} \in T_l \\ \lambda_1^l(u) & \text{if } u \in T_l,\ v \in S_l \\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_it_j,\ t_i, t_j \in T_l : j > i \\ 0 & \text{if } uv = s_is_j,\ s_i, s_j \in S_l : j \neq i. \end{cases}$$

This yields $r^l = \lambda_1^l M_1^l + \lambda_2^l M_2^l$ for all $l \in K$. Thus, $q = \rho b$ and $r = \rho a + \lambda M$. □

4.2.2.2 Section cut inequalities

In this section, we study the facial structure of the section cut inequalities (4.1). Consider a demand $k \in K$ and a section $q_j^k = (t_j, t_{j+1}) \in \mathcal{T}_q$. Consider W a subset of nodes of $V^{k,j}$ such that $t_j \in W$ and $t_{j+1} \in \overline{W}$.

In the following theorem we give necessary and sufficient conditions under which inequalities (4.1) define facets of MSOND(G, K, T).

Theorem 4.14 *Inequality (4.1) defines a facet of MSOND(G,K,T) if and only if $W \cap S_k \neq \emptyset \neq \overline{W} \cap S_k$.*

Proof Let $F_W^{k,j}$ be the face induced by the section cut inequalities (4.1) corresponding to k, j and W, that is

$$F_W^{k,j} = \{(x,y) \in \text{MSOND}(G,K,T) : \sum_{e \in \delta_{G^{k,j}}(W)} x_e^k = 1\}.$$

Neccessity.

Assume for instance that $W \cap S_k = \emptyset$. Thus, W is reduced to a single node, namely the terminal t_j. Consider a solution (U, I) of MSOND with $U = (U_1, U_2, ..., U_{|K|})$ the edge sets corresponding to the demands $1, ..., |K|$. Let $U_{k,j}$ be the restriction of U_k on $G^{k,j}$. By constraints (4.6), U_k must have two edges incident to t_j. Hence, $U_{k,j}$ must have exactly one edge incident to t_j. As $W \cap S_k = \emptyset$, the cut $\delta_{G^{k,j}}(W)$ is reduced to that edge. And therefore, the incidence vector of (U, I), (x^U, y^I) belong to $F_W^{k,j}$. But, this implies that the face induced by inequality (4.1) is equal to MSOND(G,K,T), and hence it can be facet defining.

The case where $\overline{W} \cap S_k = \emptyset$ is similar.

Sufficiency.

Throughout the proof, we will suppose that W and $\overline{W}$ contain each at least one Steiner node of S_k.

Denote inequality (4.1) corresponding to k, j and W by $ax + by \leq \alpha$ and let $F_W^{k,j} = \{(x,y) \in \text{MSOND}(G,K,T) : ax + by = \alpha\}$. Let $rx + qy \leq \beta$ be a valid inequality defining a facet F of MSOND(G,K,T) such that $F_W^{k,j} \subseteq F$. In the following, we prove that there exist $\rho \in \mathbb{R}$ and $\lambda = (\lambda^l, l \in K)$, $\lambda^l \in \mathbb{R}^{|T_l|+p_l}$ for $l \in K$, such that $q = \rho b$ and $r = \rho a + \lambda M$ (where $r = (r^1, r^2, ..., r^{|K|})$ with $r^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ and M is the matrix of equations defined above). Note here that $a = (a^1, a^2, ..., a^{|K|})$ is such that $a^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ with $a^i = 0$ for $i \in \{1, ..., |K|\} \setminus \{k\}$, $a_e^k \neq 0$ for every $e \in \delta_{G^{k,j}}(W)$ and $a_{e'}^k = 0$ for every $e' \in E \setminus \delta_{G^{k,j}}(W)$. Remark also that $b = 0$.

First, we prove that $q = 0$.
Consider the solution (U^0, I^0) and let $e \in E \setminus I^0$ be an arbitrary edge. Let (U^1, I^1) be the solution given by $I^1 = I^0 \cup \{e\}$ and $U^1 = U^0$. Both (U^0, I^0) and (U^1, I^1) induce vectors that are in $F_W^{k,j}$. This implies that their incidence vectors satisfy equation

$rx + qy = \beta$. Consequently, $rx^{U^0} + qy^{I^0} = rx^{U^1} + qy^{I^1} = qx^{U^0} + qy^{I^0} + q_e$, implying that $q_e = 0$. Since e is arbitrarily chosen in $E \setminus I^0$, we have

$$q_e = 0 \qquad \text{for all } e \in E \setminus I^0. \tag{4.58}$$

Now, consider the solution (U^2, I^2) obtained as follows. Consider a demand $l \in K$ and let $e = t_i t_{i+1}$ be an edge between two successive terminals t_i and t_{i+1} of T_l. Let $U_l^2 = (U_l^0 \setminus \{e\}) \cup \{t_i s, st_{i+1}\}$ where $s \in S_l$ is a Steiner node of demand l. In addition, let $U_j^2 = U_j^0$, $j = 1, ..., |K|, j \neq l$, and $I^2 = \bigcup_{j \in K} U_j^2$. Now, let us define the solution (U^3, I^3) given by $U^3 = U^2$ and $I^3 = I^2 \cup \{e\}$. Both are feasible solutions for the MSOND problem and hence satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^2} + qy^{I^2} = rx^{U^3} + qy^{I^3} = rx^{U^2} + qy^{I^2} + q_e$, which implies that $q_e = 0$. Recall that I^0 is the set of edges between the consecutive terminals of all the demands. As e is arbitrary in I^0, this yields

$$q_e = 0 \qquad \text{for all } e \in I^0. \tag{4.59}$$

By (4.58) and (4.59) we then have

$$q_e = 0 \qquad \text{for all } e \in E. \tag{4.60}$$

Next, we will establish some relations between the components of vector r.

First, we consider the edges between Steiner nodes.
Let $l \in K \setminus \{k\}$ and s and s' be two Steiner nodes of S_l. Consider the solution (U^0, I^0) and let (U^4, I^4) be the solution defined by $U_l^4 = U_l^0 \cup \{ss'\}$, $U_p^4 = U_p^0$ for all $p \in \{1, ..., K\} \setminus l$ and $I^4 = \bigcup_{p \in K} U_p^4$. Clearly, (x^{U^0}, y^{I^0}) and (x^{U^4}, y^{I^4}) are both in $F_W^{k,j}$ and thus in F. This implies that (x^{U^0}, y^{I^0}) and (x^{U^4}, y^{I^4}) satisfy $rx + qy = \beta$. Since $q = 0$, we have $rx^{U^0} = rx^{U^4} = r^k x^{U^0} + r_{ss'}^l$, which yields $r_{ss'}^l = 0$. As demand l and nodes s and s' are all arbitrary, we have

$$r_{ss'}^l = 0 \qquad \text{for all}\, s, s' \in S_l,\ l \in K \setminus \{k\}. \tag{4.61}$$

Now, suppose that s and s' are Steiner nodes of S_k such that $ss' \notin \delta_{G^{k,j}}(W)$. That is s and s' are both either in W or $\overline{W}$. Let (U^5, I^5) be the solution defined as follows, $U_k^5 = U_k^0 \cup \{ss'\}$, $U_p^5 = U_p^0$ for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^5 = \bigcup_{p \in K} U_p^5$. Obviously (x^{U^5}, y^{I^5}) is in $F_W^{k,j}$ and hence in F. As a consequence, the incidence vectors of (U^0, I^0) and (U^5, I^5) satisfy equation $rx + qy = \beta$. Since $q = 0$, we have $rx^{U^0} = rx^{U^5} = rx^{U^0} + r_{ss'}^k$ implying that $r_{ss'}^k = 0$. As s and s' are arbitrary, we have

$$r_{ss'}^k = 0 \qquad \text{for all}\, s, s' \in S_k,\ ss' \notin \delta_{G^{k,j}}(W). \tag{4.62}$$

The case where $ss' \in \delta_{G^{k,j}}(W)$ will be studied at the end of the proof.

Now, we will consider the edges between terminals.
Consider a demand $l \in K \setminus \{k\}$ and let t_i and t_{i+1} be two terminals of T_l. Let s be a Steiner node of S_l. Consider the solution (U^6, I^6) given by $U_l^6 = (U_l^0 \setminus \{t_i t_{i+1}\}) \cup \{t_i s, st_{i+1}\}$, $U_p^6 = U_p^0$, for each $p \in \{1, ..., K\} \setminus l$ and $I^6 = \bigcup_{p \in K} U_p^6$. As (x^{U^6}, y^{I^6}) is in $F_W^{k,j}$ and hence in F, it follows that $rx^{U^0} = rx^{U^6} = rx^{U^0} - r^l_{t_i t_{i+1}} + r^l_{t_i s} + r^l_{st_{i+1}}$, implying that $r^l_{t_i t_{i+1}} = r^l_{t_i s} + r^l_{st_{i+1}}$. Demand l as well as nodes s, t_i and t_{i+1} are all arbitrary, therefore,

$$r^l_{t_i t_{i+1}} = r^l_{t_i s} + r^l_{st_{i+1}} \qquad \text{for all } t_i, t_{i+1} \in T_l,\ s \in S_l,\ l \in K \setminus \{k\}. \tag{4.63}$$

Now, suppose that t_i and t_{i+1} are terminals of T_k such that $t_i t_{i+1} \neq t_j t_{j+1}$. As we did for the previous case, we can also prove that

$$r^k_{t_i t_{i+1}} = r^k_{t_i s} + r^k_{st_{i+1}} \qquad \text{for all } t_i, t_{i+1} \in T_k,\ s \in S_k,\ t_i t_{i+1} \neq t_j t_{j+1}. \tag{4.64}$$

Consider now two Steiner nodes s and s' of S_k such that $s \in W$ and $s' \in \overline{W}$ (recall that $W \cap S_k \neq \emptyset$ and $\overline{W} \cap S_k \neq \emptyset$). Let (U^7, I^7) be the solution defined as follows: $U_k^7 = (U_k^0 \setminus \{t_j t_{j+1}\}) \cup \{t_j s, ss', s't_{j+1}\}$, $U_p^7 = U_p^0$, for each $p \in \{1, ..., K\} \setminus k$ and $I^7 = \bigcup_{p \in K} U_p^7$. As (x^{U^0}, y^{I^0}) and (x^{U^7}, y^{I^7}) are both in $F_W^{k,j}$ and thus in F, we have $rx^{U^0} = rx^{U^7} = rx^{U^0} - r^k_{t_j t_{j+1}} + r^k_{t_j s} + r^k_{ss'} + r^k_{s't_{j+1}}$, yielding to $r^k_{t_j t_{j+1}} = r^k_{t_j s} + r^k_{ss'} + r^k_{s't_{j+1}}$. Hence, we have

$$r^k_{t_j t_{j+1}} = r^k_{t_j s} + r^k_{ss'} + r^k_{s't_{j+1}} \qquad \text{for all } s, s' \in S_k,\ s \in W \text{ and } s' \in \overline{W}. \tag{4.65}$$

In what follows, we will look at the coefficients of edges between terminals and Steiner nodes.
To this end, consider first a demand $l \in K \setminus \{k\}$ and let (U^8, I^8) be the solution obtained from (U^6, I^6) as follows, $U_l^8 = (U_l^6 \setminus \{st_{i+1}\}) \cup \{ss', s't_{i+1}\}$, $U_p^8 = U_p^6$ for each $p \in \{1, ..., K\} \setminus l$ and $I^8 = \bigcup_{j \in K} U_j^8$, where s and s' are Steiner nodes of S_l. Since (x^{U^6}, y^{I^6}) and (x^{U^8}, y^{I^8}) are both in $F_W^{k,j}$ and thus in F, this implies $rx^{U^6} = rx^{U^8} = rx^{U^6} - r^l_{st_{i+1}} + r^l_{ss'} + r^l_{s't^k_{i+1}}$. By (4.61) it follows that $r^l_{st_{i+1}} = r^l_{s't_{i+1}}$. As demand l and nodes s, s' and t_i are all arbitrary , we have

$$r^l_{st_i} = r^l_{s't_i} = \lambda_1^l(t_i) \qquad \begin{array}{l}\text{for all}\, t_i \in T_l,\ s, s' \in S_l, l \in K \setminus \{k\},\\ \text{for some } \lambda_1^l(t_i) \in \mathbb{R}.\end{array} \tag{4.66}$$

If $l = k$, along the same way we obtain that

$$r^k_{st_i} = r^k_{s't_i} = \lambda_1^k(t_i) \qquad \begin{array}{l}\text{for all}\, t_i \in T_k \setminus \{t_j, t_{j+1}\},\ s, s' \in S_k,\\ \text{for some } \lambda_1^k(t_i) \in \mathbb{R}.\end{array} \tag{4.67}$$

Suppose now that s and s' are Steiner nodes of S_k such that $ss' \notin \delta_{G^{k,j}}(W)$. Consider the solution (U^9, I^9) defined as follows, $U_k^9 = (U_k^0 \setminus \{t_j t_{j+1}\}) \cup \{t_j s, s t_{j+1}\}$, $U_p^9 = U_p^0$, for each $p \in \{1, ..., K\} \setminus k$ and $I^9 = \bigcup_{p \in K} U_p^9$. We also define (U^{10}, I^{10}) by: $U_k^{10} = (U_k^9 \setminus \{t_j s\}) \cup \{t_j s', ss'\}$, $U_p^{10} = U_p^9$, for each $p \in \{1, ..., K\} \setminus k$ and $I^{10} = \bigcup_{p \in K} U_p^{10}$. Solutions (x^{U^9}, y^{I^9}) and $(x^{U^{10}}, y^{I^{10}})$ are both in $F_W^{k,j}$ and thus in F. As a consequence, we have $rx^{U^9} = rx^{U^{10}} = rx^{U^{10}} - r_{t_j s}^k + r_{t_j s'}^k + r_{ss'}^k$, yielding $r_{t_j s}^k = r_{t_j s'}^k + r_{ss'}^k$. By (4.62), it follows that $r_{t_j s}^k = r_{t_j s'}^k$. Similarly, we can show that $r_{t_{j+1} s}^k = r_{t_{j+1} s'}^k$. And since s and s' are arbitrary in S_k, we have

$$r_{t_i s}^k = r_{t_i s'}^k = \lambda_1^k(t_i) \qquad \begin{array}{l} \text{for all } s, s' \in S_k,\ ss' \notin \delta_{G^{k,j}}(W),\ t_i \in \{t_j, t_{j+1}\}, \\ \text{for some } \lambda_1^k(t_i) \in \mathbb{R}. \end{array} \tag{4.68}$$

Now, suppose that $ss' \in \delta_{G^{k,j}}(W)$ with $s' \in W$ and $s \in \overline{W}$. Along the same line, we can prove that $r_{t_j s}^k = r_{t_j s'}^k + r_{ss'}^k$ and $r_{t_{j+1} s}^k = r_{t_{j+1} s'}^k + r_{ss'}^k$. As s and s' are arbitrary, we have the following

$$r_{t_i s}^k = r_{t_i s'}^k + r_{ss'}^k \qquad \begin{array}{l} \text{for all } s, s' \in S_k,\ s' \in W,\ s \in \overline{W}, \\ t_i \in \{t_j, t_{j+1}\}. \end{array} \tag{4.69}$$

Now, we will go back to the case that we left in the beginning of the proof, concerning edges between Steiner nodes of S_k that belong to $\delta_{G^{k,j}}(W)$.
Consider demand k and let s_1, s_2, s_3 and s_4 be Steiner nodes of S_k such that, s_1 and s_3 are in W, and s_2 and s_4 are in $\overline{W}$ (s_1 and s_3, s_2 and s_4 may be the same). By (4.69) and as $s_1 s_2 \in \delta_{G^{k,j}}(W)$ and $s_3 s_4 \in \delta_{G^{k,j}}(W)$, we have the following, $r_{t_{j+1} s_1}^k = r_{t_{j+1} s_2}^k + r_{s_1 s_2}^k$ and $r_{t_{j+1} s_3}^k = r_{t_{j+1} s_4}^k + r_{s_3 s_4}^k$. Moreover, by (4.68) and since $s_1 s_3 \notin \delta_{G^{k,j}}(W)$ and $s_2 s_4 \notin \delta_{G^{k,j}}(W)$, it follows that $r_{t_{j+1} s_1}^k = r_{t_{j+1} s_3}^k$ and $r_{t_{j+1} s_2}^k = r_{t_{j+1} s_4}^k$, yielding $r_{s_1 s_2}^k = r_{s_3 s_4}^k$. As s_1, s_2, s_3 and s_4 are all arbitrary in S_k, we then have

$$r_{s_1 s_2}^k = r_{s_3 s_4}^k = \rho \qquad \begin{array}{l} \text{for all } s_1, s_2, s_3, s_4 \in S_k, \\ s_1 s_2 \in \delta_{G^{k,j}}(W) \text{ and } s_3 s_4 \in \delta_{G^{k,j}}(W) \\ \text{for some } \rho \in \mathbb{R}. \end{array} \tag{4.70}$$

Now, let $\rho \in \mathbb{R}$ be as given by (4.70) and $\lambda^l = (\lambda_1^l, \lambda_2^l)$, $l \in K$ such that $\lambda_1^l = (\lambda_1^l(t_i), t_i \in T_k)$ where $\lambda_1^l(t_i)$ is as given by (4.66), (4.67), (4.68) and (4.69). $\lambda_2^l = (\lambda_2^l(uv),\ u, v \in T_k, uv \notin U_l^0)$ such that $\lambda_2^l(uv) = r_{uv}^l - \lambda_1^l(u) - \lambda_1^l(v)$, $l \in K$.

Overall, the coefficients r_{uv}^l for all $uv \in E$ and $l \in K$ can then be expressed in terms of ρ, λ_1^l and λ_2^l as follows

$$r_{uv}^l = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } l \neq k,\ uv = t_i t_{i+1},\ t_i, t_{i+1} \in T_l, \\ \lambda_1^k(u) + \lambda_1^k(v) & \text{if } l = k,\ uv = t_i t_{i+1},\ t_i, t_{i+1} \in T_k, uv \neq t_j t_{j+1}, \\ \rho + \lambda_1^k(u) + \lambda_1^k(v) & \text{if } l = k,\ uv = t_j t_{j+1}, \\ \lambda_1^l(u) & \text{if } l \neq k,\ u \in T_l,\ v \in S_l, \\ \lambda_1^k(u) & \text{if } l = k,\ u \in T_k \setminus \{t_j, t_{j+1}\},\ v \in S_k, \\ \rho + \lambda_1^k(u) & \text{if } l = k,\ u \in T_k \setminus \{t_j, t_{j+1}\},\ v \in S_k, uv \in \delta_{G^{k,j}}(W), \\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_i t_{i'},\ t_i, t_{i'} \in T_l,\ i' > i, \\ 0 & \text{if } l \neq k \text{ and } uv = s_i s_{i'},\ s_i, s_{i'} \in S_l,\ i' \neq i, \\ 0 & \text{if } l = k \text{ and } uv = s_i s_{i'},\ s_i, s_{i'} \in S_k,\ s_i s_{i'} \notin \delta_{G^{k,j}}(W), \\ \rho & \text{if } l = k \text{ and } uv = s_i s_{i'},\ s_i, s_{i'} \in S_k,\ s_i s_{i'} \in \delta_{G^{k,j}}(W). \end{cases}$$

Clearly, $r^l = \rho a^l + \lambda_1^l M_1^l + \lambda_2^l M_2^l$, for all $l \in K$. As a consequence, $r = \rho a + \lambda M$ and the proof is complete. $\square$

4.2.2.3 Disjunction inequalities

In this section, we examine the facial structure of the disjunction inequalities (4.2). The following theorem gives a necessary and sufficient condition under which inequalities (4.2) define facets for the MSOND(G, K, T) polytope.

Theorem 4.15 *Inequality* $\sum\limits_{e \in \delta(w)} x_e^k \leq 2$ *defines a facet for MSOND(G, K, T) if and only if w is not a terminal of T_k.*

Proof Let F_w^k define the face induced by the disjunction inequality $\sum\limits_{e \in \delta(w)} x_e^k \leq 2$, that is

$$F_w^k = \{(x, y) \in \text{MSOND}(G, K, T) : \sum_{e \in \delta_G(w)} x_e^k = 2\}.$$

The necessity follows from Proposition 4.6.

Suppose now that $w \in S_k$, that is to say w is a Steiner node of demand k. This demand has $|T_k|$ terminals that will be denoted by $t_1, t_2, t_3, ..., t_{|T_k|}$. We prove that inequalities (4.2) are facet defining by exhibiting dim(MSOND(G, K, T)) points in F_w^k that are affinely independent. These will be determined in two steps.

We first exhibit $p_1 = (|K|+1)|E| - \sum_{j\in K}\frac{|T_j|(|T_j|-1)}{2} - |T_k| + 1$ solutions of MSOND(G,K,T) that are in F_w^k. These can be obtained as follows. Consider, without loss of generality, the first section of demand k, that is $(t_1,t_2) \in \mathcal{T}_k$ (see Figure 4.2). Remark that Figure 4.2 shows the edges $e_1, e_2, \ldots, e_{|T_k|-1}, e_{|T_k|}$. These are the edges linking the different terminals of T_k to the node w. Now, consider all the possible solutions of MSOND(G,K,T) that are obtained by inserting the Steiner node w between t_1 and t_2. Note that these solutions all use edges e_1 and e_2. Note also that, since w always appears in these solutions, w can be considered as a terminal for demand k. Hence, the incidence vectors of these solutions are all in F_w^k and may be considered as the possible solutions of the problem with w added as terminal of T_k between t_1 and t_2. Note that all these solutions do not use any of the edges $e_3, e_4, \ldots, e_{|T_k|-1}, e_{|T_k|}$ since these edges are between non-successive terminals in the new problem. Hence, all the solutions built this way are also solutions of the polytope MSOND(G,K,T'), where $T' = (T \setminus T_k) \cup T_k^{'}$ with $T_k^{'} = T_k \cup \{w\}$, is the new set of terminals of demand k. By Theorem 4.11, there are $dim($MSOND$(G,K,T')) + 1$ solutions affinely independent in MSOND(G,K,T'). As all these solutions are in F_w^k and $|T_k'| = |T_k|+1$, we then obtain $(|K|+1)|E| - \sum_{j\in K}\frac{|T_j|(|T_j|-1)}{2} - |T_k| + 1$ solutions affinely independent in F_w^k.

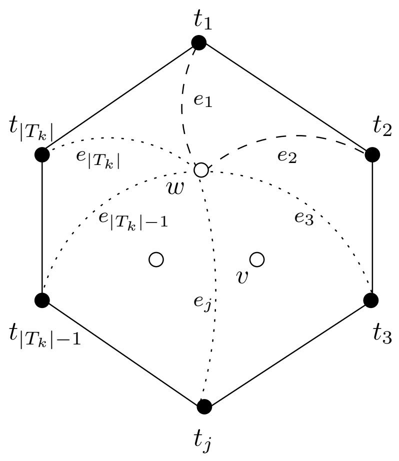


Figure 4.2:

Now, we will exhibit $p_2 = |T_k| - 1$ further solutions in F_w^k.
Consider a Steiner node $v \in S_k \setminus \{w\}$ and let (U^j, I^j), $j = 2, ..., |T_k|$, be the solution given by $U_k^j = (U_k^0 \setminus \{t_j t_{j+1}\}) \cup \{t_j w, wv, vt_{j+1}\}$, $U_p^j = U_p^0$ for each $p \in K \setminus \{k\}$, and $I^j = \bigcup_{p\in K} U_p^j$. Observe that solution (U^j, I^j) is nothing but (U^0, I^0) for which we delete edge $t_j t_{j+1}$ and insert the Steiner nodes w and v respectively, between t_j and t_{j+1}. Also note that exactly two edges related to demand k are incident to w in each of these solutions. Hence, the incidence vectors of these solutions all belong to F_w^k. Furthermore, it is clear that their incidence vectors are affinely independent.

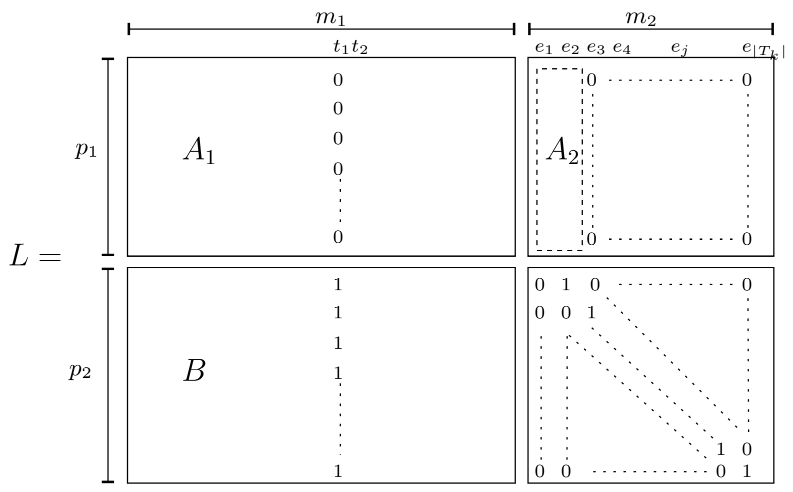


Figure 4.3:

Let L be the matrix whose rows are the incidence vectors of all the solutions obtained above (see Figure 4.3). Matrix L is organized as follows. The first m_1 columns of L correspond to all the variables $y_e, e \in E$ and $x_e^l, e \in E, l \in K$ except the variables $x_{e_1}^k, x_{e_2}^k, ..., x_{e_{|T_k|}}^k$. The last m_2 columns of L are associated with the variables $x_{e_1}^k, x_{e_2}^k, ..., x_{e_{|T_k|}}^k$. The first p_1 rows of L correspond to the solutions obtained in the first step of the proof. The ones built in the second step correspond to the last p_2 rows of L.

Observe that the first p_1 solutions use only edges among e_1 and e_2 and do not use the $m_2 - 2$ last edges of matrix L, that are edges between w and terminals $t_3, \dots, t_{|T_k|}$. Moreover, these solutions do not use the edge t_1t_2. The other p_2 solutions uses each time only one of the m_2 last edges that is different from e_1. Remark also that, in contrast with the first p_1 solutions, all the last p_2 solutions use the edge t_1t_2. As the first p_1 solutions are affinely independent, it can be easily seen that the p_1+p_2 solutions are affinely independent.

Thus, we obtain $p_1 + p_2 = (|K| + 1)|E| - \sum_{j\in K} \frac{|T_j|(|T_j|-1)}{2} = dim(\text{MSOND}(G, K, T))$ affinely independent solutions of F_w^k, which ends the proof. □

4.3 Valid inequalities and facets

In this section, we introduce several families of valid inequalities for the MSOND(G, K, T) polytope. We also study their facial aspect and give necessary conditions and sufficient conditions for these inequalities to define facets.

4.3.1 Steiner cut inequalities

The first family of valid inequalities is a straight consequence related to the connectivity requirements of the problem.

Consider the graph of Figure 4.4 which consists of four nodes, three terminals numbered $1, 2$ and 3 and a Steiner node, namely node 4. The instance consists of a demand between terminals 1 and 2 routed by the paths $L'_1 = (1, 2)$ and $L'_2 = (1, 3, 2)$. Figure 4.4 shows a fractional solution for this instance. Let $\bar{x}$ be the solution given by $\bar{x}_{e_i} = \frac{1}{2}$ for $i = 1, ..., 6$. Clearly, $\bar{x}$ satisfies all the constraints of the linear relaxation of (4.6).

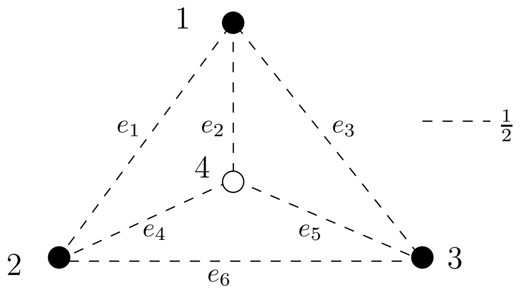


Figure 4.4: First fractional solution

However, $\bar{x}$ violates the inequality

$$x_{e_1} + x_{e_2} + x_{e_3} \geq 2,$$

which is valid for the MSOND(G, K, T) polytope.

In the following proposition, we prove that this inequality belongs to a more general class of valid inequalities for MSOND(G, K, T) that we call the *Steiner cut inequalities*.

Proposition 4.16 *Consider a demand $k \in K$ and let $W \subset V$ such that $W \cap T_k \neq \emptyset \neq \overline{W} \cap T_k$. Then*

$$\sum_{e \in \delta(W)} x_e^k \geq 2 \tag{4.71}$$

is valid for MSOND(G, K, T).

Proof Recall that the SC-MSOND problem considered for demand k consists in finding a cycle going in order through the terminals T_k. Consider, without loss of generality, two arbitrary terminals t_i and t_j of T_k. Since the cycle must go through t_i and t_j, this can be seen as 2 node-disjoint paths between t_i and t_j. By Menger's theorem, each t_it_j-cut must contain at least 2 edges, and the result follows. □

In what follows, we discuss the facial structure of inequalities (4.71). The following theorem gives necessary and sufficient conditions under which inequalities (4.71) define facets for the MSOND(G, K, T) polytope.

Theorem 4.17 *Inequality $\sum_{e\in\delta(W)} x_e^k \geq 2$ defines a facet of MSOND(G, K, T) if and only if the following conditions hold.*

1) W and $\overline{W}$ do not contain non-successive terminals of T_k,

2) If $|W \cap T_k| \geq 3$ (resp. $|\overline{W} \cap T_k| \geq 3$), then $S_k \subset W$ (resp. $S_k \subset \overline{W}$),

3) If $|W \cap T_k| = 2$ (resp. $|\overline{W} \cap T_k| = 2$), then $W \cap S_k \neq \emptyset$ (resp. $\overline{W} \cap S_k \neq \emptyset$).

Proof Let F_W^k be the face induced by inequality (4.71) corresponding to k and W, that is

$$F_W^k = \{(x, y) \in \text{MSOND}(G, K, T) : \sum_{e\in\delta(W)} x_e^k = 2\}.$$

Neccessity.

1) First, assume that condition 1) of the previous theorem is not satisfied. Suppose for instance that W contains non-successive terminals or non-successive sequences of terminals of T_k. Suppose for example that W contains two non-successive terminals of T_k, say t_i and t_j (see Figure 4.5). Consider a solution (U, I) of MSOND. By (4.1), t_i must be linked to t_{i-1} and t_{i+1} by two disjoint paths. Denote by $P_{i,1}$ and $P_{i,2}$ these paths. Similarly, there are two disjoint paths $P_{j,1}$ and $P_{j,2}$ linking terminal t_j to its predecessor and successor, respectively. As t_i and t_j are non-successive terminals, this implies that the paths $P_{i,1}, P_{i,2}, P_{j,1}$ and $P_{j,2}$ all intersect cut $\delta(W)$. Thus, $x^k(\delta(W)) \geq 4$ and therefore (x^U, y^I) does not belong to F_W^k. But this implies that $F_W^k = \emptyset$, and hence it can not define a facet of MSOND(G, K, T).

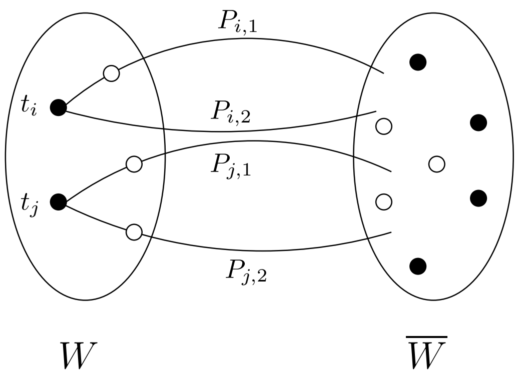


Figure 4.5:

2) Now, suppose that condition 1) is satisfied, however condition 2) is not satisfied. Suppose for instance that $|W \cap T_k| \geq 3$ and $\overline{W} \cap S_k \neq \emptyset$. Consider, without loss of generality, the case where $|W \cap T_k| = 3$. Denote the terminals of T_k that are in W by t_1, t_2 and t_3 (see Figure 4.6).

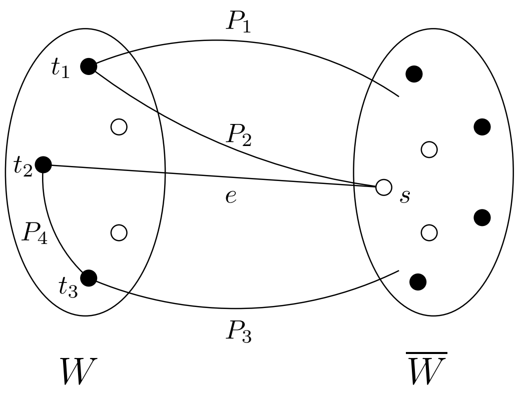


Figure 4.6:

Consider a solution (U, I) of MSOND, with $U = (U_1, U_2, ..., U_{|K|})$, such that $(x^U, y^I) \in F_W^k$. Let $e = t_2 s$ be the edge between the terminal $t_2 \in W$ and the Steiner node $s \in \overline{W}$ (see Figure 4.6). Suppose that $e \in U_k$. From constraints (4.1), it follows that U_k contains two paths linking each terminal of T_k to its predecessor and successor. In particular, there exist a path P_1 between t_1 and its successor in $\overline{W}$, a path P_3 between t_3 and its predecesor in $\overline{W}$ and a path P_4 between t_2 and t_3. Moreover, as $e \in U_k$, by (4.6), it follows that e must belong to the path joining t_1 and t_2. Therefore, there must exist a path between the Steiner node s and the terminal t_1. This path is denoted P_2. Notice that P_1, P_2, P_3 and $\{e\}$ all intersect cut $\delta(W)$ (see Figure 4.6). As a consequence, $x^k(\delta(W)) \geq 4$ and hence (x^U, y^I) does not belong to F_W^k. This implies that every solution of MSOND such that $(x^U, y^I) \in F_W^k$ satisfies $x_e^k = 0$. Hence $F_W^k \subset F_e^k$, where $F_e^k = \{(x,y) \in \text{MSOND}(G,K,T) : x_e^k = 0\}$. Here, e is not between non-successive terminals. By Theorem (4.12), it follows that F_e^k define a facet of MSOND(G,K,T). Moreover, inequality (4.71) cannot be obtained as a combination of $x_e^k \geq 0$ and the equations of MSOND(G,K,T), i.e. 4.7 and 4.8. Consequently, F_W^k cannot define a facet of MSOND(G,K,T).

3) Now, we will suppose that conditions 1) and 2) are satisfied. Assume however conditions 3) is not satisfied. Suppose, for example, that $|W \cap T_k| = 2$ but $W \cap S_k = \emptyset$.

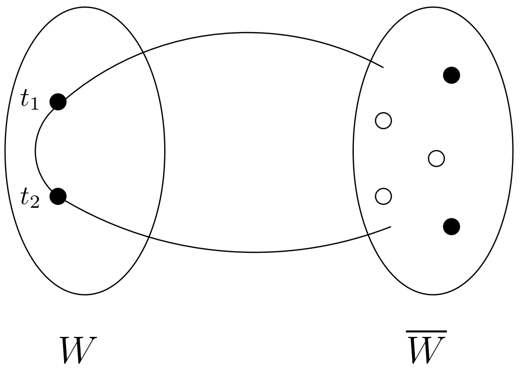


Figure 4.7:

Consider a solution (U, I) of MSOND with $U = (U_1, U_2, ..., U_{|K|})$ such that $t_1t_2 \notin U_k$. As $x^k(t_1) = 2$ and $x^k(t_2) = 2$ hold, it follows that $x^k(\delta(W)) = 4$, and hence $(x^U, y^I) \notin F_W^k$. But this implies that F_W^k is contained in the face induced by $x_{t_1t_2}^k \leq 1$. Hence F_W^k cannot define a facet of MSOND(G,K,T).

Sufficiency.

In the sequel, we suppose that both conditions 1), 2) and 3) are satisfied.

Denote inequality (4.71) corresponding to k and W by $ax + by \leq \alpha$. Let $rx + qy \leq \beta$ be a valid inequality defining a facet F of MSOND(G,K,T). In the following, we prove that there exist $\rho \in \mathbb{R}$ and $\lambda = (\lambda^l, l \in K)$, $\lambda^l \in \mathbb{R}^{|T_l|+p_l}$ for $l \in K$, such that $q = \rho b$ and $r = \rho a + \lambda M$ (where $r = (r^1, r^2, ..., r^{|K|})$ with $r^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ and M is the matrix of equations defined above). Notice here that $a = (a^1, a^2, ..., a^{|K|})$ such that $a^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ with $a^i = 0$ for $i \in \{1, ..., |K|\} \setminus \{k\}$, $a_e^k \neq 0$ for every $e \in \delta(W)$ and $a_{e'}^k = 0$ for every $e' \in E \setminus \delta(W)$. Remark also that $b = 0$.

First, we prove that $q = 0$.

Consider the solution (U^0, I^0) and let $e \in E \setminus I^0$ be an arbitrary edge. Consider also the solution (U^1, I^1) such that $I^1 = I^0 \cup \{e\}$ and $U^1 = U^0$. Both induce vectors that are in F_W^k and thus in F. This implies that the incidence vectors of these solutions satisfy equation $rx + qy = \beta$. Consequently, $rx^{U^0} + qy^{I^0} = rx^{U^1} + qy^{I^1} = qx^{U^0} + qy^{I^0} + q_e$, implying that $q_e = 0$. Since e is arbitrarily chosen in $E \setminus I^0$, we have

$$q_e = 0 \qquad \text{for all } e \in E \setminus I^0. \tag{4.72}$$

Now, consider the solution (U^2, I^2) obtained as follows. Consider a demand $l \in K$ and let $e = t_i t_{i+1}$ be an edge between two successive terminals t_i and t_{i+1} of T_l. Let $U_l^2 = (U_l^0 \setminus \{e\}) \cup \{t_i s, s t_{i+1}\}$ where $s \in S_l$ is a Steiner node of demand l. In addition, let $U_j^2 = U_j^0$, $j = 1, ..., K, j \neq l$, and $I^2 = \bigcup_{j \in K} U_j^2$. Now, let us define the solution (U^3, I^3) given by $U^3 = U^2$ and $I^3 = I^2 \cup \{e\}$. Both are feasible solutions of MSOND inducing vectors that are in F_W^k and hence satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^2} + qy^{I^2} = rx^{U^3} + qy^{I^3} = rx^{U^2} + qy^{I^2} + q_e$, which implies that $q_e = 0$. Recall that I^0 is the set of edges between the consecutive terminals of all the demands. As e is arbitrary in I^0, this yields

$$q_e = 0 \qquad \text{for all } e \in I^0. \tag{4.73}$$

By (4.72) and (4.73) we then have

$$q_e = 0 \qquad \text{for all } e \in E. \tag{4.74}$$

In what follows, we will establish some relations between the components of vector r.

First, we will determine the coefficients of edges between Steiner nodes.

Consider a demand $l \in K \setminus \{k\}$ and let s and s' be two Steiner nodes of S_l. Consider the solution (U^0, I^0) defined above and let (U^4, I^4) be the pair defined by $U_l^4 = U_l^0 \cup \{ss'\}$, $U_p^4 = U_p^0$ for all $p \in \{1, ..., K\} \setminus \{l\}$ and $I^4 = \bigcup_{p \in K} U_p^4$. It is clear that (x^{U^0}, y^{I^0}) and (x^{U^4}, y^{I^4}) are both in F_W^k and thus in F. This implies that (x^{U^0}, y^{I^0}) and (x^{U^4}, y^{I^4}) satisfy $rx + qy = \beta$. Since $q = 0$, we have $rx^{U^0} = rx^{U^4} = r^k x^{U^0} + r_{ss'}^l$, which implies $r_{ss'}^l = 0$. As demand l and Steiner nodes s and s' are arbitrary, we have

$$r_{ss'}^l = 0 \qquad \text{for all } s, s' \in S_l,\ l \in K \setminus \{k\}. \tag{4.75}$$

Now, consider demand k and suppose that s and s' are Steiner nodes of S_k such that $ss' \notin \delta(W)$. That is, ss' is an edge between two Steiner nodes situated in either W or $\overline{W}$. Let (U^5, I^5) be the solution defined as follows $U_k^5 = U_k^0 \cup \{ss'\}$, $U_p^5 = U_p^0$ for all

$p \in \{1, ..., K\} \setminus \{k\}$ and $I^5 = \bigcup_{p \in K} U_p^5$. Obviously, (x^{U^5}, y^{I^5}) is in F_W^k and hence in F. As a consequence, $rx^{U^0} = rx^{U^5} = rx^{U^0} + r_{ss'}^k$, implying that $r_{ss'}^k = 0$. Hence, we have

$$r_{ss'}^k = 0 \qquad \text{for all}\, s, s' \in S_k,\ ss' \notin \delta(W). \tag{4.76}$$

The case where $ss' \in \delta(W)$ for $s, s' \in S_k$ will be treated at the end of the proof.

Next, we will examine the coefficients of edges between terminals.

Consider a demand $l \in K \setminus \{k\}$ and let t_i and t_{i+1} be two terminals of T_l. Let s be a Steiner node of S_l. Consider the solution (U^6, I^6) given by $U_l^6 = (U_l^0 \setminus \{t_i t_{i+1}\}) \cup \{t_i s, s t_{i+1}\}$, $U_p^6 = U_p^0$, for each $p \in \{1, ..., K\} \setminus \{l\}$ and $I^6 = \bigcup_{p \in K} U_p^6$. As (x^{U^6}, y^{I^6}) is in F_W^k and hence in F, it follows that $rx^{U^0} = rx^{U^6} = rx^{U^0} - r_{t_i t_{i+1}}^l + r_{t_i s}^l + r_{s t_{i+1}}^l$, implying that $r_{t_i t_{i+1}}^l = r_{t_i s}^l + r_{s t_{i+1}}^l$. Therefore

$$r_{t_i t_{i+1}}^l = r_{t_i s}^l + r_{s t_{i+1}}^l \qquad \text{for all } t_i, t_{i+1} \in T_l,\ s \in S_l,\ l \in K \setminus \{k\}. \tag{4.77}$$

Now, suppose that t_i and t_{i+1} are terminals of T_k such that $t_i t_{i+1} \notin \delta(W)$. Suppose, without loss of generality, that t_i and t_{i+1} are in W. Consider a Steiner node $s \in S_k \cap W$. Along the same line, we can prove that $r_{t_i t_{i+1}}^k = r_{t_i s}^k + r_{s t_{i+1}}^k$. Hence,

$$r_{t_i t_{i+1}}^k = r_{t_i s}^k + r_{s t_{i+1}}^k \qquad \begin{array}{l} \text{for all } t_i, t_{i+1} \in T_k \cap W \text{ (resp. } T_k \cap \overline{W}), \\ s \in S_k \cap W \text{ (resp. } S_k \cap \overline{W}). \end{array} \tag{4.78}$$

Suppose now that $t_i t_{i+1} \in \delta(W)$ such that $t_i \in W$ and $t_{i+1} \in \overline{W}$. Consider two Steiner nodes s and s' of S_k such that $s \in W$ and $s' \in \overline{W}$. Let (U^7, I^7) be the solution given by $U_k^7 = (U_k^0 \setminus \{t_i t_{i+1}\}) \cup \{t_i s, ss', s' t_{i+1}\}$, $U_p^7 = U_p^0$, for each $p \in \{1, ..., K\} \setminus \{k\}$ and $I^7 = \bigcup_{p \in K} U_p^7$. As (x^{U^0}, y^{I^0}) and (x^{U^7}, y^{I^7}) are both in F_W^k and thus in F, we have $rx^{U^0} = rx^{U^7} = rx^{U^0} - r_{t_i t_{i+1}}^k + r_{t_i s}^k + r_{ss'}^k + r_{s' t_{i+1}}^k$, yielding $r_{t_i t_{i+1}}^k = r_{t_i s}^k + r_{ss'}^k + r_{s' t_{i+1}}^k$. As nodes t_i, t_{i+1}, s and s' are all arbitrary, we then have

$$r_{t_i t_{i+1}}^k = r_{t_i s}^k + r_{ss'}^k + r_{s t_{i+1}}^k \qquad \begin{array}{l} \text{for all } t_i, t_{i+1} \in T_k,\ t_i \in W,\ t_{i+1} \in \overline{W}, \\ \text{for all } s, s' \in S_k,\ s \in W,\ s' \in \overline{W}. \end{array} \tag{4.79}$$

In the following, we will look at the coefficients of edges between a terminal and Steiner nodes.

To this end, consider a demand $l \in K \setminus \{k\}$ and let (U^8, I^8) be the pair obtained from (U^7, I^7) as follows: $U_l^8 = (U_l^7 \setminus \{s t_{i+1}\}) \cup \{ss', s' t_{i+1}\}$, $U_p^8 = U_p^7$ for each $p \in$

$\{1, ..., K\} \setminus \{l\}$ and $I^8 = \bigcup_{j\in K} U_j^8$, where s and s' are Steiner nodes of S_l. Since (x^{U^7}, y^{I^7}) and (x^{U^8}, y^{I^8}) are both in $F_W^{k,j}$ and thus in F, this implies $rx^{U^7} = rx^{U^8} = rx^{U^7} - r^l_{st_{i+1}} + r^l_{ss'} + r^l_{s't^k_{i+1}}$. By (4.75), it follows that $r^l_{st_{i+1}} = r^l_{s't_{i+1}}$. As demand l, and nodes t_i, s and s' are arbitrary, we have

$$r^l_{st_i} = r^l_{s't_i} = \lambda_1^l(t_i) \qquad \begin{array}{l}\text{for all } t_i \in T_l,\ s, s' \in S_l,\ l \in K \setminus \{k\},\\ \text{for some } \lambda_1^l(t_i) \in \mathbb{R}.\end{array} \tag{4.80}$$

Now, consider demand k. Here, we distinguish 3 cases.

First, consider a terminal t_i of T_k, say in w, and let s and s' be two Steiner nodes of S_k. Consider the solution (U^9, I^9) defined as follows: $U_k^9 = (U_k^0 \setminus \{t_i t_{i+1}\}) \cup \{t_i s, st_{i+1}\}$, $U_p^9 = U_p^0$, for $p \in \{1, ..., K\} \setminus \{k\}$ and $I^9 = \bigcup_{p\in K} U_p^9$. We also define (U^{10}, I^{10}) by $U_k^{10} = (U_k^9 \setminus \{t_i s\}) \cup \{t_i s', ss'\}$, $U_p^{10} = U_p^9$, for each $p \in \{1, ..., K\} \setminus \{k\}$ and $I^{10} = \bigcup_{p\in K} U_p^{10}$. Note that (x^{U^9}, y^{I^9}) and $(x^{U^{10}}, y^{I^{10}})$ are both in F_W^k and thus in F. As a consequence, we have $rx^{U^9} = rx^{U^{10}} = rx^{U^{10}} - r^k_{t_i s} + r^k_{t_i s'} + r^k_{ss'}$. As by (4.76) $r^k_{ss'} = 0$, we have $r^k_{t_i s} = r^k_{t_i s'}$. And since t_i, s and s' are arbitrary, we obtain that

$$r^k_{t_i s} = r^k_{t_i s'} = \lambda_1^l(t_i) \qquad \begin{array}{l}\text{for all } t_i, t_{i+1} \in T_k \cap W \text{ (resp. } T_k \cap \overline{W}),\\ \text{for all } s, s' \in S_k \cap W \text{ (resp. } S_k \cap \overline{W}),\\ \text{for some } \lambda_1^l(t_i) \in \mathbb{R}.\end{array} \tag{4.81}$$

Now, we will consider the case where t_i and s' are in W and $s \in \overline{W}$. Similarly, we can have the following relation $r^k_{t_i s} = r^k_{t_i s'} + r^k_{ss'}$. And since t_i, s and s' are arbitrary, we have

$$r^k_{st_i} = r^k_{s't_i} + r^k_{ss'} \qquad \begin{array}{l}\text{for all } t_i \in T_k,\ s, s' \in S_k,\\ st_i \in \delta(W) \text{ and } s't_i \notin \delta(W).\end{array} \tag{4.82}$$

Now, we will go back to the case that we left in the beginning of the proof, concerning edges between Steiner nodes of S_k that belong to $\delta(W)$.

Consider demand k, a terminal t_1 of T_k and let s_1, s_2, s_3 and s_4 be Steiner nodes of S_k such that, s_1 and s_3 are in W, and t_1, s_2 and s_4 are in $\overline{W}$ (s_1 and s_3, resp. s_2 and s_4,may be the same). By (4.82) and as $s_1 s_2 \in \delta(W)$ and $s_3 s_4 \in \delta(W)$, we have the following, $r^k_{t_1 s_1} = r^k_{t_1 s_2} + r^k_{s_1 s_2}$ and $r^k_{t_1 s_3} = r^k_{t_1 s_4} + r^k_{s_3 s_4}$. Moreover, by (??) and since $s_1 s_3 \notin \delta(W)$ and $s_2 s_4 \notin \delta(W)$, it follows that $r^k_{t_1 s_1} = r^k_{t_1 s_3}$ and $r^k_{t_1 s_2} = r^k_{t_1 s_4}$, yielding $r^k_{s_1 s_2} = r^k_{s_3 s_4}$. As s_1, s_2, s_3 and s_4 are all arbitrary in S_k, we then have

$$r^k_{s_1 s_2} = r^k_{s_3 s_4} = \rho \qquad \begin{array}{l}\text{for all } s_1, s_2, s_3, s_4 \in S_k,\\ s_1, s_3 \in W \text{ and } s_2, s_4 \in \overline{W},\\ \text{for some } \rho \in \mathbb{R}.\end{array} \tag{4.83}$$

Now, let $\rho \in \mathbb{R}$ be as given by (4.83) and $\lambda^l = (\lambda_1^l, \lambda_2^l)$, $l \in K$ such that $\lambda_1^l = (\lambda_1^l(t_i), t_i \in T_k)$ where $\lambda_1^l(t_i)$ is as given by (4.80), (4.81) and (**??**). $\lambda_2^l = (\lambda_2^l(uv), u, v \in T_k, uv \notin U_l^0)$ such that $\lambda_2^l(uv) = r_{uv}^l - \lambda_1^l(u) - \lambda_1^l(v)$, $l \in K$.

Overall, the coefficients r_{uv}^l for all $uv \in E$ and $l \in K$ can then be expressed in terms of ρ, λ_1^l and λ_2^l as follows

$$r_{uv}^l = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } l \neq k,\ uv = t_i t_{i+1},\ t_i, t_{i+1} \in T_l, \\ \lambda_1^k(u) + \lambda_1^k(v) & \text{if } l = k,\ uv = t_i t_{i+1},\ t_i, t_{i+1} \in T_k, uv \notin \delta(W), \\ \rho + \lambda_1^k(u) + \lambda_1^k(v) & \text{if } l = k,\ uv \in \delta(W), \\ \lambda_1^l(u) & \text{if } l \neq k,\ u \in T_l,\ v \in S_l, \\ \lambda_1^k(u) & \text{if } l = k,\ u \in T_k,\ v \in S_k, uv \notin \delta(W) \\ \rho + \lambda_1^k(u) & \text{if } l = k,\ u \in T_k,\ v \in S_k, uv \in \delta(W) \\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_i t_j,\ t_i, t_j \in T_l,\ j > i, \\ 0 & \text{if } l \neq k \text{ and } uv = s_i s_j,\ s_i, s_j \in S_l,\ j \neq i, \\ 0 & \text{if } l = k \text{ and } uv = s_i s_j,\ s_i, s_j \in S_k,\ uv \notin \delta(W), \\ \rho & \text{if } l = k \text{ and } uv = s_i s_j,\ s_i, s_j \in S_k,\ uv \in \delta(W). \end{cases}$$

Clearly, $r^l = \rho a^l + \lambda_1^l M_1^l + \lambda_2^l M_2^l$, for all $l \in K$. As a consequence, $r = \rho a + \lambda M$ and the result follows.

□

4.3.2 Steiner non-successive terminals inequalities

In this section, we introduce a new family of valid inequalities for the MSOND problem. These come enhancing the constraints related to disjunction and order between the terminals of a given demand.

In Figure 4.8, is presented a graph consisting of six nodes, four terminals (nodes 1, 2, 3 and 4) and two Steiner nodes (nodes 5 and 6). Figure 4.8 shows a solution for an instance containing a demand between terminals 1 and 3. This is routed using the paths $L'_1 = (1, 2, 3)$ and $L'_2 = (1, 4, 3)$. Remind that, as mentioned in Chapter 3, the MSOND problem for a demand reduces to looking for a cycle going in order through the terminals of the demand. In the case of Figure 4.8, this amounts to calculating a cycle visiting the terminals 1, 2, 3 and 4, and respecting the order (1 2 3 4 1).

Figure 4.8 shows a fractional solution for this instance. Let $\bar{x}$ be the solution given by $\bar{x}_{e_i} = \frac{1}{2}$ for $i = 1, ..., 8$ and $\bar{x}_{e_i} = 1$ for $i = 9, 10$. Clearly, $\bar{x}$ satisfies all the constraints of (4.6) as well as the Steiner cut inequalities (4.71) previously introduced.

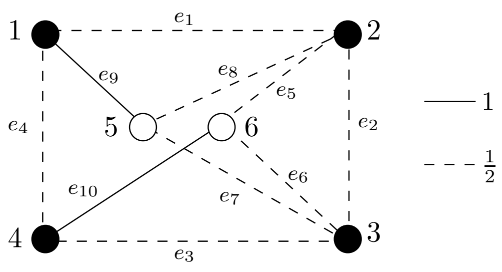


Figure 4.8: Second fractional solution

Observe that, in this solution, the Steiner node 5 has three incident edges e_7, e_8 and e_9 such that $\bar{x}_{e_7} = \frac{1}{2}$, $\bar{x}_{e_8} = \frac{1}{2}$ and $\bar{x}_{e_9} = 1$. This implies that the Steiner node 5 is used to route two sections, namely section $(1, 2)$ and section $(2, 3)$, violating hence the sections' disjunction constraint. To cut this fractional point and strengthen the disjunction aspect of problem, one can add the inequality

$$x_{e_8} \geq x_{e_9},$$

which is valid for MSOND(G, K, T). This inequality express the fact that if edge e_9 is considered in a solution, it should be used to route only one among the sections adjacent to terminal 1, that is to say either section $(1\,2)$ or section $(4\,1)$. As in this case the Steiner node 5 can be used to route section $(1, 2)$, this means that if edge e_9 is considered, then edge e_8 must also be taken in the solution.

This can be generalized as follows. Consider a demand $k \in K$ and let t_j be a terminal of T_k. Consider a Steiner node s of S_k and denote $f = st_j$. Denote the edges linking the Steiner node s with the terminals of T_k not successive to t_j by $e_1, e_2, ..., e_p$ (see Figure 4.9).

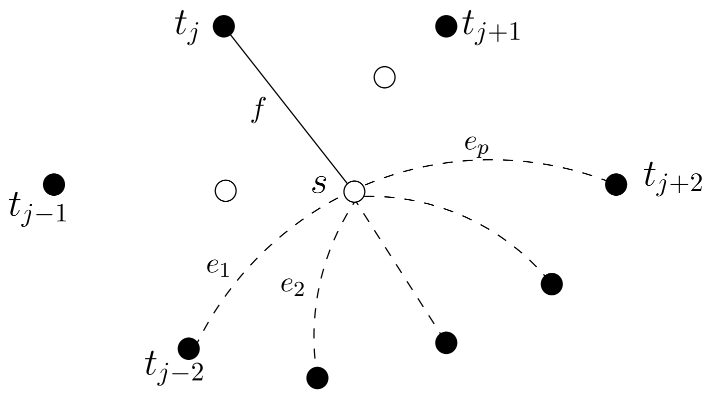


Figure 4.9:

Remark that if the edge f is considered in a solution $\mathcal{S}$, it can be used to route only one among the sections $(t_{j-1}\, t_j)$ and $(t_j\, t_{j+1})$. Thus, none of the edges $e_1, e_2, ..., e_p$ could be considered in the solution $\mathcal{S}$.

This can be expressed by the inequality

$$\sum_{e\in\delta'(s)} x_e^k \geq x_f^k, \tag{4.84}$$

where $\delta'(s) = \delta(s) \setminus \{f, e_1, e_2, ..., e_p\}$.

Note that inequality (4.84) can be viewed otherwise. Indeed, one could say that the flow entering from terminal t_j to the steiner s must be conserved when leaving the Steiner node s, and must be used to route only sections $(t_{j-1}\, t_j)$ and $(t_j\, t_{j+1})$.

In the following, we propose a generalization of inequality (4.84).

Consider a demand $k \in K$ such that $|T_k| \geq 4$ and let t_j be a terminal of T_k. Consider a subset of Steiner nodes $S \in S_k$ and let $\Pi = (V_0, V_1, ..., V_p)$, $p \geq 4$ be a partition of V (see Figure 4.10) such that:

1) $V_0 = S$,
2) $V_1 \cap T_k = \{t_{j-l}, ..., t_{j-2}, t_{j-1}\}$, that is V_1 contains a sequence of successive terminals ending by t_{j-1},
3) $V_2 = \{t_j\}$,
4) $V_3 \cap T_k = \{t_{j+1}, t_{j+2}, ..., t_{j+l'}\}$, that is V_1 contains a sequence of successive terminals ending by $t_{j+l'}$,
5) $V_4, ..., V_p$ are such that $V_i \cap T_k \neq \emptyset$ and $V_i \cap S_k = \emptyset$, $i = 5, ..., p$.

Denote F_{j-1}, F_j, F_{j+1} and E_j the sets of edges of E given by:

- $F_{j-1} = [V_0, V_1]$,
- $F_j = [V_0, V_2]$,
- $F_{j+1} = [V_0, V_3]$,
- $E_j = \bigcup\limits_{i=4}^{p}([V_0, V_i])$.

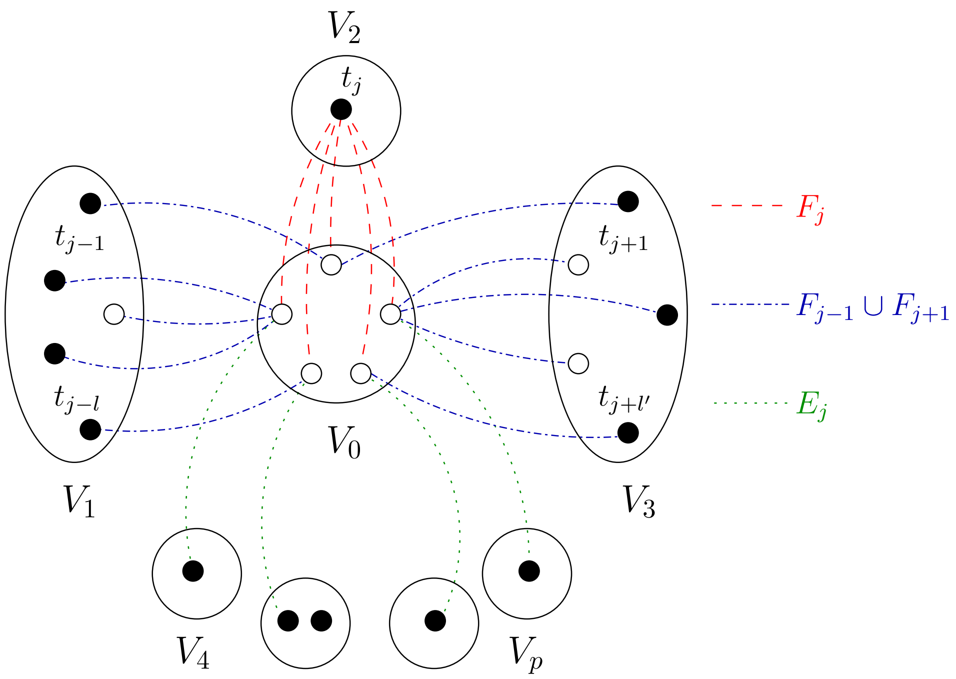


Figure 4.10:

With partition Π and the sets of edges F_{j-1}, F_j, F_{j+1} and E_j, we associate the following inequality

$$\sum_{e \in \delta'(S)} x_e^k \geq \sum_{e \in F_j} x_e^k, \tag{4.85}$$

where $\delta'(S) = F_{j-1} \cup F_{j+1} = \delta(S) \setminus \{E_j, F_j\}$.

Inequality (4.85) implies the following. The flow going from t_j to a subset of Steiner nodes $S \subseteq S_k$ must be conserved in S and used only to route sections that are adjacent to t_j.

Inequalities of type (4.85) will be called *Steiner non-successive terminals inequalities.*

Proposition 4.18 *Inequality* (4.85) *is valid for MSOND*(G, K, T).

Proof Case 1.
We will first prove the validity for the case where $V_1 \cap T_k = \{t_{j-1}\}$ and $V_3 \cap T_k = \{t_{j+1}\}$. Let (U, I) be a solution of the MSOND problem, with $U = (U_1, \ldots, U_{|K|}$. Let F_{j-1,U_k}, F_{j,U_k}, F_{j+1,U_k} and E_{j,U_k} be the intersection of U_k with the sets F_{j-1}, F_j, F_{j+1} and E_j, respectively.

- if $F_{j,U_k} = \emptyset$, that is $|F_{j,U_k}| = 0$, then trivially (x^U, y^I) satisfies (4.85).
- if $F_{j,U_k} \neq \emptyset$, this implies that there is some flow going from V_2 to V_0. Moreover, by (4.2), we know that F_{j,U_k} contains at most 2 edges, that is to say $|F_{j,U_k}| \leq 2$. Suppose by contradiction that $|F_{j-1,U_k} \cup F_{j+1,U_k}| < |F_{j,U_k}|$.
  - if $|F_{j,U_k}| = 1$, this implies that $F_{j-1,U_k} = \emptyset$ and $F_{j+1,U_k} = \emptyset$, which is impossible since every edge leaving V_2 to V_0 must be used in a path linking t_j either to t_{j-1} or to t_{j+1}. As a consequence, $|F_{j-1,U_k} \cup F_{j+1,U_k}| \geq 1$.
  - if $|F_{j,U_k}| = 2$, we have three cases. Either $|F_{j-1,U_k}| = 0$ and $|F_{j,U_k}| = 0$, or $|F_{j-1,U_k}| = 1$ and $|F_{j,U_k}| = 0$, or $|F_{j-1,U_k}| = 0$ and $|F_{j,U_k}| = 1$. As the two edges incident to t_j must be used to link t_j to t_{j-1} via two node-disjoint paths not intersecting $T_k \setminus \{t_{j-1}, t_j, t_{j+1}\}$, this is impossible. Consequently, $|F_{j-1,U_k} \cup F_{j+1,U_k}| \geq 2$.

  As a conclusion, $|F_{j-1,U_k} \cup F_{j+1,U_k}| \geq |F_{j,U_k}|$, and hence (x^U, y^I) satisfies (4.85).

Case 2.
Now suppose that V_1 and V_3 may intersect T_k in more than one terminal. Let us denote by T_{j-1} the super node obtained by contracting the terminals $t_{j-l}, ..., t_{j-2}, t_{j-1}$. Similarly, let T_{j+1} be the super node obtained by contacting the terminals $t_{j+1}, t_{j+2}, ..., t_{j+l'}$. Along the same line of Case 1., by considering the terminals T_{j-1}, t_j and T_{j+1}, we obtain the validity of inequality (4.85).

□

Theorem 4.19 *Inequality (4.85) defines a facet of MSOND(G, K, T) if and only if $V_1 \cap T_k = \{t_{j-1}\}$ and $V_3 \cap T_k = \{t_{j+1}\}$.*

Proof Let $F^k_{j,S}$ be the face induced by inequality (4.85) correponding to demand k, the terminal t_j of T_k and the set of Steiner nodes $S \subseteq S_k$, that is

$$F^k_{j,S} = \{(x, y) \in \text{MSOND}(G, K, T) : \sum_{e \in F_{j-1} \cup F_{j+1}} x^k_e = \sum_{e \in F_j} x^k_e\}.$$

Necessity

First, assume that the condition of Theorem 4.19 is not satisfied, that is V_1 contains terminals predecessor to t_{j-1} and/or V_3 contains terminals successor to t_{j+1}. Consider the partition $\Pi' = (V'_0, V'_1, ..., V'_p)$, where

$$\begin{array}{rcl} V'_1 & = & V_1 \setminus \{t_{j-l}, \dots, t_{j-2}\}, \\ V'_4 & = & V_4 \cup \{t_{j-l}, \dots, t_{j-2}\}, \\ V'_i & = & V_i, \text{ otherwise.} \end{array}$$

It is clear that the left hand side of inequality (4.85) with respect to partition Π' is less or equal than that of partition Π. Moreover, the right hand side of inequality (4.85) is the same for both partitions Π and Π'. This implies that inequality (4.85) written for partition Π' dominates the one written for partition Π. As a consequence, $F^k_{j,S}$ is not a facet defining of MSOND(G, K, T).

Sufficiency

In what follows, we will assume that $V_1 \cap T_k = \{t_{j-1}\}$ and $V_3 \cap T_k = \{t_{j+1}\}$. Note here that V_1 and V_3 could contain Steiner nodes of S_k.

Denote inequality (4.85) by $ax + by \leq \alpha$. Let $rx + qy \leq \beta$ be a valid inequality defining a facet F of MSOND(G, K, T) such that $F^k_{j,S} \subseteq F$. In the following, we will prove that there exist $\rho \in \mathbb{R}$ and $\lambda = (\lambda^l, l \in K)$, $\lambda^l \in \mathbb{R}^{|T_l|+p_l}$ for $l \in K$, such that $q = \rho b$ and $r = \rho a + \lambda M$ (where $r = (r^1, r^2, ..., r^{|K|})$ with $r^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ and M is the matrix of equations defined above). Notice here that $a = (a^1, a^2, ..., a^{|K|})$ such that $a^i \in \mathbb{R}^m$, $i = 1, ..., |K|$ with $a^i = 0$ for $i \in \{1, ..., |K|\} \setminus \{k\}$, $a^k_e \neq 0$ for every $e \in F_{j-1} \cup F_j \cup F_{j+1}$ and $a^k_{e'} = 0$ for every $e' \in E \setminus (F_{j-1} \cup F_j \cup F_{j+1})$. Remark also that $b = 0$.

First, we prove that $q = 0$.

Consider the solution (U^0, I^0) and let $e \in E \setminus I^0$ be an arbitrary edge. Consider the solution (U^1, I^1) given by $U^1 = U^0$ and $I^1 = I^0 \cup \{e\}$. Note that the incidence vectors of solutions (U^0, I^0) and (U^1, I^1) are in $F^k_{j,S}$. This means that (x^{U^0}, y^{I^0}) and (x^{U^1}, y^{I^1}) satisfy equation $rx+qy = \beta$. Consequently, $rx^{U^0}+qy^{I^0} = rx^{U^1}+qy^{I^1} = qx^{U^0}+qy^{I^0}+q_e$, implying that $q_e = 0$. Since e is arbitrarily chosen in $E \setminus I^0$, we have

$$q_e = 0 \qquad \text{for all } e \in E \setminus I^0. \tag{4.86}$$

Now, consider the solution (U^2, I^2) obtained as follows. Consider a demand $l \in K$ and let $e = t_i t_{i+1}$ be an edge between two successive terminals t_i and t_{i+1} of T_l. Let $U^2_l = (U^0_l \setminus \{e\}) \cup \{t_i s, s t_{i+1}\}$ where $s \in S_l$ is a Steiner node of demand l (if $l = k$ s could be either in S or in $S_k \setminus S$). In addition, let $U^2_j = U^0_j$, $j = 1, ..., K, j \neq l$, and $I^2 = \bigcup_{j \in K} U_j^2$. Now, let us define the solution (U^3, I^3) given by $U^3 = U^2$ and $I^3 = I^2 \cup \{e\}$. The incidence vectors of both solutions (U^2, I^2) and (U^3, I^3) are in $F_{j,S}^k$, and hence they satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^2} + qy^{I^2} = rx^{U^3} + qy^{I^3} = rx^{U^2} + qy^{I^2} + q_e$, which implies that $q_e = 0$. Recall that I^0 is the set of edges between the consecutive terminals of all the demands. As e is arbitrary in I^0, this yields

$$q_e = 0 \qquad \text{for all } e \in I^0. \tag{4.87}$$

By (4.86) and (4.87) we then have

$$q_e = 0 \qquad \text{for all } e \in E. \tag{4.88}$$

In what follows, we will establish some relations between the components of vector r.

First, we will examine the coefficients between Steiner nodes.

Consider a demand $l \in K \setminus \{k\}$ and let s and s' be two Steiner nodes of S_l. Consider the solution (U^0, I^0) defined above and let (U^4, I^4) be the pair defined by $U_l^4 = U_l^0 \cup \{ss'\}$, $U_p^4 = U_p^0$ for all $p \in \{1, ..., K\} \setminus \{l\}$ and $I^4 = \bigcup_{p \in K} U_p^4$. Clearly (x^{U^0}, y^{I^0}) and (x^{U^4}, y^{I^4}) are both in $F_{j,S}^k$ and thus in F. This implies that (x^{U^0}, y^{I^0}) and (x^{U^4}, y^{I^4}) satisfy eqation $rx + qy = \beta$. Since $q = 0$, we have $rx^{U^0} = rx^{U^4} = r^k x^{U^0} + r_{ss'}^l$, which implies $r_{ss'}^l = 0$. As demand l and Steiner nodes s and s' are arbitrary, we have

$$r_{ss'}^l = 0 \qquad \text{for all } s, s' \in S_l,\ l \in K \setminus \{k\}. \tag{4.89}$$

Now, consider demand k and suppose that s and s' are Steiner nodes of S_k such that $ss' \notin \delta(S)$. That is, ss' is an edge between two Steiner nodes which are both either in S or in $S_k \setminus S$. Let (U^5, I^5) be the solution defined as follows $U_k^5 = U_k^0 \cup \{ss'\}$, $U_p^5 = U_p^0$ for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^5 = \bigcup_{p \in K} U_p^5$. Obviously, (x^{U^5}, y^{I^5}) is in $F_{j,S}^k$ and hence in F. As a consequence, the incidence vectors of (U^0, I^0) and (U^5, I^5) satisfy equation $rx + qy = \beta$. Since $q = 0$, we have $rx^{U^0} = rx^{U^5} = rx^{U^0} + r_{ss'}^k$ implying that $r_{ss'}^k = 0$. As s and s' are arbitrary in S_k, we have

$$r_{ss'}^k = 0 \qquad \text{for all } s, s' \in S_k,\ ss' \notin \delta(S). \tag{4.90}$$

Now, consider four Steiner nodes of demand k denoted s_1, s_2, s_3 and s_4 such that $s_1, s_3 \in V_0$ and $s_2, s_4 \in V_1$ (resp. $s_2, s_4 \in V_3$). Note that s_1 may coincide with s_3, and similarly s_2 may coincide with s_4. Let (U^6, I^6) be the solution defined as follows $U_k^6 =$

$U_k^0 \setminus t_jt_{j+1} \cup \{t_js_1, s_1s_2, s_2t_{j+1}\}$, $U_p^6 = U_p^0$ for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^6 = \bigcup_{p\in K} U_p^6$. Consider also the solution (U^7, I^7) given by $U_k^7 = U_k^0 \setminus t_jt_{j+1} \cup \{t_js_3, s_3s_4, s_4t_{j+1}\}$, $U_p^7 = U_p^0$ for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^7 = \bigcup_{p\in K} U_p^7$. It is clear that (x^{U^6}, y^{I^6}) and (x^{U^7}, y^{I^7}) are in $F_{j,S}^k$ and hence in F. Since $q = 0$, this implies that $rx^{U^6} = rx^{U^7} = rx^{U^6} - r_{s_1s_2}^k + r_{s_1s_3}^k + r_{s_3s_4}^k + r_{s_4s_2}^k$. As $s_1s_3 \notin \delta(S)$ and $s_4s_2 \notin \delta(S)$, by (4.90) we obtain $r_{s_1s_2}^k = r_{s_3s_4}^k$. As Steiner nodes s_1, s_2, s_3 and s_4 are chosen arbitrarily in S_k, we have

$$r_{s_1s_2}^k = r_{s_3s_4}^k = \rho \qquad \begin{array}{l} \text{for all } s_1, s_2, s_3, s_4 \in S_k, \\ s_1s_2 \in \delta(S) \text{ and } s_3s_4 \in \delta(S) \\ \text{for some } \rho \in \mathbb{R}. \end{array} \tag{4.91}$$

Next, we will determine the coefficients between terminals.

Consider a demand $l \in K$ and let t_i and t_{i+1} be two terminals of T_l. Let s be a Steiner node of S_l and consider the solution (U^8, I^8) given by $U_l^8 = (U_l^0 \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U_p^8 = U_p^0$, for each $p \in \{1, ..., K\} \setminus \{l\}$ and $I^8 = \bigcup_{p\in K} U_p^8$. It is clear that when $l \neq k$, the incidence vector of solution (U^8, I^8) is in $F_{j,S}^k$. Note here that when $l = k$, one can also easily check that (x^{U^8}, y^{I^8}) is in $F_{j,S}^k$ for all choice of the Steiner node s (either $s \in S$ or $s \in S_k \setminus S$) and for every choice of terminals t_i and t_{i+1} (in particular t_it_{i+1} can be equal to t_jt_{j+1} or $t_{j-1}t_j$). As (x^{U^8}, y^{I^8}) is in $F_{j,S}^k$ and hence in F, it follows that $rx^{U^0} = rx^{U^8} = rx^{U^0} - r_{t_it_{i+1}}^l + r_{t_is}^l + r_{st_{i+1}}^l$, implying that $r_{t_it_{i+1}}^l = r_{t_is}^l + r_{st_{i+1}}^l$. Demand l and nodes s, t_i and t_{i+1} are all arbitrary. Therefore

$$r_{t_it_{i+1}}^l = r_{t_is}^l + r_{st_{i+1}}^l \qquad \text{for all } t_i, t_{i+1} \in T_l,\ s \in S_l,\ l \in K. \tag{4.92}$$

In what follows, we will look at the coefficients of edges between terminals and Steiner nodes.

Consider a demand $l \in K \setminus \{k\}$ and let (U^9, I^9) be the pair obtained from (U^8, I^8) as follows, $U_l^9 = (U_l^8 \setminus \{st_{i+1}\}) \cup \{ss', s't_{i+1}\}$, $U_p^9 = U_p^8$ for all $p \in \{1, ..., K\} \setminus \{l\}$ and $I^9 = \bigcup_{j\in K} U_j^9$, where s and s' are Steiner nodes of S_l. Since (x^{U^8}, y^{I^8}) and (x^{U^9}, y^{I^9}) are both in $F_{j,S}^k$ and thus in F, this implies that $rx^{U^8} = rx^{U^9} = rx^{U^8} - r_{st_{i+1}}^l + r_{ss'}^l + r_{s't_{i+1}}^l$. By (4.89), it follows that $r_{st_{i+1}}^l = r_{s't_{i+1}}^l$. As demand l, and nodes t_i, s and s' are arbitrary, we have

$$r_{st_i}^l = r_{s't_i}^l = \lambda_1^l(t_i) \qquad \begin{array}{l} \text{for all } t_i \in T_l,\ s, s' \in S_l,\ l \in K \setminus \{k\}, \\ \text{for some } \lambda_1^l(t_i) \in \mathbb{R}. \end{array} \tag{4.93}$$

Now, consider demand k. Consider two Steiner nodes s and s' of S_k such that $ss' \notin \delta(S)$. Let (U^{10}, I^{10}) be the solution defined as follows $U_k^{10} = (U_k^0 \setminus \{t_it_{i+1}\}) \cup$

$\{t_is, st_{i+1}\}$, $U_p^{10} = U_p^0$, for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^{10} = \bigcup_{p \in K} U_p^9$. We also define (U^{11}, I^{11}) by $U_k^{11} = (U_k^{10} \setminus \{t_is\}) \cup \{t_is', ss'\}$, $U_p^{11} = U_p^{10}$, for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^{11} = \bigcup_{p \in K} U_p^{11}$. Note that $(x^{U^{10}}, y^{I^{10}})$ and $(x^{U^{11}}, y^{I^{11}})$ are both in $F_{j,S}^k$ and thus in F. As a consequence, we have $rx^{U^{10}} = rx^{U^{11}} = rx^{U^{11}} - r_{t_is}^k + r_{t_is'}^k + r_{ss'}^k$. as by (4.90) $r_{ss'}^k = 0$, we have $r_{t_is}^k = r_{t_is'}^k$. And since t_i, s and s' are arbitrary, we can write

$$r_{t_is}^k = r_{t_is'}^k = \lambda_1^k(t_i) \qquad \begin{array}{l}\text{for all } t_i \in T_k,\ s, s' \in S_k,\ ss' \notin \delta(S),\\ \text{for some } \lambda_1^l(t_i) \in \mathbb{R}.\end{array} \tag{4.94}$$

Now, we will consider the case where $ss' \in \delta(S)$. Similarly, we can have the following relation $r_{t_is}^k = r_{t_is'}^k + r_{ss'}^k$. And since t_i, s and s' are arbitrary, we have

$$r_{st_i}^k = r_{s't_i}^k + r_{ss'}^k = \lambda_1^l(t_i) + \rho \qquad \begin{array}{l}\text{for all } t_i \in T_k,\ s, s' \in S_k,\ ss' \notin \delta(S),\\ \text{for some } \lambda_1^l(t_i) \in \mathbb{R},\ \rho \in \mathbb{R}.\end{array} \tag{4.95}$$

Now, let $\rho \in \mathbb{R}$ be as given by (4.91) and $\lambda^l = (\lambda_1^l, \lambda_2^l)$, $l \in K$ such that $\lambda_1^l = (\lambda_1^l(t_i), t_i \in T_k)$ where $\lambda_1^l(t_i)$ is as given by (4.93), (4.94) and (4.95). $\lambda_2^l = (\lambda_2^l(uv), u, v \in T_k, uv \notin U_l^0)$ such that $\lambda_2^l(uv) = r_{uv}^l - \lambda_1^l(u) - \lambda_1^l(v)$, $l \in K$.

Overall, the coefficients r_{uv}^l for all $uv \in E$ and $l \in K$ can then be expressed in terms of ρ, λ_1^l and λ_2^l as follows

$$r_{uv}^l = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } l \neq k,\ uv = t_it_{i+1},\ t_i, t_{i+1} \in T_l,\\ \lambda_1^l(u) & \text{if } l \neq k,\ u \in T_l,\ v \in S_l,\\ \lambda_1^k(u) & \text{if } l = k,\ u \in T_k,\ v \in S_k, uv \notin \delta(S)\\ \rho + \lambda_1^k(u) & \text{if } l = k,\ u \in T_k,\ v \in S_k, uv \in \delta(S)\\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_it_j,\ t_i, t_j \in T_l,\ j > i,\\ 0 & \text{if } l \neq k \text{ and } uv = s_is_j,\ s_i, s_j \in T_l,\ j \neq i,\\ 0 & \text{if } l = k \text{ and } uv = s_is_j,\ s_i, s_j \in S_k,\ uv \notin \delta(S),\\ \rho & \text{if } l = k \text{ and } uv = s_is_j,\ s_i, s_j \in S_k,\ uv \in \delta(S).\end{cases}$$

It is clear that $r^l = \rho a^l + \lambda_1^l M_1^l + \lambda_2^l M_2^l$, for all $l \in K$. We then deduce that $r = \rho a + \lambda M$ and the result follows.

□

4.3.3 Steiner F-partition inequalities

In this section, we show that the so-called *F-partition inequalities* may arise as valid inequalities in an appropriate form. The F-partition inequalities were first introduced by Mahjoub in 1994 [95]. Further works have shown the efficiency of this class of inequalities to solve different variants of the survivable network design problem (see for instance [76, 99, 22]).

In what follows, we discuss the F-partition inequalities for the MSOND problem. First, we show a fractional solution which is cut by an F-partition inequality.

In Figure 4.11 is shown a graph that consists of six nodes, three terminals $1, 2$ and 3, and three Steiner nodes $4, 5$ and 6.

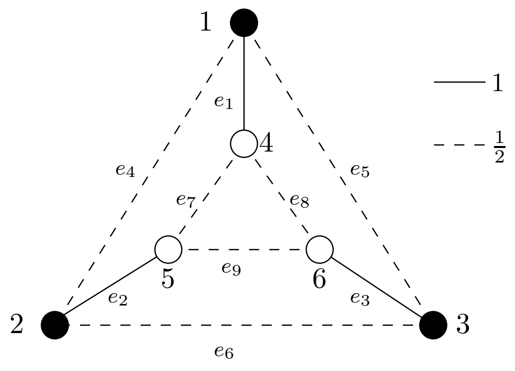


Figure 4.11: Third fractional solution

The instance consists of a demand between terminals 1 and 2 routed by the paths $L'_1 = (1, 2)$ and $L'_2 = (1, 3, 2)$. Let $\bar{x}$ be the solution given by $\bar{x}_{e_i} = 1$ for $i = 1, 2, 3$ and $\bar{x}_{e_i} = \frac{1}{2}$ for $i = 4, ..., 9$.

It is not hard to see that $\bar{x}$ satisfies all the constraints of the linear relaxation of (4.6) and all the valid inequalities previously introduced, namely the 2-connectivity inequalities (4.71) and the Steiner non-successive terminals inequalities (4.85).

However, the fractional solution of Figure 4.11 violates a valid inequality as it will be shown in the following.

Consider the partition $\Pi = (V_0, V_1, V_2, V_3)$ of V given by $V_0 = \{4, 5, 6\}$, $V_1 = \{1\}$, $V_2 = \{2\}$ and $V_3 = \{3\}$. Let $F = \{e_1, e_2, e_3\}$ (see Figure 4.12).

It is not hard to check that $\overline{x}$ violates the inequality $x_{e_4} + x_{e_5} + x_{e_6} \geq 3 - \lfloor \frac{|F|}{2} \rfloor = 2$, which is valid for MSOND(G, K, T).

In the following, we show that this inequality is a special case of a more general class of inequalities.

proposition we state the general form of this violated inequality.

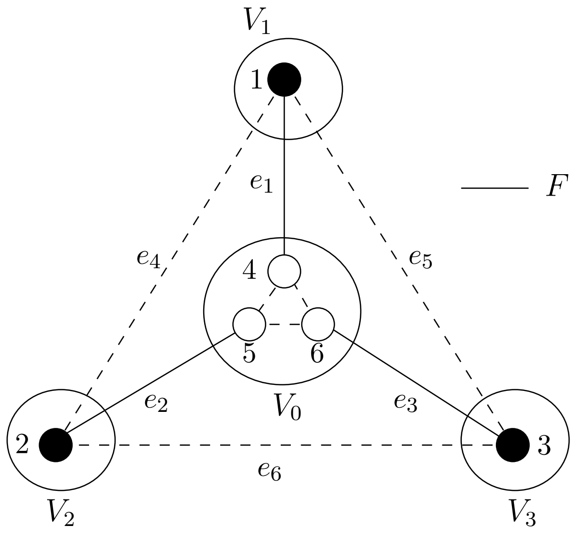


Figure 4.12:

Proposition 4.20 *Consider a demand $k \in K$ and let $\Pi = (V_0, ..., V_p)$, $p \geq 2$ be a partition of V such that $|V_i \cap T_k| \geq 1, i = 1, ..., p$. Let $F \subseteq \delta(V_0)$ such that $|F|$ is odd. Then*

$$x^k(\delta(V_0, ..., V_p) \setminus F) \geq p - \lfloor \frac{|F|}{2} \rfloor \tag{4.96}$$

is valid for MSOND(G, K, T).

Inequalities (4.96) will be called *Steiner F-partition inequalities.*

Proof We prove the validity of the inequality (4.96) for the MSOND(G, K, T) polytope using the Chvátal-Gomory procedure.

The following inequalities are valid for MSOND(G, K, T)

$$\begin{array}{lll} x^k(\delta(V_i)) & \geq 2 & \text{for all}\, i = 1, ..., p, \\ -x^k(f) & \geq -1 & \text{for all}\, f \in F, \\ x^k(g) & \geq 0 & \text{for all}\, g \in \delta(W) \setminus F. \end{array}$$

By summing these inequalities, we obtain

$$2x^k(\delta(V_0, ..., V_p) \setminus F) \geq 2p - |F|$$

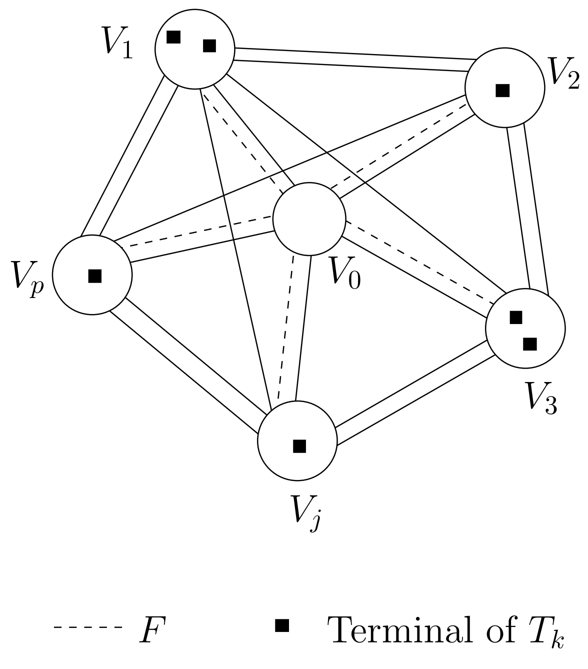


Figure 4.13: Steiner F-partition

By dividing by 2 and rounding up the right-hand side, we obtain

$$x^k(\delta(V_0, ..., V_p) \setminus F) \geq p - \lfloor \frac{|F|}{2} \rfloor.$$

□

Observe that if we denote $|F| = 2q + 1$, inequality (4.96) can also be written as

$$x^k(\delta(V_0, ..., V_p) \setminus F) \geq p - q \tag{4.97}$$

Now, we will study the facial aspect of these inequalities.

First, we give necessary conditions for these inequalities to be facet defining.

Theorem 4.21 *Inequality (4.96) defines a facet for MSOND(G, K, T) only if*

1) each V_i, $i \in \{1, ..., p\}$ is such that

i) V_i does not contain non-successive terminals of T_k, *ii) if $|V_i \cap T_k| \geq 3$, then $S_k \subset V_i$,*

iii) if $|V_i \cap T_k| = 2$, then $V_i \cap S_k \neq \emptyset$.

2) F does not contain any edge between non-successive terminals of T_k,

3) if s and s' are two Steiner nodes of S_k such that $s \in V_i$ and $s' \in V_j$, $i, j \in \{1, ..., p\}$ and $i \neq j$, then V_i and V_j must contain successive terminals.

Proof Let $F^k_{\Pi,F}$ be the face induced by inequality (4.96), that is

$$F^k_{\Pi,F} = \{(x,y) \in \text{MSOND}(G,K,T) : x^k(\delta(V_0,...,V_p) \setminus F) = p - \lfloor \frac{|F|}{2} \rfloor\},$$

1) Suppose that condition 1) is not satisfied. This means that there exists some $i \in \{1,...,p\}$ such that V_i satisfies one of the following statements,

a. V_i contains non-successive terminals of T_k,

b. V_i contains at least 3 successive terminals but not all the Steiner nodes of S_k,

c. V_i contains exactly 2 successive terminals but no Steiner nodes of S_k.

<u>Case a.</u>
Suppose that V_i contains non-successive terminals of T_k. Then, the following hold for any solution of MSOND(G,K,T)

$$\begin{aligned} x^k(\delta(V_i)) &\geq 4 \\ x^k(\delta(V_j)) &\geq 2 && \text{for all } j \in \{1,...,p\} \setminus \{i\}, \\ -x^k(f) &\geq -1 && \text{for all } f \in F, \\ x^k(g) &\geq 0 && \text{for all } g \in \delta(W) \setminus F. \end{aligned}$$

By summing these inequalities, we obtain

$$2x^k(\delta(V_0,...,V_p) \setminus F) \geq 2p + 2 - |F|$$

By dividing by 2, we obtain

$$x^k(\delta(V_0,...,V_p) \setminus F) \geq p + 1 - \frac{|F|}{2}.$$

Now let us replace $|F|$ by $2q+1$. This yields to

$$x^k(\delta(V_0, ..., V_p) \setminus F) \geq p - q + \frac{1}{2} > p - q,$$

which means that any solution of MSOND(G, K, T) does not belong to $F^k_{\Pi,F}$. Hence, $F^k_{\Pi,F}$ is not facet defining for MSOND(G, K, T).

Case b.
Now, suppose that V_i contains 3 successive terminals, say t_1, t_2 and t_3 and that $S_k \not\subseteq V_i$. Consider a Steiner node $s \in V_0$ (s can also be chosen in any other V_j, $j \neq i$ such that $V_j \cap S_k \neq \emptyset$). Let $f = st_2$ be the edge between the terminal t_2 and the Steiner node s. Obviously, the edge f can never be considered in any solution of the face $F^k_{\Pi,F}$. Indeed, if we require the use of edge f, the left hand side of inequality (4.96) rises however the right hand side is unchanged (the proof can be done similarly to Case a., since by the use of edge f we have $x^k(\delta(V_i)) \geq 4$). Consequently, every solution of MSOND inducing a vector in $F^k_{\Pi,F}$ satisfies $x^k_f = 0$, which implies that $F^k_{\Pi,F}$ is not facet defining for MSOND(G, K, T).

Case c.
Now, assume that V_i contains exactly 2 successive terminals, say t_1 and t_2, but no Steiner nodes. Observe that, in this case, to route the section $(t_1\, t_2)$, we have two possibilities. The first is to route $(t_1\, t_2)$ by inserting a Steiner node between t_1 and t_2. The second is to route by using the edge $f = t_1t_2$. Observe that in the first case, inserting a Steiner node between t_1 and t_2 will increase the left hand side of inequality (4.96) with an unchanged right hand side (along the same line as Case b.). This implies that the only possibility to route section $(t_1\, t_2)$ is to use the edge f. As a consequence, every solution of $F^k_{\Pi,F}$ would satisfy $x^k_f = 0$. But this implies that $F^k_{\Pi,F}$ is contained in the face defined by $x^k_f \geq 0$ and cannot define a facet for MSOND(G, K, T).

2) In the sequel, one can suppose that condition 1) is satisfied. Now, we prove the necessity of condition 2). To this end, consider an edge $e = t_it_j$ between two non-successive terminals t_i and t_j and suppose that $e \in F$. Clearly, the following inequalities are valid for MSOND(G, K, T)

$$\begin{aligned} x^k(\delta(V_i)) &\geq 2 && \text{for all } i = 1, ..., p, \\ -x^k(f) &\geq -1 && \text{for all } f \in F \setminus \{e\}, \\ x^k(g) &\geq 0 && \text{for all } g \in \delta(W) \setminus F, \\ -x^k(e) &= 0 && . \end{aligned}$$

By summing these inequalities, we obtain

$$2x^k(\delta(V_0, ..., V_p) \setminus F) \geq 2p - (|F| - 1).$$

As $|F| = 2q + 1$, we then have

$$2x^k(\delta(V_0, ..., V_p) \setminus F) \geq 2p - 2q.$$

By dividing by 2, we obtain

$$x^k(\delta(V_0, ..., V_p) \setminus F) \geq p - q.$$

As a consequence, $F^k_{\Pi,F}$ cannot be facet defining.

3) Suppose that there exist two sets V_i and V_j $(i \neq j)$ such that V_i and V_j contains non-successive terminals. Consider two Steiner nodes s and s' of S_k such that $s \in V_i$ and $s' \in V_j$ and denote $f = ss'$. In what follows, we will prove that f can never be considered in any solution of $F^k_{\Pi,F}$. First, we state the following result.

Claim 4.22 *If edge f is considered in some solution whose incidence vector is in $F^k_{\Pi,F}$, then $x^k(\delta(V_i)) + x^k(\delta(V_j)) \geq 6$.*

Proof If edge f is considered in some solution, there are two possible configurations.

- The first configuration corresponds to the case where we consider the edge f as an extra edge, that is not necessary used to route some section of demand k. Consider solution (U^1, I^1) defined as follows: $U^1_k = U^0 \cap f$ and $U^1_j = U^0_j$, $j = 1, ..., K, j \neq k$, and $I^1 = \bigcup_{j \in K} U^1_j$. In this case, it is not hard to see that $x^{U^1,k}(\delta(V_i)) \geq 3$ and $x^{U^1,k}(\delta(V_j)) \geq 3$ and hence $x^{U^1,k}(\delta(V_i)) + x^{U^1,k}(\delta(V_j)) \geq 6$.
- The second possible configuration is when the edge f is used to route some section of demand k. Suppose, without loss of generality, that this section is (t_i, t_{i+1}), where $t_i \in V_i$ and $t_{i+1} \notin (V_i \cap V_j)$. Consider solution (U^2, I^2) defined as follows: $U^2_k = U^0 \setminus t_i ti + 1 \cap \{t_i s, ss', s' t_{i+1}\}$ and $U^2_j = U^0_j$, $j = 1, ..., K, j \neq k$, and $I^2 = \bigcup_{j \in K} U^1_j$. Clearly, the following hold for solution (U^2, I^2): $x^{U^2,k}(\delta(V_i)) \geq 2$ and $x^{U^2,k}(\delta(V_j)) \geq 4$. And this implies that $x^{U^2,k}(\delta(V_i)) + x^{U^1,k}(\delta(V_j)) \geq 6$.

♦

Using Claim 4.22 and the results developed in the previous sections, the following inequalities are valid for MSOND(G, K, T)

$$\begin{array}{rll} x^k(\delta(V_i)) + x^k(\delta(V_j)) & \geq 6 & \\ x^k(\delta(V_l)) & \geq 2 & \text{for all}\, l \in \{1, ..., p\} \setminus \{i, j\}, \\ -x^k(f) & \geq -1 & \text{for all}\, f \in F, \\ x^k(g) & \geq 0 & \text{for all}\, g \in \delta(W) \setminus F. \end{array}$$

The sum of these inequalities implies

$$2x^k(\delta(V_0,...,V_p)\setminus F) \geq 2(p-2)+6-|F| = 2p+2-|F|$$

By dividing by 2, we obtain

$$x^k(\delta(V_0,...,V_p)\setminus F) \geq p+1-\frac{|F|}{2}.$$

This implies that

$$x^k(\delta(V_0,...,V_p)\setminus F) \geq p-q+\frac{1}{2},$$

which means that $x^k(\delta(V_0,...,V_p)\setminus F)$ can never be equal to $p-q$.

Consequently, every incidence vector of $F^k_{\Pi,F}$ satisfies $x^k_f = 0$, yielding that $F^k_{\Pi,F}$ does not define facet for MSOND(G,K,T). □

Next, we will give sufficient conditions for inequalities (4.96) to be facet defining.

Theorem 4.23 *Inequality (4.96) defines a facet for MSOND(G,K,T) if*

1) every V_i, $i=1,\dots,p$ is such that $|V_i\cap T_k|=1$,

2) V_0 is such that $V_0\cap T_k=\emptyset$ and $|V_0\cap S_k|\geq\lceil\frac{|F|}{2}\rceil$,

3) F is such that

- *i) $|F|=p$ if p is odd and $|F|=p-1$ if p is even,*
- *ii) $|F\cap\delta(V_i)|\leq 1$ for each $i\in\{1,\dots,p\}$,*
- *iii) for each $s_j\in V_0$, if $F\cap\delta(s_j)=\{s_ju,s_jv\}$, where $u\in V_i$ and $v\in V_j$, $i\neq j$, then V_i and V_j must contain successive terminals.*

Proof Recall that $F^k_{\Pi,F}$ denotes the face induced by inequality (4.96).

Denote inequality (4.96) by $ax+by\leq\alpha$. Let $rx+qy\leq\beta$ be a valid inequality defining a facet F of MSOND(G,K,T) such that $F^k_{\Pi,F}\subseteq F$. In what follows, we prove that there exist $\rho\in\mathbb{R}$ and $\lambda=(\lambda^l,l\in K)$, $\lambda^l\in\mathbb{R}^{|T_l|+p_l}$ for $l\in K$, such that $q=\rho b$ and $r=\rho a+\lambda M$. Notice here that $a=(a^1,a^2,...,a^{|K|})$ such that $a^i\in\mathbb{R}^m$, $i=1,...,|K|$ with $a^i=0$ for $i\in\{1,...,|K|\}\setminus\{k\}$, $a^k_e\neq 0$ for every $e\in\delta(\Pi)\setminus F$ and $a^k_{e'}=0$ for every $e'\in E\setminus(\delta(\Pi)\setminus F)$. Note also that $b=0$.

Throughout the proof, we will suppose that the conditions of the theorem are satisfied. In addition, for convenience, we will assume that the edges of F are linking Steiner nodes of V_0 to terminals of V_i, $i = 1, \ldots, p$. Moreover, we will restrict ourselves to the case where p is odd since the case of p even is very similar. Remark that, under these hypothesis, we have an odd-wheel configuration shown in Figure 4.14.

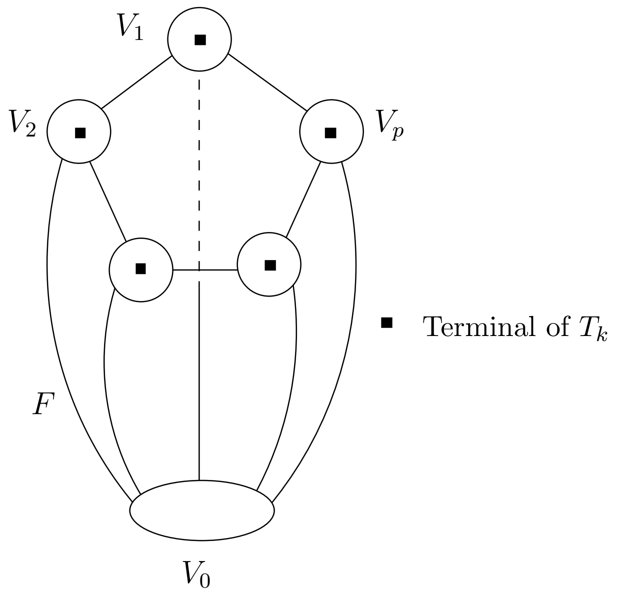


Figure 4.14: An odd wheel configuration

In the sequel, we will suppose that the terminals of $V_1, V_2, \ldots, V_p$ are successive. We will refer to the terminals of these sets by $t_1, t_2, \ldots, t_p$. In addition, denote the edges of F by $f_1, f_2, \ldots, f_p$, where $f_i = t_i s_i$, $i = 1, \ldots, p$.

First, we prove that $q = 0$.

Consider the solution (U^0, I^0) and denote $e_1, e_2, \ldots, e_p$ the edges between successive terminals ($e_i = t_i t_{i+1}$, with $t_{p+1} = t_1$). Note that $U^0 = \{e_1, \ldots, e_p\}$.

Let (U^1, I^1) be the solution given by

$U^1_k = \{e_1, e_2, f_3, s_3 s_4, f_4, e_4, f_5, \ldots, e_{p-1}, f_p, s_{p-1} s_p, f_1\}$ and $U^1_j = U^0_j$, $j = 1, ..., K, j \neq k$, and $I^1 = \bigcup_{j \in K} U^1_j$ (note here that one can have $s_3 = s_4$ and/or $s_{p-1} = s_p$). Consider now an edge $g \in E \setminus I^1$ and let (U^2, I^2) be the solution defined by $U^2 = U^1$ and $I^2 = I^1 \cup \{g\}$. It is not hard to see that the incidence vectors of solutions (U^1, I^1) and (U^2, I^2) are in $F^k_{\text{II},F}$. This means that (x^{U^1}, y^{I^1}) and (x^{U^2}, y^{I^2}) satisfy equation $rx + qy = \beta$. Consequently, $rx^{U^1} + qy^{I^1} = rx^{U^2} + qy^{I^2} = qx^{U^1} + qy^{I^1} + q_e$, implying that $q_e = 0$. Since e is arbitrarily chosen in $E \setminus I^1$, we have

$$q_e = 0 \qquad \text{for all } e \in E \setminus I^1. \tag{4.98}$$

Now consider an edge g_1 of I^1 and a demand $l \in K$.

Here we distinguish two cases.

First suppose that $l \neq k$. This means that $g_1 \in U_l^0$. Denote $g_1 = t_i t_{i+1}$ and let (U^3, I^3) be the solution defined by $U_l^3 = (U_l^1 \setminus \{g_1\}) \cup \{t_i s, s t_{i+1}\}$, where $s \in S_l$ is a Steiner node of demand l. In addition, let $U_j^3 = U_j^1$, $j = 1, ..., K, j \neq l$, and $I^3 = \bigcup_{j \in K} U_j^3$. Now, let us define the solution (U^4, I^4) given by $U^4 = U^3$ and $I^4 = I^3 \cup \{g_1\}$. The incidence vectors of both solutions (U^3, I^3) and (U^4, I^4) are in $F_{\Pi,F}^k$, and hence they satisfy equation $rx + qy = \beta$. As a consequence, we have $rx^{U^3} + qy^{I^3} = rx^{U^4} + qy^{I^4} = rx^{U^3} + qy^{I^3} + q_{g_1}$, which implies that $q_{g_1} = 0$.

Suppose now that $l = k$. Here also, we distinguish two subcases. The first subcase is when $g_1 \in I^1 \cup I^0$ and is similar to the previous one. The second subcase is when $g_1 \in I^1 \setminus I^0$, that is in our case, $g_1 \in I^1 \cap F$. Suppose, without loss of generality, that $g_1 = f_1$. Let (U^5, I^5) be the solution defined by $U_l^5 = (U_l^1 \setminus \{f_1, e_2\}) \cup \{s_p t_1, f_2, s_1 s_2\}$. Also, let $U_j^5 = U_j^1$, $j = 1, ..., K, j \neq l$, and $I^5 = \bigcup_{j \in K} U_j^5$. Now, let us define the solution (U^6, I^6) given by $U^6 = U^5$ and $I^6 = I^5 \cup \{f_1\}$. It is clear that (x^{U^5}, y^{I^5}) and (x^{U^6}, y^{I^6}) are in $F_{\Pi,F}^k$ and hence in F. Thus they satisfy equation $rx + qy = \beta$. Then, we have $rx^{U^5} + qy^{I^5} = rx^{U^6} + qy^{I^6} = rx^{U^5} + qy^{I^5} + q_{f_1}$, which implies that $q_{f_1} = 0$.

As in all cases, g_1 is arbitrary in I^1, this implies that

$$q_e = 0 \qquad \text{for all } e \in I^1. \tag{4.99}$$

By (4.98) and (4.99) we then have

$$q_e = 0 \qquad \text{for all } e \in E. \tag{4.100}$$

In what follows, we will establish some relations between the components of vector r.

First, we will determine the coefficients of edges between Steiner nodes.

Consider a demand $l \in K \setminus \{k\}$ and let s and s' be two Steiner nodes of S_l. Consider the solution (U^1, I^1) defined above and let (U^7, I^7) be the solution defined by $U_l^7 = U_l^1 \cup \{ss'\}$, $U_p^7 = U_p^1$ for all $p \in \{1, ..., K\} \setminus \{l\}$ and $I^7 = \bigcup_{p \in K} U_p^7$. Clearly (x^{U^1}, y^{I^1}) and

(x^{U^7}, y^{I^7}) are both in $F^k_{\Pi,F}$ and thus in F. This implies that (x^{U^1}, y^{I^1}) and (x^{U^7}, y^{I^7}) satisfy equation $rx + qy = \beta$. Since $q = 0$, we have $rx^{U^1} = rx^{U^7} = rx^{U^1} + r^l_{ss'}$, which implies $r^l_{ss'} = 0$. As demand l and Steiner nodes s and s' are arbitrary, we have

$$r^l_{ss'} = 0 \qquad \text{for all}\, s, s' \in S_l,\, l \in K \setminus \{k\}. \tag{4.101}$$

Now consider demand k and suppose that s and s' are Steiner nodes of S_k. Here, we shall distinguish different cases. First, suppose that $ss' \in V_j$ for a given $j \in \{1, \ldots, p\}$. Along the same line, we can prove that $r^k_{ss'} = 0$.

Similarly, we can also prove that $r^k_{ss'} = 0$ for every s and s' of $S_k \cap V_0$ such that $|\delta(s) \cap F| + |\delta(s') \cap F| \leq 1$.

Now suppose that s and s' are in $S_k \cap V_0$ but $\delta(s) \cap F \neq \emptyset$ and $\delta(s') \cap F \neq \emptyset$. Suppose, without loss of generality that $s = s_p$ and $s' = s_1$. Consider again solution (U^1, I^1) and define solution (U^8, I^8) as follows, $U^8_k = U^1_k \cup \{s_p s_1, s_p s_2, s_2 s_1\}$, $U^8_p = U^1_p$ for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^8 = \bigcup_{p \in K} U^8_p$. It is clear that (x^{U^1}, y^{I^1}) and (x^{U^8}, y^{I^8}) are both in $F^k_{\Pi,F}$ and thus in F. This implies that they satisfy equation $rx + qy = \beta$ and hence $rx^{U^1} = rx^{U^8} = rx^{U^0} - r^k_{s_p s_1} + r^k_{s_p s_2} + r^k_{s_2 s_1}$. This implies that $r^k_{s_p s_1} = r^k_{s_p s_2} + r^k_{s_2 s_1}$. By symmetry on all the Steiner nodes in $S_k \cap V_0$ whose incident edges intersect F, we have

$$\begin{aligned} r^k_{s_1 s_2} &= r^k_{s_2 s_3} + r^k_{s_3 s_1}, \\ r^k_{s_2 s_3} &= r^k_{s_3 s_4} + r^k_{s_4 s_2}, \\ &\vdots \\ r^k_{s_p s_1} &= r^k_{s_p s_2} + r^k_{s_2 s_1}, \end{aligned}$$

yielding to $r^k_{ss'} = 0$ for all Steiner nodes s and s' in $S_k \cap V_0$ whose incident edges intersect F.

Overall, we have

$$r^k_{ss'} = 0 \qquad \text{for all}\, s, s' \in S_k,\, ss' \notin \delta(\Pi). \tag{4.102}$$

Now, suppose that s and s' are two Steiner nodes such that $s \in V_j$ and $s' \in V_{j+1}$ for some $j \in \{1, \ldots, p\}$. Suppose also that there is a Steiner node $s'' \in V_{j+1}$. Notice that ss' and ss'' are both in $\delta(\Pi) \setminus F$.

Suppose, without loss of generality, that $j = 1$ (the result can be found by symmetry for all the sets V_i). Consider solution (U^1, I^1) given above and define solution (U^9, I^9)

as follows, $U_k^9 = U_k^1 \setminus \{e_1\} \cup \{t_1s, ss', s't_2\}$, $U_p^9 = U_p^0$ for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^9 = \bigcup_{p\in K} U_p^9$. Let (U^{10}, I^{10}) be the solution defined by $U_k^{10} = U_k^9 \setminus \{ss'\} \cup \{ss'', s''s'\}$, $U_p^{10} = U_p^9$ for all $p \in \{1, ..., K\} \setminus \{k\}$ and $I^{10} = \bigcup_{p\in K} U_p^{10}$. Clearly, (x^{U^9}, y^{I^9}) and $(x^{U^{10}}, y^{I^{10}})$ are both in $F_{\Pi,F}^k$ and then satisfy equation $rx + qy = \beta$. Since $q = 0$, we have $rx^{U^9} = rx^{U^{10}} = r^k x^{U^9} - r_{ss'}^k + r_{ss''}^k + r_{s''s'}^k$. By (4.102), we have $r_{s''s'}^k = 0$, which implies that $r_{ss'}^k = r_{ss''}^k$. As Steiner nodes s, s' and s'' are arbitrary of S_k, we have

$$r_{ss'}^k = r_{ss''}^k = \rho \qquad \begin{array}{l}\text{for all}\, s, s', s'' \in S_k,\ ss', ss'' \in \delta(\Pi) \setminus F,\\ \text{for some}\, \rho \in \mathbb{R}.\end{array} \tag{4.103}$$

Now, we will determine the coefficients of edges between terminals.

Consider a demand $l \in K \setminus \{k\}$ and let t_i and t_{i+1} be two terminals of T_l. Let s be a Steiner node of S_l and consider the solution (U^{11}, I^{11}) given by $U_l^{11} = (U_l^1 \setminus \{t_it_{i+1}\}) \cup \{t_is, st_{i+1}\}$, $U_p^{11} = U_p^1$, for each $p \in \{1, ..., K\} \setminus \{l\}$ and $I^{11} = \bigcup_{p\in K} U_p^{11}$. Clearly, the incidence vector of solution (U^{11}, I^{11}) is in $F_{\Pi,F}^k$ and hence in F. It follows that $rx^{U^1} = rx^{U^{11}} = rx^{U^1} - r_{t_it_{i+1}}^l + r_{t_is}^l + r_{st_{i+1}}^l$, implying that $r_{t_it_{i+1}}^l = r_{t_is}^l + r_{st_{i+1}}^l$. Demand l and nodes s, t_i and t_{i+1} are all arbitrary. Therefore

$$r_{t_it_{i+1}}^l = r_{t_is}^l + r_{st_{i+1}}^l \qquad \text{for all } t_i, t_{i+1} \in T_l,\ s \in S_l,\ l \in K \setminus \{k\}. \tag{4.104}$$

Now consider demand k and suppose that t_i and t_{i+1} are terminals of T_k, such that t_i and t_{i+1} are in V_i and V_{i+1}, respectively. Suppose, without loss of generality, that $t_i = t_1$ and $t_{i+1} = t_2$. Consider solution (U^1, I^1) and let (U^{12}, I^{12}) be the solution defined as follows, $U_k^{12} = (U_k^1 \setminus \{t_1t_2\}) \cup \{t_1s_2, s_2t_2\}$, $U_p^{12} = U_p^1$, for each $p \in \{1, ..., K\} \setminus \{k\}$ and $I^{12} = \bigcup_{p\in K} U_p^{12}$. (x^{U^1}, y^{I^1}) and $(x^{U^{12}}, y^{I^{12}})$ are both in $F_{\Pi,F}^k$ and hence in F. Therefore, they satisfy equation $rx + qy = \beta$. This implies that $rx^{U^1} = rx^{U^{12}} = rx^{U^1} - r_{t_it_{i+1}}^k + r_{t_is_2}^k + r_{s_2t_{i+1}}^k$. Hence $r_{t_it_{i+1}}^k = +r_{t_is_2}^k + r_{s_2t_{i+1}}^k$. As a consequence, we have

$$r_{t_it_{i+1}}^k = r_{t_is}^k + r_{st_{i+1}}^k \qquad \begin{array}{l}\text{for all}\, t_i, t_{i+1} \in T_k,\ s \in S_k,\\ t_it_{i+1}, t_is \in \delta(\Pi),\ \text{and } st_{i+1} \in F.\end{array} \tag{4.105}$$

In what follows, we will look at the coefficients of edges between terminals and Steiner nodes.

Consider a demand $l \in K\setminus\{k\}$ and two Steiner nodes s and s' of S_l. Let (U^{13}, I^{13}) be the solution obtained from (U^7, I^7) as follows, $U_l^{13} = (U_l^7 \setminus \{st_{i+1}\}) \cup \{ss', s't_{i+1}\}$, $U_p^{13} = U_p^7$ for each $p \in \{1, ..., K\} \setminus \{l\}$ and $I^{13} = \bigcup_{j\in K} U_j^{13}$. Since (x^{U^7}, y^{I^7}) and $(x^{U^{13}}, y^{I^{13}})$ are both in $F_{\Pi,F}^k$ and thus in F, this implies $rx^{U^7} = rx^{U^{13}} = rx^{U^7} - r_{st_{i+1}}^l + r_{ss'}^l + r_{s't_{i+1}}^l$.

By (4.101), it follows that $r^l_{st_{i+1}} = r^l_{s't_{i+1}}$. As demand l, and nodes t_i, s and s' are arbitrary, we have

$$r^l_{st_i} = r^l_{s't_i} = \lambda^l_1(t_i) \quad \begin{array}{l}\text{for all } t_i \in T_l,\ s, s' \in S_l,\ l \in K \setminus \{k\},\\ \text{for some } \lambda^l_1(t_i) \in \mathbb{R}.\end{array} \tag{4.106}$$

Now, consider demand k. Consider a terminal t_i of T_k and Steiner nodes s and s' of S_k such that t_i, s and s' belong to the same set, say V_i.

Along the same line, we can prove that

$$r^k_{st_i} = r^k_{s't_i} = \lambda^k_1(t_i) \quad \begin{array}{l}\text{for all } t_i \in T_k,\ s, s' \in S_k,\\ st_i, s't_i \notin \delta(\Pi),\\ \text{for some } \lambda^k_1(t_i) \in \mathbb{R}.\end{array} \tag{4.107}$$

Now, suppose that t_i, s and s' are such that $t_is \notin \delta(\Pi)$ and $t_is' \in \delta(\Pi) \setminus F$. Without loss of generality, we will suppose that $t_i = t_1$ and $s' = s_2$. Consider solution (U^{14}, I^{14}) defined as follows. $U^{14}_k = (U^1_k \setminus \{t_1t_2\}) \cup \{t_1s_2, s_2t_2\}$, $U^{14}_p = U^1_p$, for each $p \in \{1, ..., K\} \setminus \{k\}$ and $I^{14} = \bigcup_{p\in K} U^{14}_p$. Define also solution (U^{15}, I^{15}) given by $U^{15}_k = (U^{14}_k \setminus \{t_1s_2\}) \cup \{t_1s, ss_2\}$, $U^{15}_p = U^{14}_p$, for each $p \in \{1, ..., K\} \setminus \{k\}$ and $I^{15} = \bigcup_{p\in K} U^{15}_p$. Since $(x^{U^{14}}, y^{I^{14}})$ and $(x^{U^{15}}, y^{I^{15}})$ are both in $F^k_{\Pi,F}$ and thus in F, this implies $rx^{U^{14}} = rx^{U^{15}} = rx^{U^{14}} - r^k_{t_1s_2} + r^k_{t_1s} + r^k_{ss_2}$. As t_i, s and s' are all arbitrary, we have

$$r^k_{st_i} = r^k_{s't_i} + r^k_{ss'} = \lambda^l_1(t_i) + \rho \quad \begin{array}{l}\text{for all } t_i \in T_k,\ s, s' \in S_k,\\ t_is' \notin \delta(\Pi), \text{ and } t_is, ss' \in \delta(\Pi) \setminus F,\\ \text{for some } \lambda^k_1(t_i) \in \mathbb{R},\ \rho \in \mathbb{R}.\end{array} \tag{4.108}$$

Now, let $\rho \in \mathbb{R}$ be as given by (4.103) and $\lambda^l = (\lambda^l_1, \lambda^l_2)$, $l \in K$ such that $\lambda^l_1 = (\lambda^l_1(t_i), t_i \in T_k)$ where $\lambda^l_1(t_i)$ is as given by (4.106), (4.107) and (4.108). $\lambda^l_2 = (\lambda^l_2(uv), u, v \in T_k, uv \notin U^0_l)$ such that $\lambda^l_2(uv) = r^l_{uv} - \lambda^l_1(u) - \lambda^l_1(v)$, $l \in K$.

Overall, the coefficients r^l_{uv} for all $uv \in E$ and $l \in K$ can then be expressed in terms of ρ, λ_1^l and λ_2^l as follows

$$r_{uv}^l = \begin{cases} \lambda_1^l(u) + \lambda_1^l(v) & \text{if } l \in K,\ uv = t_i t_{i+1},\ t_i, t_{i+1} \in T_l, \\ \lambda_1^l(u) & \text{if } l \neq k,\ u \in T_l,\ v \in S_l, \\ \lambda_1^k(u) & \text{if } l = k,\ u \in T_k,\ v \in S_k, uv \notin \delta(\Pi), \\ \lambda_1^k(u) & \text{if } l = k,\ u \in T_k,\ v \in S_k, uv \in F, \\ \rho + \lambda_1^k(u) & \text{if } l = k,\ u \in T_k,\ v \in S_k, uv \in \delta(\Pi) \setminus F, \\ \lambda_2^l(uv) + \lambda_1^l(u) + \lambda_1^l(v) & \text{if } uv = t_i t_j,\ t_i, t_j \in T_l,\ j > i, \\ 0 & \text{if } l \neq k \text{ and } uv = s_i s_j,\ s_i, s_j \in T_l,\ j \neq i, \\ 0 & \text{if } l = k \text{ and } uv = s_i s_j,\ s_i, s_j \in S_k,\ uv \notin \delta(\Pi), \\ \rho & \text{if } l = k \text{ and } uv = s_i s_j,\ s_i, s_j \in S_k,\ uv \in \delta(\Pi) \setminus F. \end{cases}$$

It is clear that $r^l = \rho a^l + \lambda_1^l M_1^l + \lambda_2^l M_2^l$, for all $l \in K$. We then deduce that $r = \rho a + \lambda M$ and the result follows.

□

4.3.4 Generalized Steiner partition inequalities

The *partition inequalities* have got a particular interest and arise as valid inequalities for several well-known problems. Introduced by Nash-Williams for the spanning tree problem in the beginning of the 60's [103], these inequalities have been later widely studied [24]. In the beginning of the 90's, Grötschel and Monma use partitions for the connected subgraph polytope [68]. Later, Stoer [126] introduces partition inequalities for both kECON (k Edge Connected) and kNCON (k Node Connected) subgraph problems. The author investigates necessary conditions and sufficient conditions for these inequalities to be facet defining for the kECON and the kNCON polytopes. The partition inequalities have also been efficient to model the Steiner tree problem. In [27, 26], Chopra and Rao introduce the *Steiner partition inequalities* and study their facial aspect for the the Steiner tree polytope. In [15], Barahona and Mahjoub show that the partition inequalities, together with bound, cut and odd-wheel inequalities give a complete description of the 2NCON for Halin graphs. In a further work, Baïou, Barahona and Mahjoub [10] propose a separation algorithm for these inequalities based on submodular functions.

For more results related to partition inequalities, the reader is referred to [11].

Thus it seems to be interesting to look for an adaptation of partition inequalities to our prolem. In what follows, we give a suitable form of these inequalities that we will call *Generalized Steiner Partition Inequalities.*

Proposition 4.24 *Consider a demand $k \in K$ and let $\Pi = (V_1, ..., V_p)$ be a partition of V such that $|V_i \cap T_k| \geq 1, i = 1, ..., p$ $(p \geq 2)$. Suppose that $V_1, ..., V_r$, $r \leq p$ contain respectively $q_i \geq 2$, $i = 1, ..., r$ non-successive terminals (or sequences of terminals).*

Let $S \subseteq S_k$ be a subset of Steiner nodes of demand k. Then

$$x^k(\delta_{G\setminus S}(V_1, V_2, ..., V_p)) \geq (p + \sum_{i=1}^{r} q_i - r) - |S| \tag{4.109}$$

is valid for MSOND(G, K, T).

Proof The idea of the proof is to replace each time the subset V_i of the partition Π by q_i equivalent independent subsets. The proof is done by induction on r.

First, notice that if $|S| \geq p + \sum_{i=1}^{r} q_i - r$, inequalities (4.109) are redundant with respect to the trivial inequalities (4.4).

In the sequel, we assume that $|S| < p + \sum_{i=1}^{r} q_i - r$.

If $r = 0$, this means that all the sets $V_i, i = 1, ..., p$ contains either only one terminal or a sequence of successive terminals. In this case, inequalities (4.109) are equivalent to the following inequalities

$$x^k(\delta_{G\setminus S}(V_1, V_2, ..., V_p)) \geq p - |S| \tag{4.110}$$

Consider a solution $(U, I)$of the problem with $U = (U_1, \ldots, U_{|T_k|})$. If U_k does not use any node of S, then U_k uses ate least p edges from $\delta_{G\setminus S}(V_1, \ldots, V_p)$. As any node of S can be used to route at most one section t_it_{i+1}, then U_k must intersect $\delta_{G\setminus S}(V_1, \ldots, V_p)$ in at least $p - |S|$, and thus inequality 4.110 is satisfied.

Now, suppose that (4.109) are valid for $r = h > 0$ and let us prove its validity for $r = h + 1$.

We know that r $(r = h+1)$ subsets of the partition contain non-successive terminals (or sequences of terminals). Recall that these sets are the r first sets of the partition Π, denoted $V_1, V_2, ..., V_r$. This means that the node sets $V_{r+1}, ...V_p$ contains only successive terminals (or sequences of successive terminals). Suppose, without loss of generality, that the terminals (or sequences of terminals) of the sets V_{r-1} and V_r are pairwise non-successive. Note that this hypothesis is not restrictive since we can always obtain it by a suitable numbering of the sets $V_1, V_2, ..., V_r$.

Consequently, if we combine the sets V_{r-1} and V_r in one set called W, we get exactly $q_{r-1} + q_r$ non-successive terminals (or sequences of terminals) in W.

Now, consider the new partition $\Pi' = (V'_1, ..., V'_{p'})$ where $p' = p - 1$, obtained from Π by combining the two node sets V_{r-1} and V_r. That is, Π' is defined as follows,

$$V'_j = \begin{cases} V_j & \text{if } j \in \{1, ..., r-2\}, \\ V_{r-1} \cup V_r & \text{if } j = r-1, \\ V_{j+1} & \text{if } j \in \{r, ..., p'\}. \end{cases}$$

Observe that by construction, partition Π' contains exactly $r' = r - 1$ node sets $V'_1, V'_2, ..., V'_{r'}$ such that $V'_i, i = 1, ...r'$ contains $q'_i, i = 1, ...r'$ non-successive terminals (or sequences of terminals). Remark also that $q'_i = q_i$ for all $i = 1, ...r' - 1$ and $q'_{r'} = q_{r-1} + q_r$. Since by hypothesis, (4.109) is assumed to be valid for the rank h and $r' = r - 1 = h + 1 - 1 = h$, we can write

$$x^k(\delta_{G\backslash S}(V'_1, V'_2, ..., V'_{p'})) \geq (p' + \sum_{i=1}^{r'} q'_i - r') - |S| \tag{4.111}$$

Recall that by construction, the terminals of V_{r-1} and V_r are pairwise non-successive. This means that there are no edges linking V_{r-1} to V_r in any solution, and hence $x^k(\delta_{G\backslash S}(V_{r-1}, V_r)) = 0$. We can deduce, in consequence, that :

$$x^k(\delta_{G\backslash S}(V'_1, V'_2, ..., V'_{p'})) = x^k(\delta_{G\backslash S}(V_1, V_2, ..., V_p)) \tag{4.112}$$

Moreover we know that

$$\begin{aligned}(p' + \sum_{i=1}^{r'} q'_i - r') - |S| &= (p - 1 + \sum_{i=1}^{r-2} q_i + q'_{r-1} - (r-1)) - |S| \\ &= (p - 1 + \sum_{i=1}^{r-2} q_i + (q_{r-1} + q_r) - r + 1) - |S| \\ &= (p + \sum_{i=1}^{r} q_i - r) - |S|.\end{aligned}$$

consequently,

$$(p' + \sum_{i=1}^{r'} q'_i - r') - |S| = (p + \sum_{i=1}^{r} q_i - r) - |S|. \tag{4.113}$$

The result follows from (4.112) and (4.113). □

Notice that the Generalized Steiner Partition inequalities written for particular values of p, r and $|S|$, coincide with some known inequalities in the literature.

- If $p = 2$, $r = 0$ and $|S| = 1$, denote $W = V_1$ ($\overline{W} = V_2$) and $S = \{s\}$. The Generalized Steiner Partition inequalities (4.109) are hence equivalent to
$$x^k(\delta_{G\setminus s}(W)) \geq 1. \tag{4.114}$$
Inequalities (4.114) are known as the *node cut constraints* for the 2NCON problem [126].

- If $p > 2$ and $r = 0$, the Generalized Steiner Partition inequalities (4.109) are equivalent to
$$x^k(\delta_{G\setminus S}(V_1, V_2, ..., V_p)) \geq p - |S| \tag{4.115}$$
Inequalities (4.115) known as the *node partition constraints* for the kNCON problem [126].

Proposition 4.25 *Consider a demand $k \in K$ and let $\Pi = (V_1, ..., V_p)$ be a partition of V such that $|V_i \cap T_k| \geq 1, i = 1, ..., p$ ($p \geq 2$). Suppose that $V_1, ..., V_r$, $r \leq p$ contain respectively $q_i \geq 2$, $i = 1, ..., r$ non-successive terminals (or sequences of terminals). Let $S \subseteq S_k$ be a subset of Steiner nodes of demand k.*

Inequalities (4.109) are redundant with respect to inequalities (4.71), (4.2) and (4.4).

Proof Denote by $V_{i,j}$, $i = 1, ..., p$ and $j = 1, ..., q_i$, the j^{th} component of the set V_i consisting of only one terminal or a sequence of successive terminals. Clearly, the following inequalities are valid for MSOND(G, K, T)

$$\begin{array}{rll} x^k(\delta(V_{i,j})) & \geq 2 & \text{for all}\, i = 1, ..., p,\ j = 1, ..., q_i \\ -x^k(\delta(s)) & \geq -2 & \text{for all}\, s \in S, \\ x^k(e) & \geq 0 & \text{for all}\, e \in E(S_k) \setminus E(S). \end{array}$$

Remark that there are exactly $p+\sum_{i=1}^{r} q_i - r$ Steiner cut inequalities and $|S|$ disjunction inequalities.

By summing these inequalities, together with (4.8), we obtain

$$2x^k(\delta_{G\setminus S}(V_1, V_2, ..., V_p)) \geq 2((p + \sum_{i=1}^{r} q_i - r) - |S|),$$

and by dividing by 2, we obtain

$$x^k(\delta_{G\setminus S}(V_1, V_2, ..., V_p)) \geq (p + \sum_{i=1}^{r} q_i - r) - |S|,$$

which ends the proof.

□

4.3.5 Generalized disjunction inequalities

In this section, we introduce further valid inequalities which, as inequalities (4.2) and inequalities (4.85), come from the disjunction constraint in the problem.

Proposition 4.26 *Consider a demand $k \in K$. Let $W \subset V$ and $F \subseteq \delta(W)$ such that $|F|$ is odd. Then*

$$x^k(E(W)) + x^k(F) \leq |W| + \lfloor \frac{|F|}{2} \rfloor \tag{4.116}$$

is valid for MSOND(G, K, T).

Proof We prove the validity of inequalities (4.116) for MSOND(G, K, T) using a Chvátal-Gomory procedure.

Clearly, the following inequalities are valid for MSOND(G, K, T)

$$\begin{aligned} x^k(\delta(v_i)) &\leq 2 \quad \text{for all } v_i \in W, \\ x^k(f) &\leq 1 \quad \text{for all } f \in F, \\ -x^k(g) &\leq 0 \quad \text{for all } g \in \delta(W) \setminus F. \end{aligned}$$

By summing these inequalities, we obtain

$$2(x^k(E(W)) + x^k(F)) \leq 2|W| + |F|$$

Now, by dividing by 2 and rounding up the right-hand side, we obtain

$$x^k(E(W)) + x^k(F) \leq |W| + \lfloor \frac{|F|}{2} \rfloor.$$

□

Inequalities (4.116) will be called *generalized disjunction inequalities.*

Similarly to the Steiner F-partition inequalities, inequalities (4.116) enable to cut the fractional solution of Figure 4.11. This solution is given in Figure 4.15.

It is not hard to check that the fractional solution of Figure 4.15 violates an inequality of type (4.116). In fact, by considering $W = \{1, 2, 3\}$ and $F = \{e_1, e_2, e_3\}$, one can see that $x_{e_1} + x_{e_2} + x_{e_3} + x_{e_4} + x_{e_5} + x_{e_6} = 3 + \frac{3}{2} > 4$, where $4 = 3 + 1 = |W| + \lfloor \frac{|F|}{2} \rfloor$.

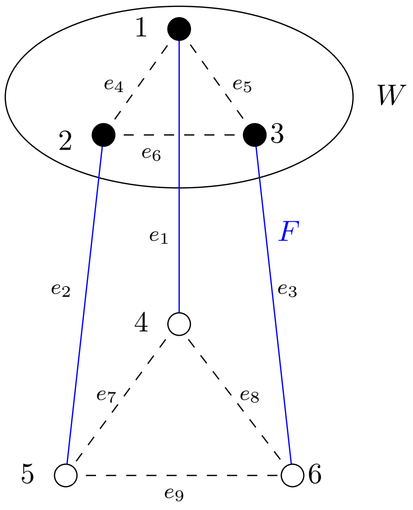


Figure 4.15:

Notice that these inequalities look like the *blossom inequalities* for the TSP which coincide with the 2-*matching inequalities* introduced by Edmonds (1965) in the context of matching problems [45]. And this, is just not strange, because of the tight relationship of our problem with the TSP.

In the following section, we will get more profit from this relationship and propose to our problem an analogue of the well known *Comb inequalities* of the TSP.

4.3.6 Steiner comb inequalities

Comb inequalities have been first discovered by Chvátal [28] in the mid-1970s. After that, Grötschel and Padberg [71] proposed a generalization of these inequalities.

The name of "comb" come from the form of these inequalities. Indeed, the vertices of the graph are partitioned in subsets H and $T_1, ..., T_{2k+1}$, where H is called the *handle* and $T_1, ..., T_{2k+1}$ are the *teeth*.

This type of inequalities have interested many researchers who show that they are a powerful source of cutting planes for some classical problems and particularly the TSP. The efficiency of the combs was first shown Grötschel [65] by finding the optimal tour through 120 German cities,using a very small number of inequalities. Comb inequalities have been also shown effective for survivable network design problems. In [126], Stoer introduces several classes of comb inequalities for 2NCON, 2ECON and the kNCON problems. A deep facial investigation of the proposed inequalities is also held by the author.

Proposition 4.27 *Consider a demand $k \in K$. Consider a family of subsets of V, H called the handle and $T_{th_1}, T_{th_2}, ..., T_{th_p}$ called the teeth such that*

1) $p \geq 3$ and odd,

2) for every two disjoint teeth T_{th_i} and T_{th_j}, $T_{th_i} \cap T_{th_j} = \emptyset$,

3) for each tooth T_{th_j}, $|T_{th_j} \cap T_k| \geq 2$,

4) for each tooth T_{th_j}, $H \cap T_{th_j} \neq \emptyset$, $T_{th_j} \setminus H \neq \emptyset$, and one of the following conditions is satisfied

i) if $(H \cap T_{th_j}) \cap T_k = \emptyset$, then $T_{th_j} \setminus H$ contains at least 2 non successive terminals (or sequences of terminals),

ii) if $(T_{th_j} \setminus H) \cap T_k = \emptyset$, then $H \cap T_{th_j}$ contains at least 2 non successive terminals (or sequences of terminals),

iii) $(H \cap T_{th_j}) \cap T_k \neq \emptyset$ and $(T_{th_j} \setminus H) \cap T_k \neq \emptyset$.

Then

$$x^k(\delta(H)) + \sum_{i=1}^{p} x^k(\delta(T_{th_j})) \geq 3p + 1 \tag{4.117}$$

is valid for MSOND(G, K, T).

Proof The proof of the validity here is in the same spirit as the one given for the general comb inequalities for the TSP [8].

Let us define for each $i = 1, ..., p$ the parameter c_i as follows:

$$c_i = \begin{cases} 1 & \text{if } x^k \text{ contains an edge between } H \cap T_{th_j} \text{ and } T_{th_j} \setminus H \\ 0 & \text{otherwise} \end{cases}$$

By condition 2) the teeth are pairwise disjoint and we have $x^k(\delta(H)) \geq \sum_{i=1}^{p} c_i$. In addition, by definition of parameter c_i, we have $\sum_{i=1}^{p} c_i \leq p$. Since $x^k(\delta(H))$ is even and p is odd, we conclude that:

$$x^k(\delta(H)) \geq 2\sum_{i=1}^{p} c_i - p + 1 \tag{4.118}$$

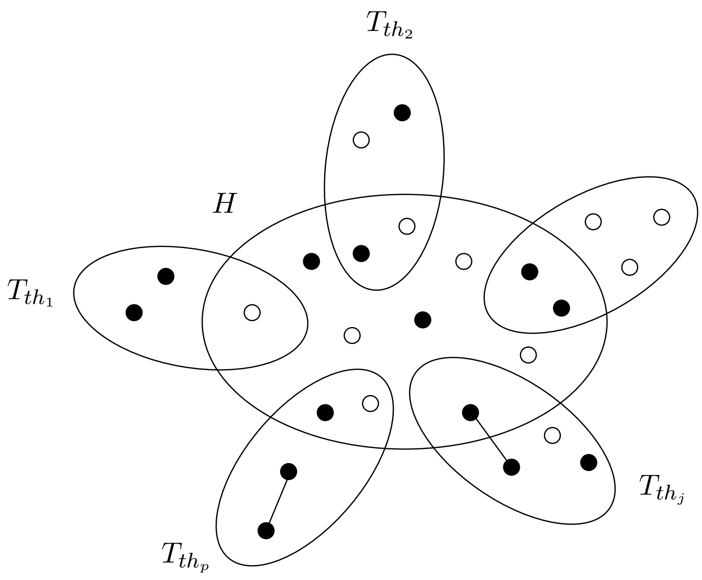


Figure 4.16: Steiner Comb inequalities

Moreover, by conditions 3) and 4), we can write for each tooth T_{th_j}

$$x^k(\delta(T_{th_j})) \geq 4 - 2ci \tag{4.119}$$

By (4.119) for all $i = 1, ..., p$ together with (4.118), we obtain

$$x^k(\delta(H)) + \sum_{i=1}^{p} x^k(\delta(T_{th_j})) \geq 2\sum_{i=1}^{p} c_i - p + 1 + \sum_{i=1}^{p}(4 - 2c_i).$$

Notice that the parameter c_i will disappear since quantity $2\sum_{i=1}^{p} c_i$ will be simplified. This yields

$$x^k(\delta(H)) + \sum_{i=1}^{p} x^k(\delta(T_{th_j})) \geq -p + 1 + \sum_{i=1}^{p} 4 = 3p + 1.$$

□

4.4 Concluding remarks

In this chapter, we have proposed an integer programming formulation for the MSOND problem. We then discussed the associated polytope, described its dimension and studied the facial aspect of the basic constraints of the formulation. We have also identified new classes of valid inequalities for the MSOND(G, K, T) polytope and discussed necessary and sufficient conditions for these inequalities to be facet defining. Using this, we propose to devise a Branch-and-Cut algorithm in the next chapter. The aim of this algorithm is to discuss algorithmic consequences of the results presented in this chapter. Separation procedures of the valid inequalities will be discussed and an extensive computational study will be presented.

Chapter 5

Branch-and-Cut algorithm

Contents

In this chapter, we devise a Branch-and-Cut algorithm for the cut formulation. Our aim is to perform algorithmic applications of the polyhedral results described in the previous chapter and discuss strategic choices made in order to solve the MSOND problem. First, we give an overview of the algorithm. Then, we describe the routines *of separation of some valid inequalities. Overall, the Branch-and-Cut algorithm is used to prove the efficiency of the valid inequalities for a powerful resolution of random as well as realistic instances.*

5.1 Branch-and-Cut algorithm

5.1.1 Description

Since the cut formulation (4.6) is given with a huge number of cut inequalities, we first consider a restricted version of the corresponding linear program. A restricted number of cut inequalities is then generated in the first LP. In our case, we generate only degree inequalities associated with terminals in each reduced graph. Therefore, the initial linear program LP_{ini} that we solve in the first step is given by the inequalities (4.1) written for terminals, the disjunction inequalities (4.2), the linking inequalities (4.3) as well as the trivial inequalities (4.4) and (4.5), that is

$$min \sum_{e\in E} c(e)y_e$$

$$\sum_{e\in\delta_{G^{k,j}}(v)} x_e^k \geq 1 \qquad \text{for all}\, k \in K,\ q_j^k = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k,\ v \in \{w_j^k, w_{j+1}^k\}, \tag{5.1}$$

$$\sum_{e\in\delta(w)} x_e^k \leq 2 \qquad \text{for all}\, w \in V,\ k \in K, \tag{5.2}$$

$$x_e^k \leq y_e \qquad \text{for all}\, e \in E,\ k \in K, \tag{5.3}$$

$$0 \leq x_e^k \qquad \text{for all}\, e \in E,\ k \in K, \tag{5.4}$$

$$y_e \leq 1 \qquad \text{for all}\, e \in E. \tag{5.5}$$

Denote by $(\overline{x}, \overline{y}) \in \mathbb{R}^{K\times E} \times \mathbb{R}^E$ the solution of the cut formulation's linear relaxation (4.6) for the MSOND problem. The obtained solution $(\overline{x}, \overline{y})$ is optimal for the restricted LP if and only if it satisfies all the cut inequalities (4.1). In general, this is not the case. Therefore, violated cut inequalities are added to the restricted LP, by solving a subproblem called *separation problem*. The process is repeated until no more violated inequality is found. The final solution, is hence optimal for the linear relaxation of (4.6). If the solution is integral then it is optimal for the MSOND problem. If not, then we create new subproblems by branching on a fractional variable. The separation routine is then considered at each node of the tree and the process continue. Algorithm 4 gives the main phases of our Branch-and-Cut algorithm.

This algorithm uses some of the valid inequalities described in Chapter 4, whose separations are performed in the following order:

1) Cut inequalities,
2) Steiner cut inequalities,
3) Steiner non-successive terminals inequalities,
4) Steiner F-partition inequalities.

Algorithm 4: Branch-And-Cut Algorithm

```
Data: An undirected graph $G = (V, E)$, a set $K$ of demands, 2 node-disjoint
      paths routing each demand $k \in K$
Result: Optimal solution for the cut formulation (4.6)
1  $LP \leftarrow LP_{ini}$;
2  Solve the linear program $LP$ and denote by $(\overline{x}, \overline{y})$ the optimal solution of LP;
3  if for all $k \in K$,$q \in \mathcal{T}_k$ no (Cut,Steiner cut,Steiner non-successive, Steiner
   F-partition) inequality is violated by $(\overline{x}, \overline{y})$ then
4      go to 8;
5  else
6      Add all possible violated inequalities by $(\overline{x}, \overline{y})$;
7      go to 2;
8  if $(\overline{x}, \overline{y})$ is integer then
9      $(\overline{x}, \overline{y})$ is an optimal solution for MSOND. Stop ;
10 else
11     Create two sub-problems by branching on a fractional variable.
12 forall open sub-problem do
13     go to 2;
14 return the best optimal solution of all the sub-problems.
```

One can here remark that the inequalities to be separated are all global, that is they are valid in the whole Branch-and-Cut tree. In our Branch-and-Cut algorithm, we choose the following strategy of separation. At each separation procedure, we can add more than one violated inequality if there is any. Moreover, when separating the valid inequalities given above, we move to the separation of a new class of inequality only if no more violated inequalities of the current one is detected. We also choose to apply the cutting plane process for all the nodes of the Branch-and-Cut tree in order to get the best possible lower bound, and then limit the number of generated nodes in the tree. Furthermore, violated inequalities are added by sets of 200 or fewer at each time.

In what follows, we describe the separation routines used to separate the inequalities mentioned above. Depending on the class of the valid inequality, we devise exact or heuristic procedures of separation. All the separations are carried out on graphs $G'_k = (V'_k, E'_k)$, $k \in K$, where $(\overline{x}^k_e, e \in E^k)$ are the weights associated with the edges of E'_k. In particular, as some valid inequalities are defined in reduced graphs, they are separated in the graphs $G'_{k,j} = (V'_{k,j}, E'_{k,j})$, $k \in K, q^k_j = (w^k_j, w^k_{j+1}) \in \mathfrak{T}_k$, where $\overline{x}^{k,j}_e$ is the restriction of vector $\overline{x}^k_e$ to the reduced graph $G'_{k,j}$.

Before giving the separation procedures, we first present the test of feasibility of a solution $(x, y) \in \mathbb{R}^{K \times E} \times \mathbb{R}^E$ described in the following section.

5.1.2 Test of feasibility

The cut formulation of the MSOND problem is given with an exponential number of inequalities. In practice, these inequalities are not enumerated and are not all present in the initial LP (LP_{ini}). As a consequence, an optimal solution of LP_{ini}, even if it is integer, is not necessary feasible for the original problem's formulation (4.6). This solution should, in fact, satisfy all the cut inequalities. To check if it is feasible for (4.6), one should solve the separation problem for the basic cut inequalities (4.1), which is detailed in the following section.

5.1.3 Separation of cut inequalities

The problem is to find one or more cut inequality (4.1) that is violated by the current solution $(\overline{x}, \overline{y})$. For each demand $k \in K$ and each section $q^k_j = (w^k_j, w^k_{j+1}) \in \mathfrak{T}_k$, this can be done exactly using the algorithm of Gomory-Hu [61] applied to the graph $G'_{k,j}$ whose edges are weighted by $\overline{x}^{k,j}$. This algorithm gives back the so-called Gomory-Hu tree having the property that between two nodes $s, t \in V'_{k,j}$, the minimum cut separating s and t in the graph $G'_{k,j}$ is nothing but the minimum cut separating s and t in the cut tree. To compute Gomory-Hu tree, we use the efficient implementation of Gusfield [73, 74]. recall that, for a graph $G = (V, E)$, this implementation consists of $|V| - 1$ maximum flow problems in G. Therefore, in our case, we compute $|V'_{k,j}| - 1$ maximum flow problems in $G'_{k,j}$.

Recall that, by the maximum flow - minimum cut theorem (Ford and Fulkerson [50]), the minimum cut problem can be solved in polynomial time. Thus, in our Branch-and-Cut algorithm, for all the problems of minimum cut (and hence maximum flow), we use the algorithm of Goldberg and Tarjan [60], which is one of the most powerful implementations of this problem. This algorithm has a complexity of $\mathcal{O}(m'_j n'_j log\frac{n'^2_j}{m'_j})$, where $m'_j = |E'_{k,j}|$ and $n'_j = |V'_{k,j}|$ are the number of edges and nodes in $G'_{k,j}$, respectively. As the total separation algorithm is carried out for each demand $k \in K$ and each section $q^k_j = (w^k_j, w^k_{j+1}) \in \mathcal{T}_k$, this implies that at most we compute $n' = |V'_k|$ minimum cuts. As a consequence, the whole separation routine of inequalities (4.1) can be done in $\mathcal{O}(m'n'^2plog\frac{n'^2}{m'})$, where $p = |K|$ and n' is an upper bound on the number of terminals (and hence sections) of each demand.

5.1.4 Separation of Steiner cut inequalities

The separation of inequalities (4.71) can be performed using a procedure similar to the one we use to separate cut inequalities (4.1). It is thus an exact separation algorithm based on the implementation of the Gusfield [73, 74] of the algorithm of Gomory-Hu [61], using Goldberg and Tarjan's algorithm [60] to calculate maximum flows. However, the separation of the Steiner cut inequalities is slightly different from the previous one, since we apply the algorithm of Gusfield [73, 74] once for each demand. This implies that in terms of complexity, the separation of the Steiner cut inequalities can be done in $\mathcal{O}(m'n'plog\frac{n'^2}{m'})$.

Moreover, in our computations and in order to improve the lower bound of the root node, we choose to generate, together with inequalities (5.1)- (5.5), degree inequalities for the terminals of each demand. The generated inequalities have the following form,

$$\sum_{e\in\delta(v)} x^k_e \geq 2 \quad \text{for all}\, k \in K, \text{ for all } v \in T_k. \tag{5.6}$$

The computations show that these terminals' degree inequalities help improving the linear relaxation of the problem and hence limit the number of generated nodes of the Branch-and-Cut tree.

5.1.5 Separation of Steiner non-successive terminals inequalities

In this section, we discuss the separation of Steiner non-successive terminals inequalities (4.84). As it was shown in Theorem 4.19, an inequality of type (4.84) defines a facet of MSOND(G, K, T) if and only if the sets V_1 and V_3 are such that $V_1 \cap T_k = \{t_{j-1}\}$ and $V_3 \cap T_k = \{t_{j+1}\}$. As a consequence, in what follows, we restrict ourselves to this case.

Consider a facet-defining configuration of inequality (4.84) as it is shown in Figure 5.1. Figure 5.1 represents a restricted graph in which only terminals t_{j-1}, t_j and t_{j+1} as well as the Steiner nodes S_k are kept. Let us denote by $\overline{V}^{k,j}$ the set of these nodes. In what follows, we maintain the same sets' notations given in Section 4.3.2, namely $F_{j-1} = [V_0, V_1]$, $F_j = [V_0, V_2]$ and $F_{j+1} = [V_0, V_3]$.

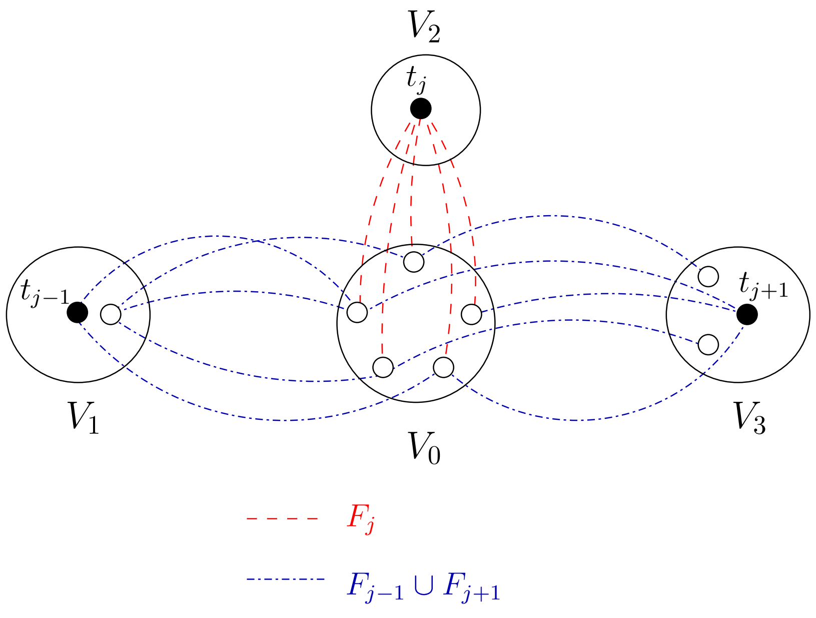


Figure 5.1: Facet-defining configuration of the Steiner non-successive terminals inequalities

Consider the reduced graph $\overline{G}^{k,j} = (\overline{V}^{k,j}, F_{j-1} \cup F_j \cup F_{j+1})$, whose edges are weighted by the corresponding restriction of vector $\overline{x}^k$ (i.e. the fractional solution that we want to separate). The Steiner non-successive terminals inequality corresponding to this configuration can be written as follows

$$\sum_{e \in F_{j-1} \cup F_{j+1}} x_e^k \geq \sum_{e \in F_j} x_e^k. \tag{5.7}$$

The separation problem of inequalities (4.84) in the graph $\overline{G}^{k,j}$ is equivalent to the following optimization problem

$$min \sum_{e \in F_{j-1} \cup F_{j+1}} \overline{x}_e^k - \sum_{e \in F_j} \overline{x}_e^k. \tag{5.8}$$

Remember that $V_0 = S$ (see Section 4.3.2). Let us denote by $p(S) = \sum_{s_i \in S} p_i$, where $p_i = \overline{x}^k_{t_j s_i}$. Also, denote by $\Gamma(S) = \overline{x}^k(\delta(S)) - p(S)$.

It is not hard to see that solving the problem (5.8) is equivalent to

$$min \sum_{S \subseteq S_k} \Gamma(S). \tag{5.9}$$

In the sequel, we will show that problem (5.9) can be solved in polynomial time. To this end, let us first present some useful results.

Definition 5.1 *Given a finite set S, a function $f : 2^S \to \mathbb{R}$ is said to be submodular if the following is true*

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \quad \textit{for all subsets } A, B \subseteq S. \tag{5.10}$$

Examples of submodular functions include cut capacity functions, matroid rank functions and entropy functions. For additional examples of submodular functions and for applications, the reader is referred to [49, 55, 101, 78].

One of the most important result concerning submodular functions is the one related to their minimization. In fact, such a problem can be solved in polynomial time. Grötschel, Lovasz and Schrijver [66] were the first to give a polynomial time and strongly polynomial time algorithms for minimizing a submodular function using the ellipsoid method. Schrijver [122] and Iwata et al. [79] develop independently strongly polynomial time algorithms for minimizing a submodular function. Both algorithms are based on the work of Cunningham [33], who develop a pseudo-polynomial time algorithm for minimizing a submodular function. Recently, Orlin [104] and Iwata et al. [80] propose algorithms that improve the previous best strongly polynomial algorithm's running time.
All the proposed algorithms have a complexity that is for the best around $\mathcal{O}(n^5)$, where n is the number of elements of set S.

Proposition 5.2 Γ *is a submodular function.*

Proof Consider $A \subseteq S$ and $B \subseteq S$. It is not hard to see that

$$p(A) + p(B) = p(A \cup B) + p(A \cap B). \tag{5.11}$$

Moreover, as the cut function is submodular, this implis that

$$\overline{x}^k(\delta(A)) + \overline{x}^k(\delta(B)) \geq \overline{x}^k(\delta(A \cup B)) + \overline{x}^k(\delta(A \cap B)). \tag{5.12}$$

By (5.11) and (5.12), we obtain

$$\Gamma(A) + \Gamma(B) \geq \Gamma(A \cup B) + \Gamma(A \cap B), \tag{5.13}$$

and the result follows. □

From Proposition 5.2 and as it was mentioned above that minimizing submodular functions can be performed in polynomial time, we deduce the following theorem.

Theorem 5.3 *Inequalities* (4.19) *can be separated exactly in polynomial time.*

By Theorem 5.3, we know that we can separate exaclty the Steiner non-successive terminals inequalities in polynomial time. However, as the complexity of the minimization of a submodular function is around $\mathcal{O}(n^5)$, separating exactly inequalities (4.19) would be time consuming. As a consequence, in our Branch-and-Cut algorithm, we will rather use a heuristic separation for Steiner non-successive terminals inequalities. To this end, we propose two heuristics.

The first heuristic is detailed in Algorithm 5 and can be described as follows. Consider a demand $k \in K$. We first go through the current solution (whose incident vector is denoted $(\overline{x}, \overline{y})$). During this step, we mark all the Steiner nodes that have been used to route demand k. Denote by $S_{k,+}$ the set of those nodes (see step 3 of Algorithm 5). We then consider a terminal $t_j \in T_k$ and consider the reduced graph $\overline{G}^{k,j}$ defined above. Recall that $\overline{G}^{k,j} = (\overline{V}^{k,j}, F_{j-1} \cup F_j \cup F_{j+1})$, where $\overline{V}^{k,j} = \{t_{j-1}, t_j, t_{j+1}\} \cup S_k$. In a second step, we fix $V_0 = S = S_{k,+}$ and calculate the value of $\Gamma(S)$. If this value is less than 0, then there exists a violated inequality (see steps 9-11 of Algorithm 5). If no violated inequality is detected, we change the set of Steiner nodes S as follows. We choose one of the most weighted edges of $F_{j-1} \cup F_{j+1}$. Denote by sv this edge, where $s \in V_0$ and $v \in V_1 \cup V_3$. We then remove s from V_0 and put it in $V_1 \cup V_3$ (i.e. we contract edge sv). We hence obtain a new set $S = S \setminus \{s\}$. Given this, we go back to step 8 of the algorithm and calculate the new value of $\Gamma(S)$. The procedure is repeated until no Steiner node remains in S. The overall heuristic requires $|T_k| \times |S_k|$ operations for each demand $k \in K$. As n' is an upper bound for both $|T_k|$ and $|S_k|$ (recall that $n' = |V'_k|$), this implies that the whole heuristic runs in $\mathcal{O}(pn'^2)$, where $p = |K|$.

Algorithm 5: Separation of the SNST inequalities

```
Data: Fractional Solution $(\overline{x}, \overline{y})$
Result: Violated SNST inequalities
1  Let $I \leftarrow \emptyset$ ;
   /* denotes the set of SNST inequalities violated by $(\overline{x}, \overline{y})$ */
2  forall $k \in K$ do
3      $S_{k,+} \leftarrow \emptyset$ ;
       /* the set of marked Steiner nodes, used in the current solution
          to route demand $k$ */
4      forall $t_j \in T_k$ do
5          Let $\overline{G}^{k,j} = (\overline{V}^{k,j}, F_{j-1} \cup F_j \cup F_{j+1})$, where $\overline{V}^{k,j} = \{t_{j-1}, t_j, t_{j+1}\} \cup S_k$;
6          Let $S \leftarrow S_{k,+}$;
7          while $|S| \geq 1$ do
8              Calculate the value of $\Gamma(S)$;
9              if $\Gamma(S) < 0$ then
                   /* there is a violated SNST inequality */
10                 Denote $I_S^{k,j}$ the violated inequality;
11                 $I \leftarrow I \cup I_S^{k,j}$
12             else
13                 Choose one of the most weighted edges of $F_{j-1} \cup F_{j+1}$;
14                 Denote $sv$ this edge;
                   /* where $s \in V_0 = S$ and $v \in V_1 \cup V_3$ */
15                 $S \leftarrow S \setminus \{s\}$;
16                 $F_{j-1} \cup F_{j+1} \leftarrow F_{j-1} \cup F_{j+1} \cup \{s\}$
17 return the detected violated SNST inequalities $I$ ;
   /* $I = \emptyset$ if no violated SNST inequalities are detected */
```

We also devise a second heuristic to separate the Steiner non-successive terminals inequalities. This heuristic is based on the calculation of minimum cuts. Given a demand k and a terminal t_j of T_k, we calculate the minimum cut between t_j and the supernode formed by t_{j-1} and t_{j+1}. Denote this cut by $\delta(W)$, where $t_j \in W$ and $t_{j-1}, t_{j+1} \in \overline{W}$. Based on the calculated cut, we build our partition as follows. $V_0 = W \cap S_k$, $V_1 = t_{j-1}$, $V_2 = t_j$ and $V_3 = t_{j+1} \cup (\overline{W} \cap S_k)$. Once the sets fixed, we calculate the value of $\Gamma(S)$, which is equal to $\overline{x}^k(\delta(W)) - \overline{x}^k(\delta(t_j))$. To calculate the minimum cut we use the procedure described in the previous sections. As the complexity of minimum cut is $\mathcal{O}(m'n'log\frac{n'^2}{m'})$, and we calculate for each demand $k \in K$ exactly $|T_k|$ minimum cuts, the complexity of the heuristic is hence $\mathcal{O}(m'n'^2plog\frac{n'^2}{m'})$.

5.1.6 Separation of Steiner F-partition inequalities

In what follows, we consider the separation of the Steiner F-partition inequalities (4.96). To this end, we propose a heuristic approach to detect Steiner F-partition inequalities violated by $(\overline{x}, \overline{y})$. Our heuristic consists for a demand k in determining an odd cycle passing through terminals of T_k and whose edges have fractional values. In particular, this is the case when the cycle visits all the terminals of the considered demand, since there are no possible edges between non-successive terminals (mainly after separating the valid inequalities mentioned above). The nodes of the cycle will then define the sets $V_1, \ldots, V_p$ and hence V_0 will contain only Steiner nodes of S_k. Moreover, the set F will be chosen in $\delta(V_0)$ such that $|F|$ is odd. These observations lead us to think about the heuristic of separation given by Algorithm 6, and which can be described as follows.

Consider a demand $k \in K$. We first start by marking the Steiner nodes that have been used to route demand k in solution $(\overline{x}, \overline{y})$. We then delete the Steiner nodes $S_k \setminus S_{k,+}$. In the resulting graph, we contract all the edges between successive terminals t_j, t_{j+1} of T_k having a value equal to 1 (i.e. $\overline{x}^k_{t_j t_{j+1}} = 1$). We denote by $T'_k = \{t'_1, \ldots, t'_p\}$ the set of the remaining terminals. Then, we construct the partition $\Pi = (V_0, V_1, \ldots, V_p)$ by considering the following sets, $V_0 = S_{k,+}$ and $V_i = t'_i$, $i = 1, \ldots, p$. Once the partition Π is built, we choose the set F. The edges of F are chosen among those of $\delta(V_0)$ having the highest values and such that $|F|$ is odd (if such an edge set F is empty, we move to the next demand). We then calculate the value $\overline{x}^k(\delta(\Pi) \setminus F)$ and compare it to the quantity $p - \left\lfloor \frac{|F|}{2} \right\rfloor$ to check if the Steiner F-partition induced by Π and F is violated or not. The heuristic here described, runs in a linear time for each demand. The overall complexity is then $\mathcal{O}(pn')$.

5.1.7 Implementation's features

During the separation procedure, to efficiently deal with the violated inequalities that are added, we create particular data structures called *pools* whose size increases dynamically. All the generated inequalities are dynamic and stored in a specific pool, that is to say that they are removed from the current linear program if they are not active. At each iteration, the separation procedure begins first by detecting violated inequalities in the pool. If no such inequality exits, then we carry out our separation procedure on the valid inequalities in the order given above.

Algorithm 6: Separation of the SFP inequalities

```
   Data: Fractional Solution $(\overline{x}, \overline{y})$
   Result: Violated SFP inequalities
 1 Let $J \leftarrow \emptyset$ ;
   /* denotes the set of SFP inequalities violated by $(\overline{x}, \overline{y})$ */
 2 forall $k \in K$ do
 3     $S_{k,+} \leftarrow \emptyset$ ;
       /* the set of marked Steiner nodes, used in the current solution
          to route demand $k$ */
 4     Delete the Steiner nodes of $S_k \setminus S_{k,+}$;
 5     In the resulting graph, contract all the edges $t_j t_{j+1}$ between two successive
       terminals of $T_k$ such that $\overline{x}^k_{t_j t_{j+1}} = 1$;
 6     Denote $T'_k = \{t'_1, \dots, t'_p\}$ the resulting set of terminals.;
 7     $V_0 \leftarrow S_{k,+}$;
       /* constitute the sets $V_1, \dots, V_p$ as follows */
 8     $V_i \leftarrow \{t'_i\}$, $i = 1, \dots, p$ Choose the set of edges $F \subseteq \delta(V_0)$ such that $|F|$ is
       odd and having values greater than $\frac{1}{2}$;
 9     Given $\Pi = (V_0, V_1, \dots, V_p)$ and $F \subseteq \delta(V_0)$, calculate $\overline{x}^k(\delta(\Pi) \setminus F)$;
10     if $\overline{x}^k(\delta(\Pi) \setminus F) < p - \lfloor \frac{|F|}{2} \rfloor$ then
           /* there is a violated SFP inequality */
11         Denote $J^k_{\pi,F}$ the violated inequality;
12         $J \leftarrow J \cup J^k_{\pi,F}$
13 return the detected violated SFP inequalities $J$ ;
   /* $J = \emptyset$ if no violated SNST inequalities are detected */
```

5.1.8 Branching strategy

Let $(\mathcal{P})$ denote the linear program of a given node in the Branch-and-Cut tree. Suppose that the optimal solution of the linear relaxation of $(\mathcal{P})$ is fractional. Denote by $(\overline{x}, \overline{y})$ this fractional solution. The branching phase consists in choosing a fractional variable $\overline{x}^k_e$ (resp. $\overline{y}_e$), $e \in E, k \in K$, and then create two subproblems $(\mathcal{P}_1)$ and $(\mathcal{P}_2)$ by adding respectively the constraints $x^k_e \leq \lfloor \overline{x}^k_e \rfloor$ and $x^k_e \geq \lceil \overline{x}^k_e \rceil$ (resp. $y_e \leq \lfloor \overline{y}_e \rfloor$ and $xy_e \geq \lceil \overline{y}_e \rceil$). As the decision variables for the MSOND problem are binary, this reduces to fix the variable x^k_e (resp. y_e) either to 0 or to 1.

There are several strategies used to select the fractional variable on which we choose to branch. In our case, we have chosen the strategy introduced by Padberg and Rinaldi [107] for the Symmetric Travelling Salesman Problem. This strategy consists in choosing the most fractional variable, that is the fractional variable which is the nearest to 0.5. If there exist many variables having the same fractional value, and satisfying this condition, then we choose the most weighted one in the objective function.

5.2 Computational study

5.2.1 Computations' context

Before giving the experimental results, we first present the tools that we have used for the implementation. The Branch-and-Cut algorithm described in the previous section has been implemented in C++ using ABACUS 3.2 (A Branch-And-CUt System) [1] to manage the Branch-and-Cut framework. To solve the linear programs, ABACUS uses CPLEX 12.5 [2]. The Branch-and-Cut algorithm was tested on Bi-Xeon quad-core E5507 2.27GHz with 8Go of RAM, running under Linux.

The tests have been done on two types of instances: random instances and realistic instances described in the following section.

5.2.2 Description of instances

Experimentations for MSOND problem are based on two types of instances: random and realistic ones. Both family of instances is euclidian. They are characterized by:

- the number of nodes V in the optical layer (graph G);
- the number of nodes V' in the logical layer (graph G'). This number constitutes the maximum number of node in G that can be terminals;
- the number K of demands.

5.2.2.1 Random instances

These instances are generated from the TSP-Library [3]. We particularly generate our instances based on data from $a280$, $bier127$, $eil101$, $lin105$ and $tsp225$. According to the size of the instance, we choose the first V nodes of the basic TSP instance. Then, we choose among these nodes, the V' nodes corresponding to the possible terminals. We then constitutes the K demands as follows. Based on the number of the demands, the choice of the origins and destinations of these demands is completely random. To generate after the two paths routing the demands, we first begin by choosing the number of terminals of the first path and after we randomly generate this number from the nodes of the graph (except the origin and destination). We do the same for the second path. We choose the number of terminals and generate these ones from the nodes that have not been yet used in the first path. Here the random aspect is first ensured by the number of terminals but also by the choice of terminals themselves. For each size of the instance, we generate five examples of instances based on the previously chosen TSP basic instances.

5.2.2.2 Realistic instances

The realistic instances are instances that we generate so as to be the nearest to the real instances. We generate two kinds of instances described in what follows.

Gravity model instances

These instances are based on geographic data taken from a web data base [4]. We choose V as the most populated cities and among them the most populated V' cities as well. In a second step, we calculate for all the possible demands (all possible combinations of pairs of nodes V') a quantity indicating its volume. The volume of a demand between two cities is calculated using a gravity model [23] that uses the distance between the considered cities and the population of each city. The gravity model is based on spatial interactions. its general formula for a demand $k = (o_k, d_k, v_k) \in K$ is $\frac{P_{o_k}^{\alpha} P_{d_k}^{\beta}}{D_{o_k,d_k}}$, where P_{o_k}, P_{d_k} are the populations of the origin and destination cities, respectively. D_{o_k,d_k} is the distance between the two cities. We also fix $\alpha = 1.2$ and $\beta = 0.8$. We then choose among the calculated demands the K most important ones. After this step, to generate the two paths, we forbid the direct edge between the origin and destination as well as some edges of the graph. We then calculate two shortest paths routing each demand to stay the nearest to reality.

SNDlib-based instances

These instances are realistic and based on the SNDlib [5]. We particularly choose to test on the following basic instances: dfn-bwin, nobel-us, newyork, geant, cost266, pioro40 and germany50. For these instances, we choose to set $V = V'$, equal to the original number of nodes of the instances. For the demands, we choose among the demands of the original instances of SND the K most important demands (according to the quantity of the demands). After this step, to generate the two paths, we forbid the direct edge between the origin and destination as well as some edges of the graph to avoid generating trivial instances. We then calculate two shortest paths routing each demand to stay the nearest to reality.

5.2.3 Experimental results

We have tested the Branch-and-Cut algorithm on the instances described in the previous sections. We have fixed the maximum CPU time to 5 hours. The results are reported in the tables that will be presented in the sequel. The entries of the various tables are the following:

Instance	:	name of the instance (for realistic instances);
V	:	number of node in graph G;
V'	:	number of node in graph G' (for random instances);
K	:	number of demands;
Term	:	average number of terminals;
Nsub	:	number of subproblems (nodes);
Nsub-basic	:	number of subproblems for the basic formulation;
Ncut	:	number of generated cut inequalities;
NScut	:	number of generated Steiner cut inequalities;
NSNST	:	number of generated Steiner non-successive; terminals inequalities;
NSFP	:	number of generated Steiner F-partition inequalities;
Gap(%)	:	the relative error between the best upper bound (the optimal solution if the problem has been solved to optimality) and the lower bound obtained at the root;
Gap-basic(%)	:	the gap of the basic formulation;
Opt	:	number of instances solved to optimality (for random instances);
TT	:	total CPU time (in seconds for random instances and hh:mm:ss for realistic ones);
TT-basic	:	total CPU time (in seconds).

5.2.3.1 Efficiency of the valid inequalities

Before giving the series of experimentations, we propose, in this section, to evaluate the impact of the valid inequalities that we separate in the Branch-and-Cut algorithm. Consider, for this purpose, the instances of the TSPlib [?] previously described, and let show the results we obtain by computing the algorithm first with the basic formulation and then considering the valid inequalities. The results are reported in Table 5.1. In Table 5.1, we present the results obtained for instances having up to 12 nodes in G and for which the demands range from 6 to 10. It appears from this table that the formulation with valid inequalities performs better than the basic one for all the instances. In fact, from Table 5.1, we notice that the number of the Branch-and-Cut tree's nodes for the basic formulation is more important than the one of the formulation with valid inequalities. See, for example, instance $(a, 12, 10, 8)$ for which the Branch-and-Cut tree explored 8593 nodes with the basic formulation and only 589 when adding valid inequalities. For the same instance, the gap and the total time of execution were much more important using the basic formulation. And this remark can be generalized for all the tested instances, for which we can clearly note that the gap and the total time of execution were always better with the formulation using the valid inequalities. Notice also that, using the basic formulation, some of the instances like $(tsp, 8, 6, 10)$ have not been solve to optimality within 3 hours. However, the optimum for the same instance have been found within some seconds when adding the valid inequalities.

All these observations lead us to conclude about the importance of the added valid inequalities for a better resolution of the MSOND problem. In fact, these inequalities allow a significant improvement of the number of the Branch-and-Cut tree's nodes the gap and the total time of execution.

In the next sections, we will get profit from this to efficiently solve random and realistic instances.

5.2.3.2 Random instances

Our first series of experiments concerns random instances. The instances we have considered have graphs with 6 up to 20 nodes in G and 4 up to 18 nodes in G'. The number of demands for each size of graph ranges from 2 to 10 with an average number of terminals varying between 3 and 6.50. For each triplet (V, V', K), we tested over 5 TSPlib-based instances previously described. The results are reported in Table 5.2. It appears from Table 5.2 that, in average (i.e. $Opt = 5/5$), 24 over 35 groups of instances have been solved to optimality within the time limit. In addition, among the remaining groups of instances, only 6 groups did not reach any optimal solution over the 5 tested instances. Moreover, 13 over 24 of the groups solved to optimality have a gap that does not exceed 3%. For the other groups of instances, this value reaches at most 6.49% for the group of instances having 12 nodes in G and 10 demands.

Table 5.2 shows also that the difficulty of solving an instance depends not only on its size (i.e. size of the graph and number of demands), but also on the average number of its terminals. Remark for example that the group of instances $(14, 12, 2)$ have a gap less than the one obtained for the group of instances $(12, 10, 2)$. In fact, the former has an average number of terminals equal to 6.50, whereas the same number is equal to only 4.50 for the latter.

Note also, that for the majority of group of instances we generate a significant number of cut, Steiner cut and Steiner non-successive terminals inequalities. This means that these inequalities are helpful for the random instances. However, only small numbers of the Steiner F-partition inequalities have been separated. This can be explained by the structure of random instances. In fact, from experimentations we have noticed that these inequalities are as more efficient as the number of terminals is low.

Instance	V	V'	K	Nsub-basic	Gap-basic(%)	TT-basic(s)	Nodes-2	Gap(%)	TT(s)
a	8	6	6	21	7.50	0.69	11	4.31	0.58
bier	8	6	6	43	10.03	1.40	37	6.82	0.63
eil	8	6	6	7	3.34	0.32	1	0.00	0.51
lin	8	6	6	5	1.08	0.36	1	0.00	0.5
tsp	8	6	6	17	3.31	0.53	13	1.65	0.64
a	8	6	8	45	7.85	257.91	19	3.01	30.76
bier	8	6	8	25	8.38	3578.49	3	0.49	15.12
eil	8	6	8	3	0.26	35.59	1	0.00	19.99
lin	8	6	8	19	2.38	50.71	3	0.27	16.69
tsp	8	6	8	17	3.04	16.85	11	1.50	14.70
a	8	6	10	17	5.18	120.81	3	1.73	1.22
bier	8	6	10	19	5.39	3063.59	15	2.23	0.96
eil	8	6	10	7	1.90	2533.11	7	0.72	0.61
lin	8	6	10	17	*5.86*	10800.00	5	1.64	1.10
tsp	8	6	10	21	*5.25*	10800.00	15	4.20	1.80
a	10	8	6	161	9.89	7.60	41	5.95	2.16
bier	10	8	6	23	4.18	1.27	7	1.09	4.16
eil	10	8	6	37	4.80	1.61	23	2.91	0.64
lin	10	8	6	133	11.66	5.30	3	0.70	0.49
tsp	10	8	6	209	7.26	9.70	49	5.09	2.88
a	10	8	8	29	7.53	7.14	21	5.16	3.15
bier	10	8	8	17	4.71	7.18	7	2.60	1.8
eil	10	8	8	107	7.37	24.75	23	4.67	0.54
lin	10	8	8	13	5.22	13.40	1	0.00	0.82
tsp	10	8	8	19	5.36	10.76	23	3.20	2.16
a	10	8	10	79	8.54	7.10	25	4.38	0.88
bier	10	8	10	83	7.16	6.82	47	6.11	1.87
eil	10	8	10	43	7.37	6.42	21	5.85	0.92
lin	10	8	10	43	8.38	3.63	9	2.52	1.16
tsp	10	8	10	163	11.03	15.88	25	7.78	3.12
a	12	10	6	5877	13.08	785.59	225	6.16	15.26
bier	12	10	6	937	11.63	98.76	153	8.59	3.39
eil	12	10	6	259	7.15	20.39	33	4.11	3.6
lin	12	10	6	2609	14.24	290.74	55	5.01	2.1
tsp	12	10	6	813	*10.53*	10800.00	41	3.48	22.37
a	12	10	8	8593	14.58	2333.51	589	7.52	3.55
bier	12	10	8	259	8.59	50.82	85	6.47	2.35
eil	12	10	8	1729	10.92	365.69	207	8.64	4.77
lin	12	10	8	1117	15.62	259.12	113	5.82	0.84
tsp	12	10	8	161	7.94	33.13	19	3.35	5.68
a	12	10	10	3899	12.97	1218.84	153	6.41	3.92
bier	12	10	10	1479	11.63	398.21	161	8.37	7.23
eil	12	10	10	489	8.22	128.49	59	6.05	4.58
lin	12	10	10	1845	*13.56*	10800.00	181	7.10	1.78
tsp	12	10	10	457	*11.06*	10800.00	83	7.64	5.52

Table 5.1: The impact of valid inequalities

5.2.3.3 Realistic instances

Our second series of experiments concerns realistic SNDlib-based instances. The tested instances have a graph whose number of nodes varies from 10 (instances dfn-bwin) to 50 (instances germany), and a number of demands ranging from 2 to 45 at most. Overall, we tested 92 instances and reported the results in Table 5.3 and Table 5.4. It appears from Table 5.3 and Table 5.4 that 72 over 92 of the instances have been solved to optimality within the time limit of execution. The remaining instances, often having more than 30 demands and/or more than 40 nodes, have not reached the optimal value within 5 hours. Moreover, for all the instances, except the two last instances of newyork, the gap does not exceed 6%. We also notice that, 30 instances have been solved at the root node, which proves the efficiency of the Branch-and-Cut algorithm to solve realistic instances in comparison with the random ones. In addition, we remark that the CPU time for the solved instances is relatively small. In fact, 62 instances have been solved to optimality in at most 10 minutes.

Like the random instances, the difficulty of solving the realistic instances does not depend only on its seize (i.e. the number of nodes of the graph and the number of demands) but also on the average number of terminals. However, these only can not explain the evolution of resolution for some instances. Look for example to the instances newyork, in particular (newyork,16,16) and (newyork,16,18). Note that, in spite of being smaller in terms of graph, number of demands and terminals, the former took almost 5 hours to be solved to the optimality, when the latter has reached the optimum within only 10 minutes. This strange behaviour, is in reality caused by some conflicts that can be created between the demands, related mainly to the order between terminals. In fact, moving from instance (newyork,16,14) to (newyork,16,16), by adding only two demands, have brutally augmented the difficulty of resolution. Notice, however, that this difficulty disappears strangely with instance (newyork,16,18), which means that adding demands could sometimes make the problem easier to solve. A deeper analysis of a similar situation encountered for the Branch-and-Price algorithm can be found in Chapter 6.

5.2.4 A French instance

In this section, we present the result that we have obtained for a realistic french instance, generated along the gravity model previously described. The instance consists of 30 nodes representing some french cities for which we look for a routing of 20 demands.

The instance have been solved by the Branch-and-Cut algorithm within 20 minutes and the optimal solution that we have obtained is given in Figure 5.2.

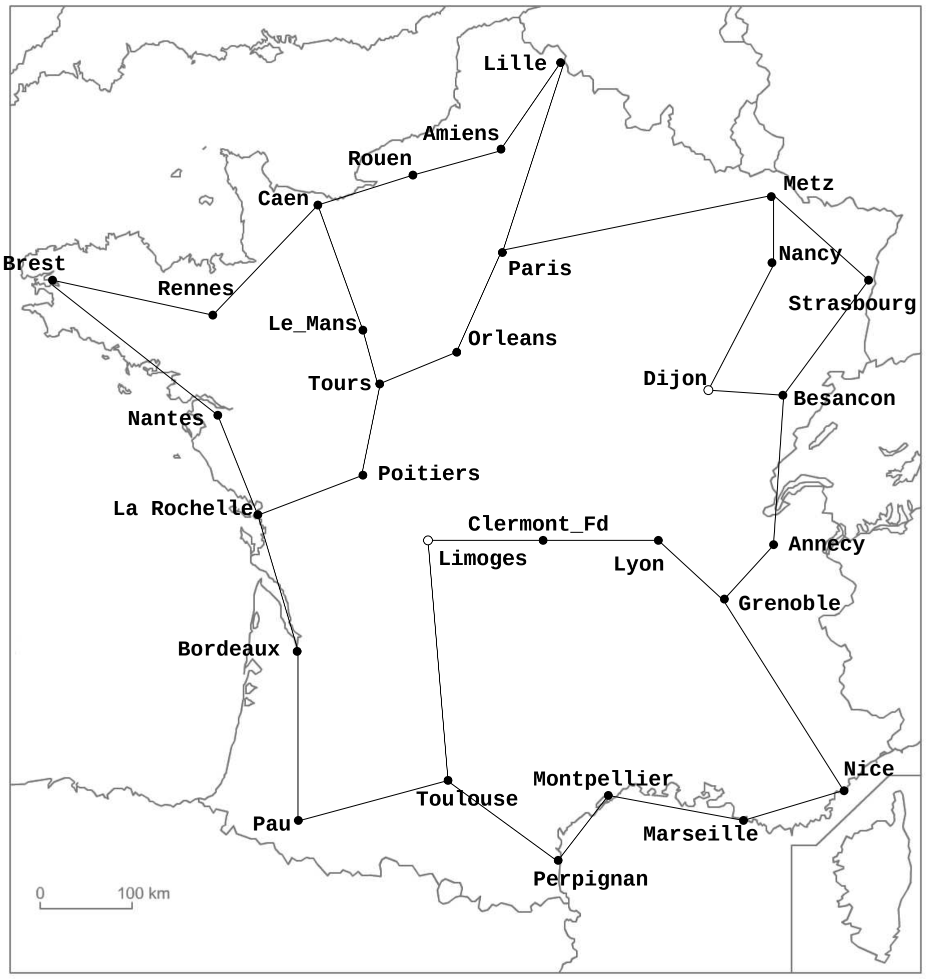


Figure 5.2: Solution for a french instance with 30 nodes and 20 demands

5.3 Concluding remarks

In this chapter, we have devised a Branch-and-Cut algorithm to solve the cut formulation introduced in Chapter 4. We have first presented the different steps of the algorithm and discussed the separation problems associated with some valid inequalities. In particular, we have proposed exact algorithms of separation for both the section cut inequalities (4.1) and the Steiner cut inequalities (4.71). We then proved that the separation problem of the Steiner non-successive terminals inequalities (4.84) reduces to the minimization of a submodular function and hence can be done in polynomial time. For convenience, we have chosen to separate them with a heuristic procedure. We have also proposed a heuristic to separate the Steiner F-partition inequalities (4.96).

Based on this, we have then tested the Branch-and-Cut algorithm on random and realistic instances. The computational results have shown the efficiency of the valid inequalities to improve the lower bound and reduce the time of execution. It has been also shown that the difficulty of an instance does not depend only on its size (i.e. the size of the graph, the number of demands and the average number of terminals), but that it is highly correlated to the order constraints of the demands. In fact, we have proved that an instances that presents conflicting order constraints between its demand is harder to solve than an instance without conflicts. The experiments have also shown that the Branch-and-Cut algorithm performs better for realistic instances with respect to random ones.

V	V'	K	Term	Nsub	Ncut	NScut	NSNST	NSFP	Gap	Opt	TT
6	4	2	3.00	1.0	16.6	10.8	0.0	0.0	0.00	5/5	0.17
6	4	4	3.00	1.0	30.8	21.6	0.0	0.0	0.00	5/5	0.49
6	4	6	3.00	1.0	46.4	32.8	0.0	0.0	0.00	5/5	0.18
8	6	2	4.50	1.0	19.6	9.0	31.0	0.0	0.00	5/5	0.72
8	6	4	4.50	15.4	49.2	84.0	62.4	0.2	3.35	5/5	0.90
8	6	6	4.17	17.8	81.8	138.6	78.0	0.8	2.45	5/5	0.66
8	6	8	4.12	21.8	120.8	227.4	110.8	0.2	3.38	5/5	19.45
8	6	10	4.10	16.2	130.4	249.4	125.2	1.8	4.72	5/5	1.14
10	8	2	4.50	13.4	30.2	23.4	49.0	0.0	1.44	5/5	0.47
10	8	4	4.25	36.6	77.8	172.2	74.6	0.0	2.81	5/5	2.27
10	8	6	4.67	112.6	214.6	751.6	123.6	0.6	3.56	5/5	5.10
10	8	8	4.75	37.0	191.2	551.6	144.2	0.4	3.04	5/5	12.65
10	8	10	5.10	82.2	314.4	999.0	189.0	1.0	4.53	5/5	7.97
12	10	2	6.50	173.0	114.0	317.0	71.8	0.0	3.04	5/5	2.39
12	10	4	5.50	447.8	293.0	187.6	138.0	0.0	3.46	5/5	39.64
12	10	6	5.50	702.0	881.0	658.4	205.4	0.0	4.33	5/5	252.13
12	10	8	5.62	371.8	1200.2	471.4	277.2	0.6	5.53	5/5	608.45
12	10	10	5.70	633.8	1650.2	663.8	347.8	0.8	6.49	5/5	476.78
14	12	2	4.50	194.0	77.7	158.6	85.6	0.0	1.04	5/5	6.22
14	12	4	4.00	805.0	1374.0	804.4	125.0	0.0	2.19	5/5	9045.32
14	12	6	4.50	752.4	1933.2	417.8	218.4	0.0	-	4/5	6133.47
14	12	8	4.87	1251.2	561.4	673.4	306.0	0.0	-	1/5	13000.47
14	12	10	4.90	976.7	527.2	864.0	394.2	0.0	-	3/5	10413.94
16	14	2	4.50	37.0	83.2	111.8	103.4	0.0	2.76	5/5	2.96
16	14	4	4.50	227.4	1257.2	208.0	163.8	0.0	3.39	5/5	2417.54
16	14	6	4.83	809.6	3632.0	622.6	278.6	0.0	-	0/5	18000.00
16	14	8	4.87	661.6	863.4	1127.4	386.0	0.0	-	0/5	18000.00
18	16	2	5.00	697.8	324.8	658.4	128.4	0.0	2.90	5/5	70.94
18	16	4	4.75	305.8	1243.2	1053.4	160.0	0.0	-	3/5	8618.60
18	16	6	4.83	362.6	702.8	2938.2	277.2	0.0	-	0/5	18000.00
18	16	8	5.00	655.7	883.2	1056.6	357.0	0.0	-	0/5	18000.00
20	18	2	4.50	930.0	677.7	883.0	139.6	1.0	4.14	5/5	464.21
20	18	4	4.25	162.0	3673.2	1737.8	214.2	0.0	-	2/5	15346.05
20	18	6	4.83	291.3	2175.0	1236.4	382.0	0.0	-	0/5	18000.00
20	18	8	5.12	904.3	2942.6	1805.6	542.2	0.0	-	0/5	18000.00

Table 5.2: Branch-and-Cut results for random instances

Instance	V	K	Term	Nsub	Ncut	NScut	NSNST	NSFP	Gap	TT
dfn-bwin	10	2	3.50	1	15	7	0	0	0.00	0:00:00.21
dfn-bwin	10	4	3.26	1	29	13	0	0	0.00	0:00:00.28
dfn-bwin	10	6	3.15	1	53	19	0	0	0.00	0:00:00.32
dfn-bwin	10	8	3.50	1	69	28	37	0	0.00	0:00:00.37
dfn-bwin	10	10	3.40	1	94	34	15	0	0.00	0:00:00.46
dfn-bwin	10	12	3.34	1	153	40	73	0	0.00	0:00:00.58
dfn-bwin	10	14	3.28	1	178	46	59	0	0.00	0:00:00.74
dfn-bwin	10	16	3.32	1	240	59	128	0	0.00	0:00:01.35
dfn-bwin	10	18	3.38	19	415	707	145	0	3.98	0:00:08.66
dfn-bwin	10	20	3.45	47	541	1609	193	4	2.21	0:00:41.63
dfn-bwin	10	25	3.48	367	142	403	265	0	3.77	0:05:59.12
dfn-bwin	10	30	3.50	143	1108	761	337	3	3.83	0:03:13.75
dfn-bwin	10	40	3.52	917	451	749	458	8	4.48	4:12:43.08
dfn-bwin	10	45	3.55	581	906	502	531	17	3.36	1:04:33.10
nobel-us	14	2	3.50	1	14	7	40	0	0.00	0:00:00.25
nobel-us	14	4	4.00	1	33	16	56	0	0.00	0:00:00.36
nobel-us	14	6	3.66	15	117	134	76	1	2.39	0:00:02.82
nobel-us	14	8	3.50	15	146	210	125	0	3.38	0:00:04.96
nobel-us	14	10	3.40	15	183	254	98	0	4.70	0:00:08.36
nobel-us	14	12	3.50	17	202	330	205	5	3.74	0:00:08.44
nobel-us	14	14	3.64	19	243	653	253	3	4.54	0:00:22.52
nobel-us	14	16	3.62	71	377	902	293	3	5.92	0:01:21.52
nobel-us	14	18	3.66	31	818	534	373	1	3.87	0:00:51.65
nobel-us	14	20	3.75	63	874	853	458	0	4.18	0:02:13.26
nobel-us	14	25	3.76	5	533	754	618	2	1.56	0:00:24.28
nobel-us	14	30	3.70	525	785	975	698	0	5.79	0:52:36.75
nobel-us	14	40	3.62	447	953	879	858	7	5.97	4:38:19.42
newyork	16	2	3.00	1	22	6	0	0	0.00	0:00:00.28
newyork	16	4	3.00	1	34	12	0	0	0.00	0:00:00.34
newyork	16	6	3.17	1	89	19	18	0	0.00	0:00:00.53
newyork	16	8	3.25	859	1381	526	96	3	1.80	0:30:50.13
newyork	16	10	3.20	363	422	474	74	4	2.18	0:29:33.83
newyork	16	12	3.16	325	609	540	96	6	3.81	0:42:55.89
newyork	16	14	3.35	823	1305	685	199	1	3.42	0:34:31.09
newyork	16	16	3.37	2953	2322	892	345	13	5.32	4:42:27.43
newyork	16	18	3.44	187	174	322	302	5	3.19	0:10:24.59
newyork	16	20	3.50	503	258	878	398	7	5.76	1:16:00.66
newyork	16	25	3.56	451	297	411	590	2	4.27	1:05:44.96
newyork	16	30	3.63	1089	309	432	746	10	*4.26*	5:00:00.00
newyork	16	40	3.60	779	995	1120	986	8	*7.81*	5:00:00.00
newyork	16	45	3.55	1092	357	1166	1034	17	*7.14*	5:00:00.00
geant	22	2	3.50	1	15	7	16	0	0.00	0:00:00.36
geant	22	4	3.75	1	167	15	56	0	0.00	0:00:01.38
geant	22	6	3.67	1	351	22	73	0	0.00	0:00:02.42
geant	22	8	3.50	3	829	252	157	0	0.01	0:00:16.52
geant	22	10	3.40	3	198	206	229	0	0.01	0:00:19.93

Table 5.3: Branch-and-Cut results for realistic instances (1)

Instance	V	K	Term	Nsub	Ncut	NScut	NSNST	NSFP	Gap	TT
geant	22	12	3.33	1	837	164	264	0	0.00	0:00:14.63
geant	22	14	3.43	15	884	722	373	0	0.40	0:01:37.37
geant	22	16	3.50	31	886	498	517	0	0.31	0:02:50.41
geant	22	18	3.56	91	218	384	531	0	1.56	0:25:27.80
geant	22	20	3.50	101	482	726	602	0	1.56	0:33:20.98
geant	22	25	3.44	89	115	364	674	0	1.45	0:51:01.46
geant	22	30	3.53	116	624	1260	916	4	*5.83*	5:00:00.00
geant	22	40	3.55	68	752	1293	289	2	*4.70*	5:00:00.00
geant	22	45	3.60	73	951	1182	531	7	*5.91*	5:00:00.00
cost266	37	2	3.50	1	18	7	28	0	0.00	0:00:01.04
cost266	37	4	3.25	1	30	13	37	0	0.00	0:00:03.98
cost266	37	6	3.17	1	48	19	66	0	0.00	0:00:05.34
cost266	37	8	3.37	1	69	27	58	2	0.00	0:00:11.46
cost266	37	10	3.30	1	132	61	396	0	0.00	0:00:22.59
cost266	37	12	3.33	1	138	40	258	0	0.00	0:00:24.12
cost266	37	14	3.35	1	296	47	660	0	0.00	0:00:34.78
cost266	37	16	3.37	1	201	54	792	0	0.00	0:00:32.52
cost266	37	18	3.33	3	861	230	628	1	0.03	0:03:11.10
cost266	37	20	3.35	3	862	251	924	0	0.03	0:04:04.78
cost266	37	25	3.44	106	556	778	1452	9	*2.67*	5:00:00.00
cost266	37	30	3.57	62	151	781	1034	14	*3.47*	5:00:00.00
cost266	37	40	3.60	20	357	856	1854	3	*4.81*	5:00:00.00
cost266	37	45	3.53	23	244	741	1231	1	*3.92*	5:00:00.00
pioro40	40	2	3.50	1	260	55	144	0	0.00	0:00:11.01
pioro40	40	4	3.75	17	155	375	432	0	1.73	0:04:07.92
pioro40	40	6	3.50	7	175	393	432	0	1.94	0:03:57.54
pioro40	40	8	3.50	5	307	270	576	0	0.72	0:06:45.67
pioro40	40	10	3.50	25	370	1111	720	0	1.40	0:19:54.46
pioro40	40	12	3.50	125	326	596	864	0	1.58	3:32:44.09
pioro40	40	16	3.53	207	265	662	1039	1	*2.06*	5:00:00.00
pioro40	40	18	3.53	97	249	617	1183	1	*1.85*	5:00:00.00
pioro40	40	20	3.45	79	897	636	1355	1	*1.85*	5:00:00.00
pioro40	40	25	3.56	26	674	825	903	1	*3.38*	5:00:00.00
pioro40	40	30	3.57	33	756	790	822	2	*4.17*	5:00:00.00
pioro40	40	40	3.57	6	910	601	973	3	*5.83*	5:00:00.00
germany50	50	2	3.50	1	14	7	184	0	0.00	0:00:02.41
germany50	50	4	3.75	1	404	27	552	0	0.00	0:01:07.01
germany50	50	6	3.67	7	221	176	736	1	0.21	0:10:51.68
germany50	50	8	3.65	59	159	413	920	0	2.66	0:37:55.66
germany50	50	10	3.60	57	300	556	1104	1	2.35	1:07:04.18
germany50	50	12	3.58	159	331	1729	1288	8	2.24	3:09:31.40
germany50	50	14	3.64	51	275	1401	656	5	2.21	1:54:45.51
germany50	50	16	3.68	72	463	697	1024	16	*2.74*	5:00:00.00
germany50	50	18	3.62	57	524	467	1024	1	*2.39*	5:00:00.00
germany50	50	20	3.60	32	596	476	1208	10	*2.72*	5:00:00.00
germany50	50	25	3.64	15	802	915	944	21	*4.65*	5:00:00.00

Table 5.4: Branch-and-Cut results for realistic instances (2)

Chapter 6

Path formulation and Branch-and-Price algorithm

Contents

In this chapter, we propose to formulate the MSOND problem with a polynomial number of constraints, yet an exponential number of variables. First, we present an integer programming formulation in terms of path variables. We start from an original section *formulation and deduce by a Dantzig-Wolfe decomposition a path formulation. Second, we compare the path formulation to the cut formulation introduced in Chapter 4, and prove that the former has a tighter linear relaxation. We then devise a Branch-and-Price algorithm to solve the path formulation. We particularly describe the way to find an initial feasible solution, discuss the pricing problem and propose a suitable branching rule for the Branch-and-Price algorithm. We also present a primal heuristic used to prune some uninteresting branches of the Branch-and-Price tree. Finally, we show some computational results obtained by the application of the Branch-and-Price algorithm on random and realistic instances.*

6.1 Path formulation

In this section, we propose a formulation based on path variables. For this purpose, we first begin by giving a formulation whose variables are associated with sections of the demands. This will be detailed in the following section.

6.1.1 Section formulation

Let $x_e^{q,k}$ be a binary variable associated with demand $k \in K$, section $q \in \mathcal{T}_k$ and edge $e \in E$. $x_e^{q,k}$ are called *section variables* and we have $x_e^{q,k}$ equal to 1 if edge e is used to route section q of demand k and 0 otherwise. $y_e, e \in E$ are the *design variables* and we have y_e equal to 1 if e is installed and 0 otherwise.

Consider the following integer programming formulation.

$$min \sum_{e\in E} c(e)y_e$$

$$\sum_{e\in\delta_{G^{q,k}}(W)} x_e^{q,k} \geq 1 \qquad \begin{array}{l} for\,all\, k \in K, q=(w_j,w_{j+1}) \in \mathcal{T}_k, \\ for\,all\, W \subset V,\ w_j \in W\, and\, w_{j+1} \in \overline{W}, \end{array} \tag{6.1}$$

$$\sum_{q\in\mathcal{T}_k}\sum_{e\in\delta_G(w)} x_e^{q,k} \leq 2 \qquad for\,all\, w \in V, k \in K, \tag{6.2}$$

$$x_e^{q,k} \leq y_e \qquad for\,all\, e \in E, k \in K, \tag{6.3}$$

$$0 \leq x_e^{q,k} \leq 1 \qquad for\,all\, e \in E, k \in K, q \in \mathcal{T}_k, \tag{6.4}$$

$$0 \leq y_e \leq 1 \qquad for\,all\, e \in E, \tag{6.5}$$

$$x_e^{q,k} \in \{0,1\} \qquad for\,all\, e \in E, k \in K, q \in \mathcal{T}_k, \tag{6.6}$$

$$y_e \in \{0,1\} \qquad for\,all\, e \in E. \tag{6.7}$$

Inequalities (6.1) are *section st-cut inequalities*. These inequalities ensure a path for each section q of a demand k and hence two paths for the demand k. Inequalities (6.2) are called the *node-disjunction inequalities*. They guarantee the node-disjunction between the different sections of the demand. This makes sure that the two paths routing the demand are node-disjoint. Inequalities (6.3) are the *linking inequalities*. Inequalities (6.3) indicate that an edge e which is not installed, can not be used to route any section q of a demand k. Inequalities (6.4) and (6.5) are the *trivial inequalities* and inequalities (6.6) and (6.7) are *integrity inequalities*.

It is not hard to see that the integer linear programming formulation (6.1)- (6.7) is equivalent to the MSOND problem. Formulation (6.1)- (6.7) will be called *section formulation*.

The section formulation will serve as a base to obtain a path formulation. To this end, we apply on it a Dantzig-Wolf decomposition. Details are given in the following section.

6.1.2 Dantzig-Wolf decomposition

The Dantzig-Wolf decomposition as applied to an integer program is a specific form of the problem reformulation that aims at providing a tighter linear programming relaxation bound. The reformulation gives rise to an integer master problem with a typically large number of variables [131].
In this section we propose a Dantzig-Wolf decomposition of the integer linear program given by (6.1)-(6.7).
Let us define P_k^q the set of paths routing the section q of demand k calculated in the reduced graph $G^{q,k}$. Define also the variables $z_p^{q,k}$ for each $p \in P_k^q$. The binary variable $z_p^{q,k}$ takes 1 if $p \in P_k^q$ is selected to route section q of demand k and 0 otherwise. These variables are called *path variables* and are linked to the section variables by the following relation

$$x_e^{q,k} = \sum_{p \in P_k^q : p \ni e} z_p^{q,k}. \tag{6.8}$$

When replacing variables $x_e^{q,k}$ by the right hand side of equation (6.8), we obtain a new for of formulation (6.1)- (6.7) given by the following integer linear program.

$$min \sum_{e \in E} c(e) y_e$$

$$\sum_{p \in P_k^q} z_p^{q,k} \geq 1 \qquad for\, all\, k \in K, q \in \mathfrak{T}_k, \tag{6.9}$$

$$\sum_{q \in \mathfrak{T}_k} \sum_{e \in \delta_G(w)} \sum_{p \in P_k^q : p \ni e} z_p^{q,k} \leq 2 \quad for\, all\, w \in V, k \in K, \tag{6.10}$$

$$\sum_{p \in P_k^q : p \ni e} z_p^{q,k} \leq y_e \qquad for\, all\, e \in E, k \in K, q \in \mathfrak{T}_k, \tag{6.11}$$

$$0 \leq y_e \leq 1,\ y_e \in \{0,1\} \qquad for\, all\, e \in E, \tag{6.12}$$

$$0 \leq z_p^{q,k} \leq 1,\ z_p^{q,k} \in \{0,1\} \qquad for\, all\, k \in K, q \in \mathfrak{T}_k, p \in P_k^q. \tag{6.13}$$

Inequalities (6.9) are equivalent to inequalities (6.1) (by Menger's Theorem). Inequalities (6.9)-(6.13) give a path formulation for the MSOND problem. In the following section, we propose an improved version of this formulation.

6.1.3 Path formulation

To simplify notations, we first introduce some definitions that will be used in the sequel. We define coefficient $a_p^{q,k}(w), k \in K, q \in \mathfrak{T}_k, p \in P_k^q, w \in V$ that characterizes the degree of a vertex w in path p routing section q of demand k. It is equal to 1 if w is one of the extremities of section q, 2 if w belongs to p and 0 otherwise. We also define coefficient $b_p^{q,k}(e), k \in K, q \in T_k, p \in P_k^q, e \in E$ that indicates if or not an edge e belongs to the path p routing section q of demand k. It is equal to 1 if e belongs p and 0 otherwise.

The integer linear program given by (6.9)-(6.13) gives a path formulation for the MSOND problem. However, this formulation can be improved to get a tighter linear relaxation bound. In fact, Figure 6.1 illustrates a fractional solution for the linear relaxation of the integer program (6.9)-(6.13). In Figure 6.1 is given a solution for the demand $1-3$ routed on the node-disjoint paths $(1,3)$ and $(1,2,3)$. All the inequalities (6.9)-(6.11) are satisfied. However, we can easily point out that the sections' disjunction constraints are violated. In fact, the Steiner node 4 is a common node the sub-paths routing the sections $(1,3)$ and $(2,3)$ of the demand.

To better see this, consider the incidence vector corresponding to the fractional point of Figure 6.1 given as follows,

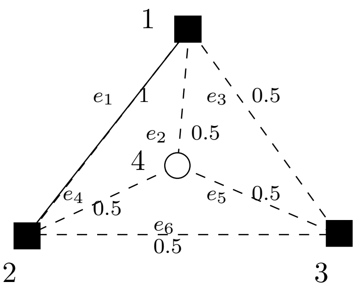


Figure 6.1: Example of fractional solution for the initial path formulation

$z_{p1}^{(1-3),k} = 0.5, \quad p1 = (e_3),$
$z_{p2}^{(1-3),k} = 0.5, \quad p2 = (e_2, e_5),$
$z_{p1}^{(1-2),k} = 1, \quad p1 = (e_1),$
$z_{p1}^{(2-3),k} = 0.5, \quad p1 = (e_6),$
$z_{p2}^{(2-3),k} = 0.5, \quad p2 = (e_4, e_5),$
and $y_{e_1} = 1$, $y_{e_2} = y_{e_3} = y_{e_4} = y_{e_5} = y_{e_6} = 0.5$.

Clearly, vector z satisfy all the inequalities of formulation (6.9)- (6.13). However, remark that path p_2 routing section $(1,3)$ and path p_2 routing section $(2,3)$ both go through the Steiner node 4, which violates the sections disjunction.

To enhance disjunction between the different sections, we add a sum on the different sections in inequalities (6.11) linking the decision variables, as it will be shown in the following integer program.

$$\min \sum_{e\in E} c(e) y_e$$

$$\sum_{p\in P_k^q} z_p^{q,k} \geq 1 \qquad for\, all\; k\in K, q\in \mathcal{T}_k, \tag{6.14}$$

$$\sum_{q\in\mathcal{T}_k}\sum_{p\in P_k^q} a_p^{q,k}(w) z_p^{q,k} \leq 2 \qquad for\, all\; w\in V, k\in K, \tag{6.15}$$

$$\sum_{q\in\mathcal{T}_k}\sum_{p\in P_k} b_p^{q,k}(e) z_p^{q,k} \leq y_e \qquad for\, all\; e\in E, k\in K, \tag{6.16}$$

$$0 \leq y_e \leq 1,\; y_e\in\{0,1\} \qquad for\, all\; e\in E, \tag{6.17}$$

$$0 \leq z_p^{q,k} \leq 1,\; z_p^{q,k}\in\{0,1\} \qquad for\, all\; k\in K, q\in\mathcal{T}_k, p\in P_k^q. \tag{6.18}$$

Remark that inequality (6.11) is satisfied by vector (z, y) given above for the fractional solution of Figure 6.1. However, the equivalent constraint in the ILP given by (6.14)- (6.18), namely inequality (6.16) is no more satisfied by the same vector

(z, y). To prove this, consider edge e_5. By inequality (6.16) applied for e_5, we have the following $\sum_{q\in\mathfrak{T}_k}\sum_{p\in P_k} b_p^{q,k}(e_5)z_p^{q,k} = z_{p2}^{(1-3),k} + z_{p2}^{(2-3),k} = 0.5 + 0.5 = 1$, which is greater than $y_{e_5} = 0.5$. This implies that, with the new form of coupling inequality (6.16), we cut the fractional point of Figure 6.1, which means that the integer program given by (6.14)-(6.18) constitutes a tight path formulation for the MSOND problem.

Notice that this formulation contains an exponential number of variables. To solve the linear relaxation of this formulation, it is necessary to use a column generation procedure. Next sections will be devoted to detail the pricing problem of formulation (6.14)- (6.18), as well as other features to be used among a Branch-and-Price algorithm.

But before discussing the Branch-and-Price algorithm's features, we propose a comparison of the cut and path linear relaxations.

6.2 Cut versus path formulation

To compare the cut and path formulations, we first set relations between their corresponding variables.

6.2.1 Relation between variables

Consider the demand variables x_e^k defined in Chapter 4. It is not hard to see that these variables can be linked to the section variables $x_e^{q,k}$ introduced in Section 6.1 as follows

$$x_e^k = \sum_{q\in\mathfrak{T}_k} x_e^{q,k}. \tag{6.19}$$

Moreover, from (6.8) we know that

$$x_e^{q,k} = \sum_{p\in P_k^q; p\ni e} z_p^{q,k} = \sum_{p\in P_k^q} b_p^{q,k}(e)z_p^{q,k}. \tag{6.20}$$

Similarly, we can write

$$\sum_{e\in\delta(w)} x_e^{q,k} = \sum_{p\in P_k^q} a_p^{q,k}(w)z_p^{q,k}. \tag{6.21}$$

To set up relation (6.21), we distinguish two cases.

1) Consider a section $q = (w_j, w_{j+1})$ of demand k and consider a Steiner node w (i.e. $w \neq w_j$ and $w \neq w_{j+1}$).

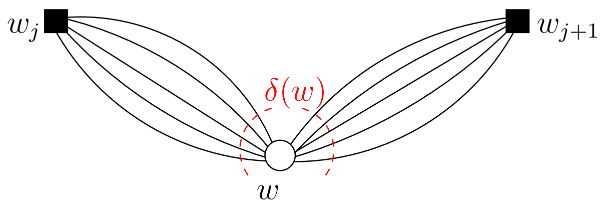


Figure 6.2: Degree of a Steiner node in a path

From 6.20, we have $\sum_{e\in\delta(w)} x_e^{q,k} = \sum_{e\in\delta(w)} \sum_{p\in P_k^q;p\ni e} z_p^{q,k}$. Since w is crossed by two edges of p for all $p \in P_k^q$ then we have $\sum_{e\in\delta(w)} x_e^{q,k} = \sum_{p\in P_k^q;p\ni w} 2z_p^{q,k}$ (see figure 6.2). Obviously, this is equivalent to (6.21) when w is a Steiner node.

2) Assume now that $w = w_j$ (or by symmetry that $w = w_{j+1}$). Note here that w is crossed only once by every path $p \in P_k^q$. This implies that $\sum_{e\in\delta(w)} x_e^{q,k} = \sum_{p\in P_k^q;p\ni w} z_p^{q,k}$ (see figure 6.3). Clearly, this is equivalent to (6.21) when w is an extremity of the considered section.

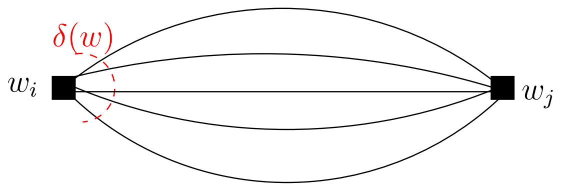


Figure 6.3: Degree of a section's extremities in a path

Based on these relations, we will discuss in the next section the relationship between cut and path linear relaxations.

6.2.2 Relation between linear relaxations

Proposition 6.1 *A solution for the linear relaxation of cut formulation is not necessarily a solution for the linear relaxation of path formulation*

Proof To prove the proposition, we give a counterexample.

Consider the fractional point illustrated in Figure 6.4. This point represents a solution for the demand $1-2$ routed on the node-disjoint paths $(1,2)$ and $(1,3,2)$. Denote k this demand, and let x^k be the corresponding incidence demand vector. We have $x^k = (0.5, 0.5, 0.5, 0.5, 0.5, 0.5)$. Clearly, x^k satisfies all the inequalities of the linear relaxation of the cut formulation (4.6).

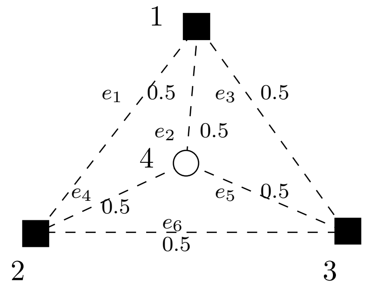


Figure 6.4: Example of fractional solution for the cut formulation

Now, let $x^{q,k}$ be the section vector corresponding to the point of Figure 6.4. We have

$$x^{(1-2),k} = (0.5, 0.5, 0, 0.5, 0, 0),$$
$$x^{(1-3),k} = (0, 0.5, 0.5, 0, 0.5, 0),$$
$$x^{(3-2),k} = (0, 0, 0, 0.5, 0.5, 0.5).$$

In addition, let $z_p^{q,k}$ represents the corresponding path vector. Remark that the only possible values of path variables routing demand k are the following.

$$z_{p1}^{(1-2),k} = 0.5, \quad p1 = (e_1),$$
$$z_{p2}^{(1-2),k} = 0.5, \quad p2 = (e_2, e_4),$$
$$z_{p1}^{(1-3),k} = 0.5, \quad p1 = (e_3),$$
$$z_{p2}^{(1-3),k} = 0.5, \quad p2 = (e_2, e_5),$$
$$z_{p1}^{(3-2),k} = 0.5, \quad p1 = (e_6),$$
$$z_{p2}^{(3-2),k} = 0.5, \quad p2 = (e_5, e_4).$$

Observe that vector $z_p^{q,k}$ satisfy all the constraints of the path formulation excepted inequalities (6.15) corresponding to the disjunction requirement. To see this, let us write inequality (6.15) for the Steiner node 4.

$\sum_{q\in\mathcal{T}_k}\sum_{p\in P_k^q} a_p^{q,k}(4) z_p^{q,k} = 2*(0+0.5+0+0.5+0+0.5) = 2*1.5 = 3 > 2$, which violates the disjunction constraint.

This implies that the solution of Figure 6.4 can not induce a path vector that satisfy (6.14)- (6.18). $\square$

Proposition 6.2 *Every solution for the linear relaxation of path formulation is a solution for the linear relaxation of cut formulation*

Proof Consider a feasible solution for the path formulation (6.14)- (6.18). Denote the incidence vector induced by this solution by (z, y), where z are the paths variables and y are the design variables. Let $(x, y) \in \{0,1\}^{(|K|+1)|E|}$ be the incidence vector defined as follows, $x_e^k = \sum_{q \in \mathcal{T}_k} \sum_{p \in P_k} b_p^{q,k}(e) z_p^{q,k} = \sum_{q \in \mathcal{T}_k} x_e^{q,k}, k \in K, e \in E$. These variables are called demand variables. $y_e, e \in E$ are called design variables and are equal to the design variables corresponding to the path formulation.

In what follows, we will show that (x, y) defines a feasible solution for cut formulation.

First, we prove that x satisfies inequality (4.1). Consider a section $q = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k$. Let $W \subset V$; $w_j^k \in W \, and \, w_{j+1}^k \in \overline{W}$. Consider the cut defined over W in the reduced graph $G^{q,k}$ and written in terms of section variables. By relation (6.19) we have

$$\sum_{e \in \delta_{G^{q,k}}(W)} x_e^{q,k} = \sum_{e \in \delta_{G^{q,k}}(W)} \sum_{p \in P_k^q; p \ni e} z_p^{q,k}. \tag{6.22}$$

Remark that all the paths $p \in P_k^q$ uses mandatory edges that belong to the cut $\delta_{G^{q,k}}(W)$. This implies that

$$\sum_{e \in \delta_{G^{q,k}}(W)} \sum_{p \in P_k^q; p \ni e} z_p^{q,k} = \sum_{p \in P_k^q} z_p^{q,k}. \tag{6.23}$$

Recall that as (z, y) is feasible for the path formulation, we have $\sum_{p \in P_k^q} z_p^{q,k} \geq 1$ by (6.14).

Hence by (6.22) and (6.23), we can write $\sum\limits_{e \in \delta_{G^{q,k}}(W)} x_e^{q,k} \geq 1$. Moreover, as by (6.19) $x_e^k = \sum_{q \in \mathcal{T}_k} x_e^{q,k}$, this implies that $x_e^k \geq x_e^{q,k}$. Then $\sum\limits_{e \in \delta_{G^{q,k}}(W)} x_e^k \geq \sum\limits_{e \in \delta_{G^{q,k}}(W)} x_e^{q,k} \geq 1$, and the result follows.

Now, we show that inequality (4.2) is satisfied by vector x. By inequalities (6.15) and relation (6.21), we have $\sum_{q \in \mathcal{T}_k} \sum_{p \in P_k^q} a_p^{q,k}(w) z_p^{q,k} = \sum_{q \in \mathcal{T}_k} \sum_{e \in \delta(w)} x_e^{q,k} = \sum_{e \in \delta(w)} \sum_{q \in \mathcal{T}_k} x_e^{q,k} \leq 2$. By relation (6.19), we can then deduce that $\sum_{e \in \delta(w)} x_e^k \leq 2$.

Consider now inequality (4.3) and let us prove that they are satisfied by vector x. By (6.19), (6.20) and (6.16) we have $x_e^k = \sum_{q \in \mathcal{T}_k} \sum_{p \in P_k} b_p^{q,k}(e) z_p^{q,k} \leq y_e$. Knowing that $\sum_{p \in P_k} b_p^{q,k}(e) z_p^{q,k} = x_e^{q,k}$, we can write that $x_e^k \leq y_e$.

Finally, by (6.17), we can write $x_e^k \leq 1$ and by (6.18) $z_p^{q,k} \geq 0$ yielding $x_e^k \geq 0$.

In conclusion, (x, y) is a feasible solution for the cut formulation, and this ends the proof. □

Corollary 6.3 *Any solution of the linear relaxation of path formulation implies a feasible one for the linear relaxation of cut formulation (the converse is not true).*
This means that $Z_{path} \geq Z_{cut}$.

Recall that the path formulation (6.14)- (6.18) has a huge number of variables. To deal with this, we devise an algorithm called a *Branch-And-Price algorithm* that we present in details in the next section.

6.3 Branch-and-Price algorithm

To solve the integer linear programming formulation (6.14)- (6.18), it is necessary to use a column generation method combined with a branch-and-Bound algorithm. Such an algorithm is called Branch-and-Price algorithm. Algorithm 7 summarizes the steps of this algorithm.

6.3.1 Initial solution

Since the path formulation contains a huge number of variables (columns), it is necessary to begin with a restricted version that contains only a subset of these columns. We generate an initial feasible solution as follows. For each section q of a demand k, we generate the following feasible paths: the first path is given by the edge between the extremities of q. The other paths are given by inserting Steiner nodes between the extremities of section q.

To illustrate the procedure of the initial solution's generation, consider a section $q = (t_1, t_2)$ of a demand $k \in K$. The paths that we generate for section q are the following:

1) Consider the edge between the successive terminals t_1 and t_2, yielding to path $p_1 = (t_1, t_2)$,

Algorithm 7: Branch-And-Price Algorithm

```
   Data: An undirected graph $G = (V, E)$, a set $K$ of demands, 2 node-disjoint
         paths routing each demand $k \in K$
   Result: Optimal solution for $MSOND_{path}$
 1 $LP \leftarrow LP_{ini}$;
 2 Solve the linear program $LP$ and denote by $(z^*, y^*)$ the optimal solution of LP;
 3 Consider the dual variables and solve the pricing problem ;
 4 if for all $k \in K$,$q \in \mathcal{T}_k$ the optimal value of the pricing problem is positive then
 5     go to 9;
 6 else
 7     Add the optimal solution with the most negative reduced cost;
 8     go to 2;
 9 if $(z^*, y^*)$ is integer for MSOND then
10     $(z^*, y^*)$ is an optimal solution. Stop ;
11 else
12     Create two sub-problems using a specific branching rule.
13 forall open sub-problem do
14     go to 2;
15 return the best optimal solution of all the sub-problems ;
```

2) Choose a Steiner node of S_k, say s_1, and insert it between terminals t_1 and t_2 thus resulting in the path $p_2 = (t_1, s_1, t_2)$,

3) Choose two Steiner nodes of S_k, say s_1 and s_2, and generate the paths inserting the two Steiner nodes s_1 and s_2 between t_1 and t_2 while considering the two possible permutations, so as to obtain the paths $p_3 = (t_1, s_1, s_2, t_2)$ and $p_4 = (t_1, s_1, s_2, t_2)$.

Note that using this procedure, the number $|S_i^{q,k}|$ of initial solutions for section q of demand $k \in K$, is given by $|S_i^{q,k}| = 1 + C^1_{|S_k|} + 2 \times C^2_{|S_k|} = 1 + |S_k| + 2\frac{|S_k|!}{2!(|S_k|-2)!} = 1 + |S_k| + |S_k|(|S_k| - 1) = 1 + |S_k|(1 + |S_k| - 1)$, implying that $|S_i^{q,k}| = 1 + |S_k|^2$.

Remark that generating other paths by inserting 3 Steiner nodes or more between the extremities of each section is also possible. However, in our computations, we restricted ourselves to 2 Steiner nodes. This choice was is not arbitrary. In fact, experimentations show that generating initial solutions by inserting at most 2 Steiner nodes gives the best trade-off between resolution's efficiency and time of execution for the majority of instances.

Overall, by the way we generate our initial solution, we are sure that there exists at least one feasible solution. This can be for instance the one given by all the edges between successive terminals of each demand. Indeed, this solution is satisfying all the path formulation inequalities and in particular the node disjunction constraint (6.15).

6.3.2 Pricing algorithm

As it has been mentioned above, we begin by a restricted version of the problem involving only an initial subset of variables. Additional columns will then be generated only when they are needed. The restricted version of the LP form is called *restricted master problem (RMP).*

Denote by $\pi^{q,k}$, λ_w^k and β_e^k the dual variables associated respectively with inequalities (6.14), (6.15) and (6.16), with respect to primal variable $z_p^{q,k}$. The reduced cost of the variable $z_p^{q,k}$ is given by $R_p^{q,k} = -(\pi^{q,k} + \sum_{w \in V} a_p^{q,k}(w)\lambda_w^k + \sum_{e \in E} b_p^{q,k}(e)\beta_e^k)$. Here, the pricing problem consists in finding, for each section q of a demand k, a path of P_k^q such as $R^{q,k} = min_{p\prime \in P_k^q} R_{p\prime}^{q,k}$ and $R^{q,k} < 0$.
Observe that $R_p^{q,k}$ contains a fixed term $-\pi^{q,k}$ and a sum of terms on vertices λ_w^k and a sum of terms related to edges β_e^k. To calculate $R^{q,k} = min_{p\prime \in P_k^q} R_{p\prime}^{q,k}$, we can hence calculate the minimum only on the sums, that is $min_{p\prime \in P_k^q} - (\sum_{w \in V} a_p^{q,k}(w)\lambda_w^k + \sum_{e \in E} b_p^{q,k}(e)\beta_e^k)$. Since this quantity contains node-related terms and edge-related ones, we then can resort to the graph structure to solve our pricing problem.

Consider the reduced graph $G^{q,k}$ corresponding to variables $z_p^{q,k}, p \in P_k^q$ and let us put on each vertex w the weight $-a_p^{q,k}(w)\lambda_w^k$ (such that $a_p^{q,k}(w) = 1$ if w is a section's extremity and 2 if it is a Steiner node). Let us also put on each edge e the weight $-b_p^{q,k}(e)\beta_e^k$ (such that $b_p^{q,k}(e) = 1$). The pricing problem can thus be seen as a shortest path problem in a reduced graph $G^{k,q}$ with weights $-a_p^{q,k}(w)\lambda_w^k$ on vertices and $-\beta_e^k$ on edges. λ_w^k can be after split and put on edges incident to w to hence obtain weights only on the edges of $G^{k,q}$.

As dual variables λ_w^k and β_e^k are negative, edge weights are non negative and the shortest path pricing problem can be solved in polynomial time. In our Branch-and-Price algorithm, we solve the pricing problem using Dijkstra Algorithm [41].

If the value of the shortest path is such that $-(\sum_{w \in V} a_p^{q,k}(w)\lambda_w^k + \sum_{e \in E} b_p^{q,k}(e)\beta_e^k) < \pi^{q,k}$ then $R^{q,k} < 0$ and hence at least one column has to be added to the RMP. If not, then $R^{q,k} \geq 0$ for all $k \in K$ and $q \in \mathcal{T}_q$, which means that the optimal solution of the current linear program is optimal for the linear relaxation of the path formulation.

To illustrate the pricing procedure, we consider the example given by Figure 6.5.

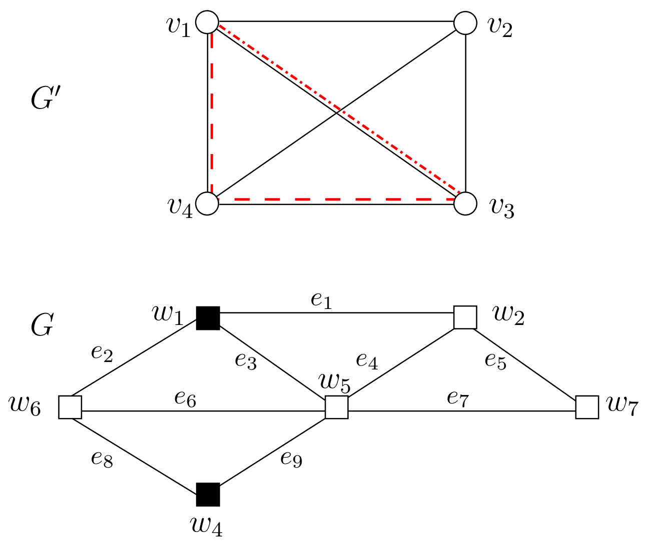


Figure 6.5: Reduced graph for section $(1,4)$

Figure 6.5 shows a demand in G' between (v_1, v_3) routed on paths (v_1, v_3) and (v_1, v_4, v_3). Suppose that we are looking for a candidate column $z_p^{(w_1,w_4),k}$ to insert in the LP for the section (w_1, w_4) of the demand. To see if such column exists or not, one should solve the corresponding pricing problem. Consider the reduced graph $G^{(w_1,w_4),k}$ illustrated in graph G. Recall that this reduced graph is obtained by deleting all the terminals of the demand and their incident edges, excepted the extremities of section (w_1, w_4). In this case, we delete the terminal w_3 as well as its incident edges. To solve the corresponding pricing problem, we insert the dual values as shown in Figure 6.6. Observe that in the graph of Figure 6.6, there are weights on edges given by $-\beta_e^k$ and weights on vertices given by $-a_p^{q,k}(w)\lambda_w^k$. Solving the pricing problem here, is no more than looking for a shortest path between w_1 and w_4. To bring this to the classical shortest path problem, we propose to split the weights on the Steiner nodes as illustrated in figure 6.7. We also delete weights on terminals w_1 and w_4. These values ($-\lambda_{w_1}^k$ and $-\lambda_{w_4}^k$) will be added after to the resulting value of the shortest path.

The solution of the LP relaxation solved by column generation could not be integral, and hence not a solution for the MSOND problem. In this case, to obtain the optimal solution for the MSOND problem, we must proceed to a branching phase. This will be detailed in next section.

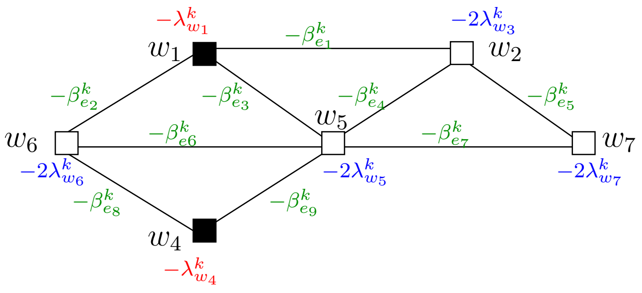


Figure 6.6: Dual values on the reduced graph

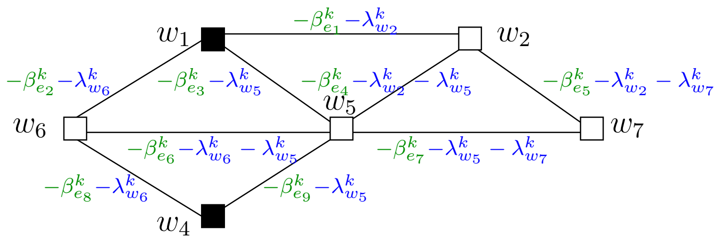

$-\beta^k_{e_1}-\lambda^k_{w_2}$
w_1
w_2
$-\beta^k_{e_2}-\lambda^k_{w_6}$
$-\beta^k_{e_3}-\lambda^k_{w_5}$
$-\beta^k_{e_4}-\lambda^k_{w_2}-\lambda^k_{w_5}$
$-\beta^k_{e_5}-\lambda^k_{w_2}-\lambda^k_{w_7}$
w_5
w_6
w_7
$-\beta^k_{e_6}-\lambda^k_{w_6}-\lambda^k_{w_5}$
$-\beta^k_{e_7}-\lambda^k_{w_5}-\lambda^k_{w_7}$
$-\beta^k_{e_8}-\lambda^k_{w_6}$
$-\beta^k_{e_9}-\lambda^k_{w_5}$
w_4

Figure 6.7: Reduced graph with weights only on edges

6.3.3 Branching scheme

6.3.3.1 Examples of branching rules

At each node of the Branch-and-Price tree, if no more interesting column is found and if the solution is not integer, we manage the branching phase. The difficulty in incorporating column generation with branch-and-bound is that conventional integer programming branching may not be effective because fixing variables can destroy the structure of the pricing problem. In fact, when the satellite problem is a shortest path problem, at a given level of the tree, the pricing problem could change to the k^{th} shortest path problem.

The challenge is hence to identify a branching rule that eliminates the current fractional solution without compromising the tractability of the pricing problem. In [16], Barnhart et al develop branching rules for a number of different master problem structures.

In general, the branching rules for path-based formulation are defined on original edge (arc) variables. In [17], Barnhart et al propose a genereliazed Ryan and Foster [117] branching rule for origin-destination integer multicommodity flow problems. Their branching rule is based on forbidding the use of specific arcs. At a divergence point, the idea is to divide in a specified way the leaving arcs from this point and forbid in a branch a subset of this arcs and on the other branch the other subset. In [47], Feillet gives a branching on the arc variables as follows. Forbidding the arc (i_0, j_0) consists in removing this arc when calculating the shortest path problem. Imposing the arc (i_0, j_0) can be seen also as forbidding some arcs. Imposing the use of (i_0, j_0) amounts to forbid all the arcs leaving i_0 and entering to j_0 but the arc (i_0, j_0).

6.3.3.2 Our branching strategy

For the path formulation, the branching rule can be described as follows. Given a fractional LP solution $(\bar{z}, \bar{y})$, we select the most fractional design variable, say $\bar{y}_e$, and we branch by setting the variable to 0 or 1. Notice here that when branching on the design variables, the structure of the pricing problem does not change. At a certain level of the tree, we may have integrity for all the design variables and no optimal solution is yet found. Here, a branching on the path variables is necessary to get the optimal solution. To maintain the same structure of the pricing problem, we choose to branch by imposing to go through edge e on one hand and forbidding the use of e on the other. A possible procedure is to branch by adding constraints $\sum_{p \in P^{q,k}} b_p^{q,k}(e) z_p^{q,k} = 1$ for the first branch and $\sum_{p \in P^{q,k}} b_p^{q,k}(e) z_p^{q,k} = 0$ for the second. When adding these constraints, the reduced cost changes inducing the appearance of negative costs and eventually negative cycles in the shortest path problem. And hence the pricing problem can no longer be solved with a polynomial algorithm. In our case, this difficulty can be overcome using the following procedure.

As all the design variables $\bar{y}_e$ are integer, we choose the most fractional path variable, say $\bar{z}_p^{q,k}$. Assume that this variable is such that $q = (t_0, t_1)$ and $p = (t_0 s_1, s_1 s_2, ..., s_j t_1)$. The idea is to choose in a clever way the edge for branching. For this purpose, we opt for the first edge of the selected path p, say e_0 (observe here that $e_0 = (t_0, s_1)$). On the first branch, the use of e_0 is forbidden. This can be done by deleting the edge e when calculating the shortest path pricing problem. On the second branch, e_0 is imposed in the path. Since the first extremity of e coincides with t_0, the pricing problem reduces to a shortest path calculated in the reduced graph $G^{q,k}$ between s_1 (second extremity of e) and t_1 (destination of section q). At a given depth of the Branch-and-Price tree, one could have to branch again on (q, k). In this case, $e_1 = (s_1, s_2)$ constitues the current edge of branching, for which the process described above is similarly applied. By this way of branching, at each branching on (q, k), a sequence of edges that is nothing but a sub-chain originating in t_0, is imposed. The pricing problem remains always a shortest path problem calculated for each section of a demand in a reduced graph and such that the origin of the shortest path changes depending on wether the section in question has been considered before in the branching tree or not.

6.3.4 Primal heuristic

To accelerate the Branch-and-Price algorithm and enable a fast pruning of uninteresting branches of the tree, we propose a primal heuristic. Given a fractional solution, we deduce a feasible solution for the MSOND problem as shown in algorithm 8. The idea of the primal heuristic we suggest is the following. We start from a feasible fractional solution. After, for each demand k and for each section q of the considered demand, we retain a fractional feasible path routing the section q. If this path is using Steiner nodes of the demand k, we mark the used Steiners, and we continue the procedure. The Steiner nodes that are marked once, could not be used in the other sub-paths routing the remaining sections of the demand. When no possible solution is found for the section q, we look back into the pool of variables already generated for section q and we fix the most suitable path.

Based on these features, we devise a Branch-and-Price algorithm that we tested on random as well as realistic instances. The results we obtain are discussed in the following section.

Algorithm 8: Primal Heuristic

```
   Data: Fractional Solution $(z, y)$
   Result: Integer Feasible Solution $(z, y)$
 1 forall $k \in K$ do
 2     $S_{k,+} = \emptyset$ ;
       /* the set of marqued Steiner nodes for the demand $k$ */
 3     Let $q = (t_i, t_j) = (t_0, t_1) \in \mathcal{T}_k$ the first section of the demand $k$ ;
 4     repeat
 5         let the path variable $z^{q,k}_{p_i} = (t_i, Sp_i, t_j)$ be the nearest to 1 ;
           /* where $Sp_i \subseteq S_k$ is the set of Steiner nodes used in the
              path $z^{q,k}_{p_i}$ */
           /* if there are so many paths with the same fractional value,
              choose the one that uses the most edges already used by
              the other demands or choose the one with the minimum
              distance length */
 6         $z^{q,k}_{p_i} \leftarrow 1$ ;
 7         $S_{k,+} = S_{k,+} \cup Sp_i$ ;
 8         $q = (t_i, t_j) \leftarrow (t_{i+1}, t_{j+1})$ ;
 9     until $S_{k,+} = S_k$ or $t_{j+1} = t_0$ ;
       /* end when all the Steiner nodes are marqued or when all the
          sections of the demand $k$ are routed */
10     if $S_{k,+} = S_k$ and $t_{j+1} \neq t_0$ then
11         repeat
12             look over the pool of path variable routing section $j + 1$ of demand $k$ ;
13             choose the best path variable $z^{q,k}_{p_{j+1}}$ routing $(t_{j+1}, t_{j+2})$ ;
14             $z^{q,k}_{p_{j+1}} \leftarrow 1$ ;
               /* We choose the path variable that passes through the
                  edges already used by the other demands unless the
                  direct edge */
15         until $t_{j+1} = t_0$ ;
16 update design variables $y$ ;
17 return the integer feasible solution $z, y)$ ;
```

6.4 Computational results

The Branch-and-Price algorithm described in the previous section has been implemented in C++ using ABACUS 3.2 [1] to manage the Branch-and-Price tree and

CPLEX 12.0 [2] as LP solver. It was tested on a Bi-Xeon quad-core E5507 2.27GHz with 8Go of RAM, running under Linux.
The algorithm was tested on two types of instances: random instances and realistic instances that have been described in Section 5.2.2. In the sequel, the entries of the different tables are :

V'	:	number of node in the logical layer (graph G');
V	:	number of node in the optical layer (graph G);
K	:	number of demands;
Path-ini	:	number of columns generated in the initial solution;
Path-gen	:	number of columns generated during the pricing;
Nodes	:	number of nodes in the tree;
Relaxations	:	number of solved linear relaxations;
Gap	:	the relative error between the best upper bound (the optimal solution if the problem has been solved to optimality) and the lower bound obtained at the root node of the Branch-and-Price tree;
Gap-Y	:	the relative error between the best upper bound obtained only by branching on the design variables y and the lower bound obtained at the root node of the Branch-and-Price tree;
Opt	:	number of instances solved to optimality over 5 tested ones;
TR	:	the time spent to find the lower bound at the root node;
TB	:	the time necessary to find the best feasible solution;
TT	:	total CPU time.

The CPU times are given in h:mm:ss for realistic instances and in seconds for the random instances since for these later we calculate the average over 5 instances. Some rows of the result tables are preceded by a star *. This refers to instances that could not be solved to optimality within the time limit of 3 hours.

Our first series of experiments concern the random instances. Tables 6.1 and 6.2 report, for each size of the instances, the average results obtained over the TSP-based instances. The reported results concern 78 instances with a number of nodes in the higher layer (graph G') varying from 6 to 55, a number of nodes in the optical layer (graph G) ranging from 4 to 50 and a number of demands going from 2 to 16. For each group of instance, the row of the table represents the average value over 5 tested instances based on $a280$, $bier127$, $eil101$, $lin105$ and $tsp225$ and having the same size, but with different composition of the demand. In table 6.1, 5 over 5 of all the tested size of instances were be solved to optimality with at maximum $V' = 14$, $V = 12$ and

$K = 16$. Moreover, the instances were solved to optimality within at most 1 hour and many instances took only few seconds to reach the optimal value. This means that for small instances, our algorithm performs very well. In table 6.1, for big-sized instances, the problems become much more difficult to solve. In particular, for the biggest graph, only 2 demands could be considered. Notice that the difficulty in solving the instances is mainly related to their sizes. We remark that, the greatest the instance is, the most difficult the resolution become. The difficulty of these instances is also related to their random aspect. In fact, only 2 of instances $V' = 16$, $V = 14$ and $K = 8$ were solved to optimality. Remark that the instances here are of the same size. However, someones could be solved optimally, the others not. This is due to the composition of the demands and the random aspect of the instances. In fact, when generating randomly the demands, one demand could be routed using very distant intermediair terminals between the source and the destination and this make the demands difficult to route.

Tables 6.1 and 6.2 show also that 49 groups of instances was solved with a root lower bound situated at most at 10% from the optimum value and the necessary time to obtain these bounds didn't exceed some seconds. However, for the remaining groups of instances the gap become very important and could reach 46% for the instances with $V' = 20$, $V = 18$ and $K \geq 15$. This is due mainly to the difficulty of the random instances.

The same tables show that for small instances, branching only on the design variables was sufficient to get an optimal solution. However, when instances become greater, a branching involving the path variables was necessary to have an integer optimal solution.

From tables 6.1 and 6.2, we can see that the number of generated paths during the pricing procedure ($Path - gen$) is not very huge in particular for small instances for which we could obtain the optimal solution just after some iterations. In fact, for 23 groups of instances, we reached the optimum solving less than 50 linear relaxations and generating in average less than 30 columns. This is thanks to the good quality of the initial solution given by $Path - ini$. Recall that for each section of each demand, we generated a first solution by inserting up to two Steiner nodes between the terminals extremities of the section, and this gives a very good initial base. For the remaining instances, the number of generated paths become more important as the size of the instance increases. In particular, we notice a very huge number of generated paths for the group of instances that could not be solved to optimality within 3 hours.

Tables 6.1 and 6.2 also show that the number of nodes of the Branch-and-Price tree

V'	V	K	Path-ini	Path-gen	Nodes	Relaxations	Gap	Gap-Y	Opt	TR	TB	TT
6	4	2	60	0,2	1	1,2	0	0	5/5	0	0	0,6
6	4	4	120	0,6	1	1,4	0	0	5/5	0	0	0,6
6	4	5	150	1	1	1,4	0	0	5/5	0	0	0,6
6	4	6	180	0,6	1	1,6	0	0	5/5	0	0	0,6
8	6	2	118	0,4	1	1,4	0	0	5/5	0	0	0,6
8	6	4	236	2,6	8,6	10,2	7,33	0	5/5	0	0	0,5
8	6	5	314	5,4	9,8	14,2	7,33	0	5/5	0	0	0,3
8	6	6	382	6,6	12,6	15	5,01	0	5/5	0	0	0,2
8	6	8	518	6,8	7,4	9,6	0,39	0	5/5	0	0	0,3
8	6	10	646	5,2	9	10,2	2,14	0	5/5	0	0	0,4
8	6	12	782	3,8	7	8,4	2,14	0	5/5	0	0	0,4
8	6	15	986	4,6	13	13,2	4,35	0	5/5	0	0	0,0
10	8	2	278	1,8	1	1,8	0	0	5/5	0	0	0,6
10	8	4	558	78,6	10,2	42,6	3,23	0	5/5	0	0	3,3
10	8	5	660	60,2	17	47,6	1,14	0	5/5	0	0	3,0
10	8	6	790	78,2	24,6	79,6	1,26	0	5/5	0	2,8	7,5
10	8	8	1010	20,2	15	24,8	2,11	0	5/5	0	0	0,2
10	8	10	1182	15,8	25,4	30,4	5,32	0	5/5	0	0	0,0
10	8	12	1478	20,2	28,6	34,6	5,32	19,16	5/5	0	0	0,0
10	8	15	1858	50,2	75	108,2	8,37	3,11	5/5	0	10,4	15,4
10	8	16	1960	47,2	63,8	94	8,57	1,93	5/5	0	5,2	19,2
12	10	2	404	16,8	17,8	29,4	3,6	0	5/5	0	0	0,3
12	10	4	914	272,2	45,4	202,2	3,91	0	5/5	0	10,6	21,9
12	10	5	1096	313	112,6	351,4	7,24	0	5/5	0	17,2	40,2
12	10	6	1356	453,2	101,4	397,6	7,4	19,6	5/5	0	18	50,9
12	10	8	1800	366,4	202,6	509,6	6,23	19,61	5/5	0	57	74,4
12	10	10	2232	269,2	127,4	331	7,97	3,9	5/5	0	33	54,4
12	10	12	2590	288,2	302,6	572,8	7,37	19,63	5/5	0	75,2	120,3
12	10	15	3204	536,8	935	1841	8,86	13,98	5/5	0	209	595,8
12	10	16	3426	582	1201,8	2204,6	8,86	19,62	5/5	0	180,4	837,3
14	12	2	814	46	1	17,4	0	0	5/5	0	0	0,0
14	12	4	1584	327,6	1,8	69,6	0	0	5/5	0	0	4,0
14	12	5	1974	7585,8	480,2	6184,2	5,08	21,58	5/5	0	567,4	2042,8
14	12	6	2384	4676,6	235	2969,6	3,59	19,66	5/5	2,8	437	903,1
14	12	8	3084	3839,4	303,8	2444,2	7,1	14,15	5/5	0	299,8	604,6
14	12	10	3878	5405	545,8	3804,8	6,4	55,6	5/5	2,6	697,6	1371,9
14	12	12	4698	6009,4	887,4	5008,8	6,69	50,51	5/5	2,8	704	2643,9
14	12	15	5770	3362,6	871	3190,6	8,08	19,68	5/5	2,6	1525	1954,9
14	12	16	6174	3431	886,6	3218,4	8,08	4,64	5/5	3	1736,4	2206,4

Table 6.1: Branch-and-Price results for random instances (1)

is not huge. This is thanks to the primal heuristic that enabled a fast convergence of our algorithm.

V'	V	K	Path-ini	Path-gen	Nodes	Relaxations	Gap	Gap-Y	Opt	TR	TB	TT
16	14	2	1190	74,8	1	21,6	0	0	5/5	0	0	0,0
16	14	4	2306	847,2	7	226	0,39	0	5/5	6,6	11	28,8
16	14	5	2912	9263	450,2	6207	3,87	13,89	5/5	0	567,2	2334,2
16	14	6	3522	11858,6	533	7090,2	6,3	19,12	5/5	0	1723	3403,1
16	14	8	4708	14177,4	764,6	8749,6	7,11	42,25	2/5	8	2334,4	7156,1
16	14	10	5802	11848,2	1174,2	9540,2	11,08	22,57	2/5	0	4094,8	7959,9
*16	14	12	6982	12233	1306	10555,2	24,17	0	0/5	11,6	8387,2	10800
*16	14	15	8628	7406,2	1211,2	7573,4	29,87	0	0/5	18,8	3548,6	10800
*16	14	16	9138	6993	1050,8	6762,4	26,48	0	0/5	20,2	5739,8	10800
18	16	2	1658	125	1,4	31,6	0	0	5/5	0	0	0,0
18	16	4	3206	2642,4	14,6	1032,6	1,29	0	5/5	22,6	80,8	221,6
18	16	5	4060	14849	327,6	7680,6	5,97	38,63	3/5	14,8	3152,8	6533,9
18	16	6	4738	14295	255	6897	6,01	38,63	2/5	27	4417,4	6564,3
*18	16	8	6224	17655,4	382,4	8202,4	13,62	15,84	1/5	39,6	7383,2	8829,9
*18	16	10	7948	17216,4	529,6	8646,8	21,43	15,98	0/5	31,2	6869,4	10800
*18	16	12	9672	13843,4	558,6	7364,4	25,29	0	0/5	31,2	5986,8	10800
*18	16	15	12008	11597,8	369,2	5382,4	32,63	0	0/5	52,6	8020,6	10800
*18	16	16	12878	11041,2	316,2	4835,8	32,06	0	0/5	58,2	8166	10800
20	18	2	2158	166,6	1	39	0	0	5/5	0	0	0,0
20	18	4	4158	2789,8	9,4	740,2	0,01	0	5/5	39	192,6	246,3
20	18	5	5340	18493,4	156,4	7285,6	2,89	19,54	2/5	57,6	3373,2	7549,8
20	18	6	6522	21695,2	126,4	7150,6	7,3	0	1/5	100,6	4801,4	9429,5
*20	18	8	8710	23298,2	101,8	6280,8	13,18	19,08	0/5	212,2	5804	10800
*20	18	10	11060	19414,8	257,2	6408,8	30,21	0	0/5	67,8	6084,4	10800
*20	18	12	13432	16025,2	175,4	4560	34,01	0	0/5	104,4	6971	10800
*20	18	15	16874	14645,2	123,6	3659,8	46,8	0	0/5	162,6	7261,2	10800
*20	18	16	18004	14225	106,6	3404,2	46,7	0	0/5	189,2	4616,4	10800
25	20	2	3627	241,6	1	70,2	0	0	5/5	0	0	5,5
25	20	4	7357	7672,6	52,2	2977,4	0,53	0	4/5	147,8	325,8	3257,7
30	25	2	6592	6052,4	44,6	3043,4	0,67	0	5/5	14,8	460,4	2043,8
*30	25	4	12762	10374	7,6	1418,6	2,92	0	0/5	3309,4	7619,6	10800
35	30	2	10000	7576,4	59,8	3360	0,12	0	3/5	105,2	2628,4	4554,9
35	30	4	19343	8535,6	1,8	833,4	15,04	0	0/5	5087,8	8372,6	10800
40	35	2	13760	7037,8	31,2	2310,6	0,3	0	3/5	646,8	2365	6441,9
*40	35	4	26578	10098,4	1,2	776,2	27,42	0	0/5	5678,6	6404,8	10800
45	40	2	18120	5239	6,8	1016,2	1,05	0	1/5	5140,2	6583,8	10582,4
*45	40	4	34963	19170,6	1	1286,2	28,47	0	0/5	3561,8	7393,2	10800
50	45	2	23080	3403,8	1,2	440,4	2,84	0	2/5	2188,8	3759,2	10640,4
*55	50	2	28640	4457,6	1,6	615,6	3,26	0	0/5	5893,8	6003,8	10800

Table 6.2: Branch-and-Price results for random instances (2)

Our second series of experiments concern realistic SNDlib-based instances. The results for these instances are reported in tables 6.3 and 6.4 report results of the Branch-and-price algorithm for the realistic instances. Results are given for 88 instances with a number of nodes ranging from 10 to 65 and a number of demands going from 2 to 30. The tables show that only 18 over the 88 instances where not solved to optimality within the time limit. All the remaning have reached the optimal solution before 3 hours and 62 among them were solved to optimality in less than 10 minutes. In particular, all the instances of $dfn - bwin$ with 10 nodes and demands from 2 to 30 have been solved to optimality within at most 2 minutes. Remark also that for a graph with 12 nodes (instances *polska*), we could solve to the optimum up to 30 demands. However, with random instances having the same number of nodes, we are limited to 16 demands. This proves that realistic instances are easier to solve than the random ones. This is logic since demands in the realistic instances are generated by calculating shortest paths and are in contrast randomly generated for the random instances.

As long as the number of nodes of the graph increases, the number of variables increases as well, yielding to much difficult instances to solve. This is the example of the instances of more than 30 nodes in the graph. Notice that half of the instances $pioro40$ and $germany50$ have not been solved to optimality within the time limit. The execution time depends also on the number of the demands in the instances. Instances with restricted number of demands are generally solved faster than the ones with a greater number of commodities. However, this remains not sufficient to explain the behaviour of some instances. See for example the instances of *newyork*. For these instances, passing from 6 to 8 demands leads to a sudden augmentation of the difficulty of the instances which could not be solved to optimality before 3 hours. This difficulty persists also for the instances with 10 and 12 demands which are not solved optimally within the time limit. An unexpected behaviour begins with instances of more than 14 demands that reach the optimum within less than 2 hours. Notice here that instances with 8, 10 and 12 demands are more difficult to solve than the greater ones. This leads us to think about another aspect causing the difficulty of the instances which is the composition of the demands.

Recall that each demand is characterized by a number of terminals that must be visited in a predefined order. This induces that when two or more demands share a number of terminals with conflicting order constraints, the instance become more difficult to solve. This is, in particular, the case of the instance *newyork* with 8 demands. In fact, this instance contains 2 conflicting asymmetric demands: $(13, 14)$ routed by paths $(13, 6, 14)$ and $(13, 0, 14)$ and $(14, 13)$ routed by paths $(14, 13)$ and $(14, 0, 6, 13)$.

Remark in this case that, while the first demand requires that terminals 6 and 0 are visited independently in different paths, the second demand is forcing the passage through 0 and 6 in the same path and successively. This leads to a conflict between the two demands and hence to a difficult instance to solve.

Surprisingly, such a situation could be unblocked when adding other demands. In fact, when adding one or more demands, new nodes and edges of the graph become necessary to visit, offering hence the possibility to be used by the previous demands since the cost of installation does not depend on the number of demands routed on an edge. This explains why instances *newyork* with 14 demands and more, have been solved optimally within 3 hours. In fact, when we add demand $(0, 8)$ routed on paths $(0, 6, 8)$ and $(0, 7, 8)$, the contradictory situation between the previous demands disappears since new edges and nodes come into play. Hence, depending on the composition of the demands, adding new demands to an instance could make the problem easier to solve. Let us also take the example of instances *newyork* with 16 demands solved within about 2 hours and *newyork* with 18 demands which became easier and is solved to optimality within 17 minutes.

The tables show also two other kind of times: the time passed to get the lower bound at the root of the tree and the necessary time to get the best feasible solution. Notice that for 70 instances over the 88 tested, the bound of the root is obtained within less than 10 minutes. Moreover, for these instances, the root's bound has a gap which does not exceed 10%. The same remark is valid for the necessary time to have the best feasible solution. In fact, for all the instances of less than 22 nodes in the graph, we got the best feasible solution in less than 3 hours and with a good gap not exceeding 12% except for the instances of *geant* with 25 and 30 demands. For the instances containing more than 22 nodes (the *cost*266, *pioro*40, *germany*50, *zib*54 and *ta*2 based instances), 11 instances over 18 took only few minutes to get the best feasible solution. The remaining instances, *pioro*40 (5, 6 and 8 demands), *germany*50 (4, 5 and 6 demands) and *ta*2 (6 demands) exhaust all the 3 hours (or a bit more in the experiments) without even terminating the resolution of the relaxations at the root. For these instances, the calculation of the gaps was not possible and this is why we put " $-$ " in the corresponding columns. Notice that for the mentioned instances, an important number of linear relaxations is solved, only at the root of the tree, reaching 5444 for the instance *germany*50 with 5 demands. This means that, at this stage, an introduction of a cutting plane procedure could improve the linear relaxation and yields to a faster resolution of the instances. Tables 6.3 and 6.4 also report the number of generated paths in the column generation procedure. Remark that this number is not very huge. In fact, it does not exceed 3 times the number of constraints for each instances. This is thanks to the initialisation procedure that seems to be good.

6.5 Concluding remarks

In this chapter, we have introduced a second formulation for the MSOND problem. In contrast with the cut formulation developed in Chapter 4, the new formulation, called path formulation, uses a polynomial number of constraints but an exponential number of variables. We have discussed the pricing problem for this formulation and we prove that it reduces to a shortest path problem. We have devised a Branch-and-Price algorithm to solve this formulation and propose a specific branching scheme. Moreover, we have presented a primal heuristic that aims at enhancing the convergence of the Branch-and-Price algorithm. Finally, we have given some computational results. These results show that our Branch-and-Price algorithm performs well for the resolution of random as well as realistic instances. However, for some large instances, the algorithm does not success to reach the optimal solution within the time limit. A very interesting perspective to overcome this shortage is to get profit from the valid inequalities identified in Chapter 4 and combine the column generation with a row generation, yielding to an efficient Branch-and-Price-and-Cut algorithm.

Instance	V	K	Path-ini	Path-gen	Nodes	Relaxations	Gap	Gap-Y	TR	TB	TT
dfn-bwin	10	2	298	4	1	3	0	0	0:00:00	0:00:00	0:00:00
dfn-bwin	10	4	598	2	1	2	0	0	0:00:00	0:00:00	0:00:00
dfn-bwin	10	5	748	12	1	9	0	0	0:00:00	0:00:00	0:00:01
dfn-bwin	10	6	898	48	1	11	0	0	0:00:01	0:00:01	0:00:01
dfn-bwin	10	8	1176	65	1	27	0	0	0:00:02	0:00:02	0:00:03
dfn-bwin	10	10	1476	155	1	53	0	0	0:00:05	0:00:05	0:00:05
dfn-bwin	10	12	1776	523	1	96	0	0	0:00:10	0:00:10	0:00:10
dfn-bwin	10	14	2076	615	1	115	0	0	0:00:15	0:00:15	0:00:15
dfn-bwin	10	15	2224	483	1	79	0	0	0:00:10	0:00:10	0:00:11
dfn-bwin	10	16	2374	542	1	77	0	0	0:00:02	0:00:02	0:00:11
dfn-bwin	10	18	2670	1086	5	347	2,13	0	0:00:26	0:00:26	0:00:50
dfn-bwin	10	20	2966	971	23	457	5,39	0,83	0:00:02	0:00:02	0:01:12
dfn-bwin	10	25	3710	1201	63	575	6,31	2,28	0:00:08	0:01:23	0:02:15
dfn-bwin	10	30	4454	916	37	437	6,14	0	0:00:02	0:00:06	0:01:50
polska	12	2	492	56	1	24	0	0	0:00:02	0:00:02	0:00:02
polska	12	4	1002	155	1	57	0	0	0:00:00	0:00:00	0:00:05
polska	12	5	1262	573	7	224	2,96	0	0:00:04	0:00:09	0:00:23
polska	12	6	1508	580	3	133	1,68	0	0:00:00	0:00:12	0:00:14
polska	12	8	1976	2259	63	1084	4,5	1,66	0:00:06	0:00:00	0:02:33
polska	12	10	2472	2358	61	1098	4,5	0	0:00:10	0:00:00	0:02:32
polska	12	12	2978	3174	51	1296	5,92	1,64	0:00:00	0:01:26	0:03:49
polska	12	14	3470	4130	51	1752	5,92	1,64	0:00:00	0:00:08	0:06:23
polska	12	15	3716	4668	55	2120	5,92	1,13	0:00:00	0:00:42	0:08:06
polska	12	16	3976	5323	47	2036	4,66	2,41	0:00:11	0:00:11	0:08:34
polska	12	18	4496	3092	17	911	3,46	0	0:00:15	0:00:04	0:03:33
polska	12	20	4988	4334	21	1531	3,46	0	0:00:05	0:01:37	0:06:48
polska	12	25	6232	4384	21	1313	3,46	0	0:00:24	0:01:17	0:09:48
polska	12	30	7490	7205	55	3017	5,71	4,38	0:00:41	0:00:45	0:32:02
nobel-us	14	2	770	25	1	8	0	0	0:00:00	0:00:00	0:00:01
nobel-us	14	4	1584	102	1	20	0	0	0:00:00	0:00:00	0:00:02
nobel-us	14	5	1950	58	1	14	0	0	0:00:00	0:00:00	0:00:02
nobel-us	14	6	2316	124	1	32	0	0	0:00:04	0:00:04	0:00:04
nobel-us	14	8	3048	379	1	64	0	0	0:00:09	0:00:09	0:00:09
nobel-us	14	10	3780	812	1	206	0	0	0:00:02	0:00:02	0:00:24
nobel-us	14	12	4588	1044	1	235	0	0	0:00:02	0:00:02	0:00:32
nobel-us	14	14	5344	824	11	363	4,18	0	0:00:12	0:00:23	0:00:55
nobel-us	14	15	5710	590	11	276	4,18	0	0:00:13	0:00:23	0:00:44
nobel-us	14	16	6114	1172	19	566	5,18	2,56	0:00:17	0:00:21	0:01:47
nobel-us	14	18	6922	1948	17	548	2,62	0	0:00:57	0:02:30	0:02:55
nobel-us	14	20	7736	2294	17	547	1,91	1,91	0:01:29	0:02:29	0:03:31
nobel-us	14	25	9718	1579	1	134	0	0	0:01:43	0:01:43	0:01:43
*nobel-us	14	30	11624	19299	69	6225	3,44	0	0:01:26	2:08:33	3:00:00
newyork	16	2	1020	128	1	72	0	0	0:00:00	0:00:00	0:00:06
newyork	16	4	2040	53	1	29	0	0	0:00:01	0:00:01	0:00:03
newyork	16	5	2550	139	1	59	0	0	0:00:06	0:00:06	0:00:06
newyork	16	6	3130	983	1	182	0	0	0:00:25	0:00:25	0:00:25
*newyork	16	8	4220	24366	263	13570	8,02	5,56	0:01:00	0:18:37	3:00:00

Table 6.3: Branch-and-Price results for realistic instances (1)

Instance	V	K	Path-ini	Path-gen	Nodes	Relaxations	Gap	Gap-Y	TR	TB	TT
*newyork	16	10	5240	23602	211	12149	7,57	5,65	0:02:05	1:24:03	3:00:00
*newyork	16	12	6260	22721	142	10573	7,35	5,42	0:03:33	2:22:59	3:00:00
newyork	16	14	7450	14599	83	5803	4,94	3,96	0:01:32	0:15:14	1:12:02
newyork	16	15	8030	16625	97	6694	5,03	0,72	0:02:38	1:23:55	1:47:26
newyork	16	16	8540	18103	131	6661	5,03	1,4	0:02:02	1:33:01	2:05:26
newyork	16	18	9660	6919	31	1931	1,35	0	0:02:22	0:17:41	0:17:42
newyork	16	20	10820	10277	27	4027	4,05	0	0:03:01	0:14:35	0:37:41
newyork	16	25	13650	8849	31	1569	2,97	0	0:03:08	0:30:19	0:31:00
newyork	16	30	16436	7688	51	1693	3,28	3,28	0:04:48	0:45:07	0:46:13
geant	22	2	2386	33	1	12	0	0	0:00:02	0:00:02	0:00:01
geant	22	4	4922	186	1	25	0	0	0:00:04	0:00:04	0:00:06
geant	22	5	6008	835	1	115	0	0	0:00:21	0:00:21	0:00:23
geant	22	6	7308	1120	1	108	0	0	0:01:09	0:01:09	0:01:09
geant	22	8	9480	4932	1	2308	0	0	0:10:43	0:10:43	0:10:44
geant	22	10	11652	9319	1	5362	0	0	0:30:13	0:30:13	0:30:13
geant	22	12	13824	11491	1	1860	0	0	1:46:30	1:46:30	1:46:31
*geant	22	14	16424	16133	5	2185	11,38	0	1:22:15	1:33:08	3:00:00
*geant	22	15	17724	17207	3	2327	10,16	0	2:24:42	2:11:09	3:00:00
*geant	22	16	19024	17861	3	2866	7,01	0	2:39:22	2:52:46	3:00:00
*geant	22	18	21560	28849	3	5308	6,67	0	1:45:07	2:10:10	3:00:00
*geant	22	20	23732	33268	3	7391	6,92	0	2:04:22	2:46:08	3:00:00
*geant	22	25	29376	14769	1	953	21,78	0	2:15:12	2:36:36	3:00:00
*geant	22	30	35748	49111	1	6782	27,41	0	0:26:01	2:30:19	3:00:00
cost266	37	2	7831	144	1	82	0	0	0:00:03	0:00:03	0:00:11
cost266	37	4	14773	443	1	246	0	0	0:00:08	0:00:08	0:00:38
cost266	37	5	18244	453	1	367	0	0	0:00:51	0:00:51	0:00:51
cost266	37	6	21715	18	1	6	0	0	0:00:28	0:00:28	0:00:28
cost266	37	8	30435	64	1	34	0	0	0:01:09	0:01:09	0:01:09
pioro40	40	2	9298	630	1	162	0	0	0:00:15	0:00:15	0:00:29
pioro40	40	4	19674	4314	1	1026	0	0	2:05:03	2:05:03	2:05:03
*pioro40	40	5	23784	6784	1	764	-	-	3:00:00	3:00:00	3:00:00
*pioro40	40	6	27894	11233	1	2441	-	-	3:00:00	3:00:00	3:00:00
*pioro40	40	8	37192	29181	1	3065	-	-	3:00:00	3:00:00	3:00:00
germany50	50	2	15098	61	1	28	0	0	0:00:00	0:00:00	0:00:11
*germany50	50	4	32034	7443	1	1424	-	-	3:00:00	3:00:00	3:00:00
*germany50	50	5	38664	14159	1	5444	-	-	3:00:00	3:00:00	3:00:00
*germany50	50	6	47132	13412	1	2761	-	-	3:00:00	3:00:00	3:00:00
zib54	54	2	20008	4209	1	3195	0	0	0:00:15	0:00:15	0:07:02
zib54	54	4	35620	2958	1	1610	0	0	0:04:30	0:04:30	0:04:30
zib54	54	5	43426	253	1	91	0	0	0:01:41	0:01:41	0:01:41
zib54	54	6	53430	5461	1	4330	0	0	0:19:01	0:19:01	0:19:01
ta2	65	2	26423	506	1	248	0	0	0:00:33	0:00:33	0:01:06
ta2	65	4	49493	87	1	36	0	0	0:01:35	0:01:35	0:02:24
ta2	65	5	61028	2	1	2	0	0	0:03:19	0:03:19	0:03:20
*ta2	65	6	72563	7017	1	740	-	-	3:00:00	3:00:00	3:00:00

Table 6.4: Branch-and-Price results for realistic instances (2)

Chapter 7

Natural and Extended Formulations

Contents

In this chapter, we propose further integer linear programming formulations to the MSOND problem. The chapter is divided in two main parts. In the first part, we propose a formulation using only the design variables. We prove that this formulation, called also natural formulation, is valid when the number of terminals for each demand is equal to three. For demands with four terminals and more, we show that the formulation of the problem using only natural variables is as difficult as finding a natural formulation for the vertex disjoint paths problem. In the second part, we propose an extended compact formulation using extra families of variables for each demand. The extended formulation is based on a layered view for each demand of the MSOND problem. Depending on the number of its sections (and hence terminals), each demand is represented by layers, where each layer stands for a section of the demand. We give *some experimental results for the extended formulation and propose a method to cut its fractional solutions. We also prove by a column generation procedure that valid inequalities coupling two or more demands are the most efficient to cut fractional points for the extended formulation.*

7.1 Natural formulation

7.1.1 Natural formulation and difficulty

As it was said in Chapter 1, the most natural way to formulate a combinatorial optimization problem $\mathcal{P}$ is to define integer variables y_e for each $e \in E$ and find a suitable set of constraints to represent $\mathcal{F}$. Recall that E represents the basic set of $\mathcal{P}$ and $\mathcal{F}$ is a family of subsets of E, defining the possible solutions of problem $\mathcal{P}$. However, finding a natural formulation is in some cases difficult and can even be impossible. And this is mainly due to the underlying difficulty caused by some constraints and requirements of the problem.

This was actually the case of the MSOND problem, for which finding a natural formulation was a hard task. The principle difficulty of the MSOND problem is the sections' disjunction requirement for each demand. This constraint can be easily managed if we choose to bound the degree of each node of the graph by 2 using a family of binary variables for each demand (see the disjunction inequality (4.2)). However, it is difficult to formulate using only the so-called design variables $y_e, e \in E$.

In what follows, we try to overcome this difficulty and propose a natural integer linear programming formulation for the MSOND problem. Consider the binary design variables $y_e, e \in E$ previously introduced. Recall that $y_e = 1$ if the edge e is installed and 0 otherwise. To formulate the MSOND problem, one of the most intuitive procedures consists in translating its constraints using "known" inequalities from the literature.

Recall that the first requirement of the MSOND problem consists in ensuring, for each demand $k \in K$, two paths routing it and respecting a given order on the terminals of T_k. Note that this amounts to finding a path joining each successive terminals of every demand, which can be formulated using an inequality similar to (4.1), yet written in terms of the design variables y_e (see (7.1)).

$$\sum_{e \in \delta_{G^{k,j}}(W)} y_e \geq 1 \qquad \begin{array}{l} \text{for all}\, k \in K,\ q_j^k = (w_j^k, w_{j+1}^k) \in \mathcal{T}_k, \\ W \subset V^{k,j} :\ w_j^k \in W \, \textit{and} \, w_{j+1}^k \in \overline{W}. \end{array} \tag{7.1}$$

Then, to guarantee the node-disjunction constraint between the sections of a demand, one can think of writing some classical inequalities that have been used to model similar problems in the literature [126]. The first step, is to ensure the edge-disjunction by the following inequality

$$\sum_{e\in\delta_G(W)} y_e \geq 2 \qquad \begin{array}{l} for\, all\, k \in K, for\, all\, W: \\ W \cap T_k \neq \emptyset \neq \overline{W} \cap T^k. \end{array} \tag{7.2}$$

Remark that inequalities (7.2) guarantees two edge-disjoint paths between any pair of terminals of T_k.

Then, we write the inequality that guarantees the sections' node-disjunction requirement. This can be expressed by the following inequality that ensures node-disjoint paths between any pairs of terminals of T_k.

$$\sum_{e\in\delta_{G\setminus u}(W)} y_e \geq 1 \qquad \begin{array}{l} for\, all\, k \in K,\ for\, all\, u \in S_k, \\ for\, all\, W: \\ W \cap T_k \neq \emptyset \neq \overline{W} \cap T_k. \end{array} \tag{7.3}$$

Consider moreover the trivial inequality on the design variables given by

$$0 \leq y_e \leq 1 \quad \text{for all}\, e \in E. \tag{7.4}$$

Now, consider the formulation given by

$$\min\{cy \,|\, y \in \{0,1\}^m : y\, \text{satisfies}\, (7.1) - (7.4)\}. \tag{7.5}$$

Intuitively, one can think that formulation (7.5) is sufficient to guarantee a feasible solution for the MSOND problem since it perfectly translates all the problem's requirements. Unfortunately, this is not the case and this what we will show in what follows.

Consider the example of Figure 7.1.1. This figure illustrates a graph with 6 nodes, three terminals 1, 3 and 5 and three Steiner nodes 0, 2 and 4. The graph represents an integer solution obtained by applying formulation (7.5) for several demands among them demand 1 5 (1 5) (1 3 5).

It is not hard to see that inequalities (7.1), (7.2) and (7.3) are all satisfied (for the considered demand). However, the example represented in Figure 7.1.1 does not constitute a feasible solution for the MSOND problem. In fact, to route demand 1 5

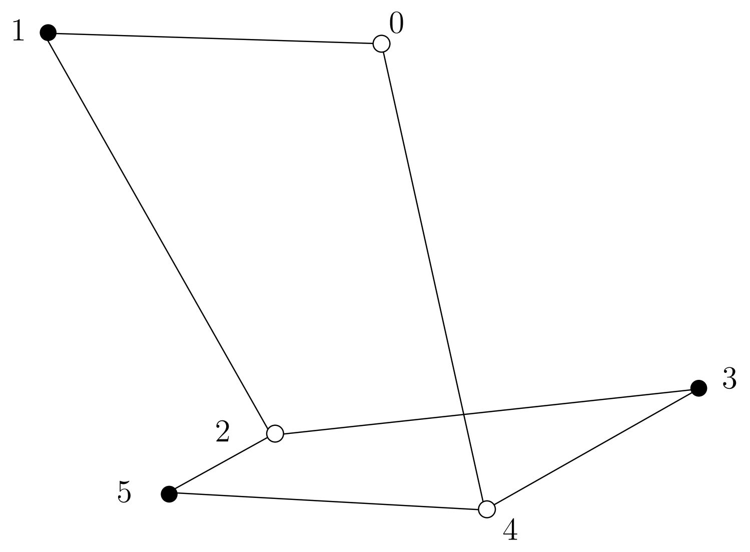


Figure 7.1: Integer non-feasible solution for the MSOND problem

we have two possibilities for each path. The first path can indeed be routed either by $(1, 2, 5)$ or $(1, 0, 4, 5)$ and the second either by $(1, 0, 4, 3, 2, 5)$ or $(1, 2, 3, 4, 5)$. Obviously, all the possible combinations of the two paths lead to an infeasible solution for the problem, since the two paths are not disjoint.

This leads us to think about adding another inequality that enhance the sections' disjunction. To this end, consider a demand $k \in K$ and let $S \subseteq S_k$ be a subset of Steiner nodes of the demand k. Consider a terminal node $t_i \in T_k$ and define the subset $W_{t_i} \subset V$ such that terminal node $t_i \in W_{t_i}$ and its predecessor and successor $t^k_{i-1}, t_{i+1} \in \overline{W_{t_i}}$. We define the *Steiner multi-cuts inequality* as follows

$$\sum_{t_i \in T_k} \sum_{e \in \delta_{G \setminus S}(W_{t_i})} y_e \geq 2(|T_k| - |S|). \tag{7.6}$$

Proposition 7.1 *The Steiner multi-cuts inequality is valid for the MSOND problem.*

Proof Consider a demand $k \in K$, a terminal node $t_i \in T_k$ and let $S \subseteq S_k$ be a subset of Steiner nodes of the demand k. Let us denote by $W_{t_i} \subset V$ the subset of nodes of V containing the terminal node t_i, and by $\overline{W_{t_i}}$ its complementary in V such that $\overline{W_{t_i}}$ contains the super node consisted of the predecessor and successor of t_i, namely t_{i-1} and t_{i+1}. Notice here that, in every feasible solution of the MSOND problem, each terminal is connected by a sub-path to its predecessor and successor. This induces a degree at least equal to 2 for each terminal. And in consequence, each $(t_i, \{t_{i-1}, t_{i+1}\})$-cut calculated in the original graph G must contain at least 2 edges. In particular, each minimum $(t_i, \{t_{i-1}, t_{i+1}\})$-cut contains exactly 2 edges. In this proof, all the considered $(t_i, \{t_{i-1}, t_{i+1}\})$-cuts represent minimum cuts.

- If $|S| \geq |T_k|$, inequality (7.6) is valid and are redundant regarding the trivial inequality (7.4).
- If $S = \emptyset$, this means that $|S| = 0$ and that all the $(t_i, \{t_{i-1}, t_{i+1}\})$-cuts are calculated in the original graph G. In this case, inequality (7.6) can be written as given by (7.7). Since each $(t_i, \{t_{i-1}, t_{i+1}\})$-cut has at least 2 edges, summing over all the terminals T_k, gives a quantity that contains at least $2|T_k|$ edges and hence inequality (7.7) is valid.

$$\sum_{t_i \in T_k} \sum_{e \in \delta_G(W_{t_i})} y_e \geq 2|T_k|. \tag{7.7}$$

- If $0 < |S| < |T_k|$, the proof of validity is given by an induction on $|S|$.
  (i) $|S| = 1$ means that the $(t_i, \{t_{i-1}, t_{i+1}\})$-cuts are calculated in a graph from which we delete only one Steiner node, say s. Due to the sections' disjunction requirement, we know that the Steiner node s is visited just once. This means that s is situated inevitably in the sub-path linking two successive terminals of demand k, say t_j and t_{j+1} (see Figure 7.2). Consequently, when deleting the Steiner node s, we will disconnect the terminals t_j and t_{j+1} and loose 2 from the right hand side of inequality (7.7). Remark here that the value 2 comes from the minimum cuts $(t_j, \{t_{j-1}, t_{j+1}\})$ and $(t_{j+1}, \{t_j, t_{j+2}\})$, since both contains only 1 edge after deleting the steiner s.

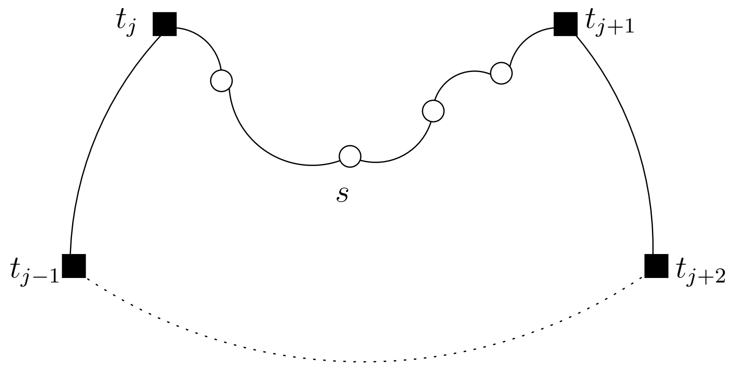


Figure 7.2: Sub-path between t_j and t_{j+1} using the Steiner node s

(ii) Now, assume that inequality (7.6) is valid for $|S| = p-1$ and let us prove its validity for $|S| = p$. Let us denote $S = \tilde{S} \cup \{s\}$ such that $|\tilde{S}| = p-1$. We first begin by deleting the Steiner node s and assume that this disconnect the two successive terminals t_j and t_{j+1}. Then, we contract the two terminals t_j and t_{j+1} so as to get a super terminal that we will call $\tilde{t}_j^k$. Denote by $\tilde{T}_k$ the new set of terminals, $\tilde{T}_k = T_k \setminus \{t_j, t_{j+1}\} \cup \tilde{t}_j^k$. Notice that by deleting the Steiner node s, the left hand side of inequality (7.7), regarding always minimum cuts, is the same before and after contraction of the terminals t_j and t_{j+1} (see (7.8)).

$$\sum_{t_i \in T_k} \sum_{e \in \delta_{G \setminus S}(W_{t_i})} y_e = \sum_{\tilde{t}_i^k \in \tilde{T}_k} \sum_{e \in \delta_{G \setminus \tilde{S}}(W_{\tilde{t}_i^k})} y_e. \tag{7.8}$$

Now, consider the set of terminals $\tilde{T}_k$ and let us delete the set of Steiner nodes $\tilde{S}$. By the induction hypothesis, we can write

$$\sum_{\tilde{t}_i^k \in \tilde{T}_k} \sum_{e \in \delta_{G \setminus \tilde{S}}(W_{\tilde{t}_i^k})} y_e \geq 2(|\tilde{T}_k| - |\tilde{S}|) = 2((|T_k| - 1) - (|S| - 1))$$

and hence

$$\sum_{\tilde{t}_i^k \in \tilde{T}_k} \sum_{e \in \delta_{G \setminus \tilde{S}}(W_{\tilde{t}_i^k})} y_e \geq 2(|T_k| - |S|). \tag{7.9}$$

Finally, combining (7.8) and (7.9) prove the validity of (7.6) for minimum $(t_i, \{t_{i-1}, t_{i+1}\})$-cuts.
Since the result is proved for minimum $(t_i, \{t_{i-1}, t_{i+1}\})$-cuts, it remains always true for the general case of $(t_i, \{t_{i-1}, t_{i+1}\})$-cuts, and the result follows.

□

As a consequence, the natural integer linear programming formulation that we propose to the MSOND problem is given as follows

$$\min\{cy \,|\, y \in \{0,1\}^m : y \,\text{satisfies}\, (7.1) - (7.6)\}. \tag{7.10}$$

In the following, we prove that formulation (7.10) is equivalent to the MSOND problem in case of 3 terminals per demand.

7.1.2 Case of three terminals

Consider the case when each demand has exactly three terminals. We have the following result.

Theorem 7.2 *If $|T_k| = 3$ for all $k \in K$, then the ILP given by (7.10) is equivalent to the MSOND problem.*

Proof In this proof, we will consider the case of only one demand. The result can be after generalized for the other demands.

First, it is not hard to see that every solution of the MSOND problem satisfy all the inequalities of formulation (7.10).

Conversely, consider a solution $\mathcal{S}$ inducing an incidence vector that satisfies (7.10). In what follows, we prove that $\mathcal{S}$ is a solution for the MSOND problem.

To this end, we will distinguish different cases based on the combinatorial signification of the inequalities (7.1)-(7.6). Let us consider inequalities (7.6). Inequalities (7.6) implies the existence of 3 node-disjoint paths, say P_1, P_2 and P_3, joining pairs of consecutive terminals and not passing through the remaining terminals of the demand. Denote by Π the set representing the union of those paths, $\Pi = \{P_1, P_2, P_3\}$. This can be represented by one of the configurations of Figure . This figure illustrates three possibilities:

(a) the three paths are parallel, i.e. they join the same pair of terminals. Without loss of generality, we suppose those terminals to be 2 and 3 (case (a)).

(b) There are two parallel paths. Without loss of generality, we assume that the parallel paths are between 2 and 3 and that the remaining path join 1 to 2(case (b)) .

(c) There are no parallel paths (case (c)).

Consider now inequalities (7.1)-(7.3). These inequalities say that for each terminal t_i, there exists node-disjoint paths between (t_{i-1}, t_i) and (t_i, t_{i+1}). Denote Π' the set representing the union of all those paths. Now, we will consider the different possible configurations for paths Π represented in Figure and introduce the paths Π' ensured by inequalities (7.1)-(7.3) in the same graph. Paths Π' will be represented in the following figures by dotted lines.

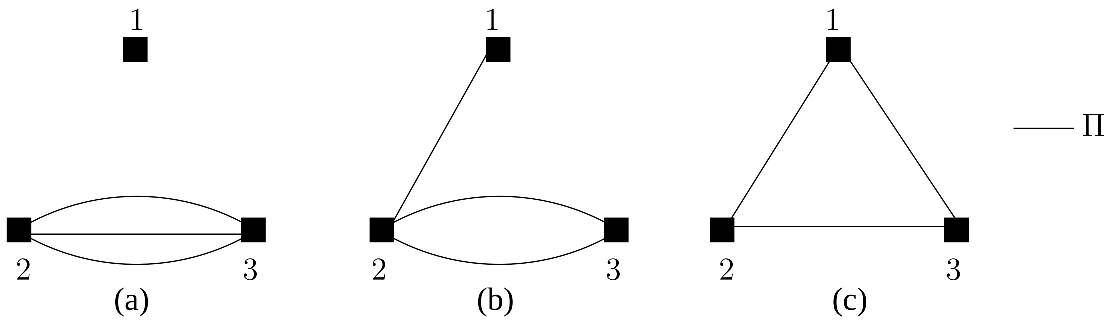


Figure 7.3: Different configurations of paths Π due to inequalities (7.6)

1) Case (a): The three paths are parallel
In this configuration, we know that by inequalities (7.6), there are 3 paths connecting 2 and 3 and so to have a solution, we must ensure one path between 1 and 2 and one path between 1 and 3. Here, one should distinguish three subcases:

- First, suppose that the two paths connecting $(1, 2)$ and $(1, 3)$ intersect both the originals paths between 2 and 3 as shown in Figure 1. The new paths (dotted line) can intersect both the same original path (case (a-2)) or different original paths (case (a-1)). In both cases, we can check that paths Π and paths Π′ induce a feasible solution for the MSOND problem.

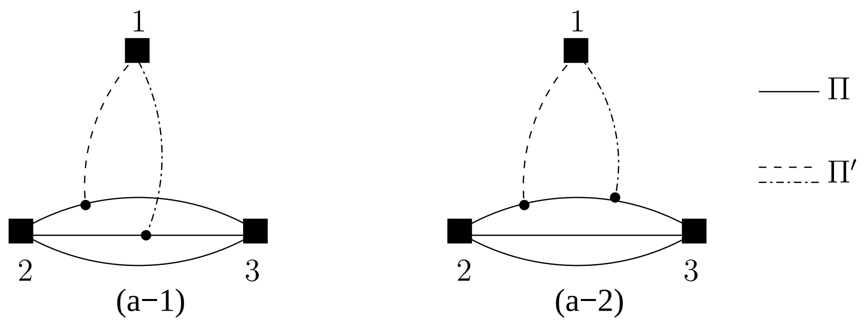


Figure 7.4: All paths in Π are parallel (1)

- Second, suppose now that one of the paths Π′ is joining directly two successive terminals and the other intersect one of the original paths Π between 2 and 3. Case (a-3) of Figure 1 shows a direct path between 1 and 2 and a path between 1 and 3 intersecting one of the original paths Π between 2 and 3. It is not hard to see that this case implies a feasible solution of the MSOND problem.

- Finally, suppose that the new paths are both joining directly two successive terminals, connecting for instance $(1,2)$ and $(1,3)$ (see Figure 1 case (a-4)). In this case, we can trivially check that Π and Π' give a feasible solution for the MSOND problem.

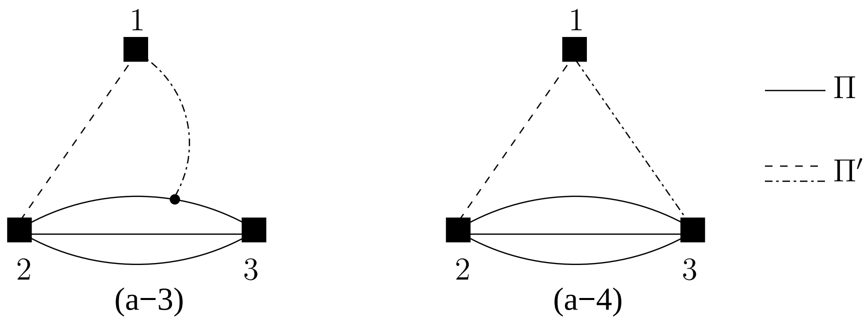


Figure 7.5: All paths in Π are parallel (2)

2) Case (b): Only two from the three original paths Π are parallel
In this case, we distinguish two subcases for the paths Π' due to inequalities (7.1)-(7.3).

- Only one of the new paths Π' does not intersect the original paths (case (b-1)).
  In this case, it is evident that we easily get a solution for the MSOND problem.
- The paths Π' both intersect the original paths Π in different ways (case (b-2),(b-3)).
  Consider the intersections' points between the paths Π and paths Π' as shown in Figure 7.6. Consider in particular the *last leaving* intersection's points between the paths Π' leaving node 1 and the path Π between 1 and 2. These points are called c and d, respectively (see Figure 7.6). Consider also the *first reaching* intersections' points between the paths Π' reaching nodes 2 and 3, respectively. These points are called e and f in the figure. Now, based on those information, we will construct a feasible solution for the MSOND problem.

  Lemma 7.3 *A feasible solution for the MSOND must use both last leaving points but only one of the first reaching points.*

  Proof Suppose that the solution uses only one of the last leaving point, represented in case (b-2) by points c and d. This means that only one path

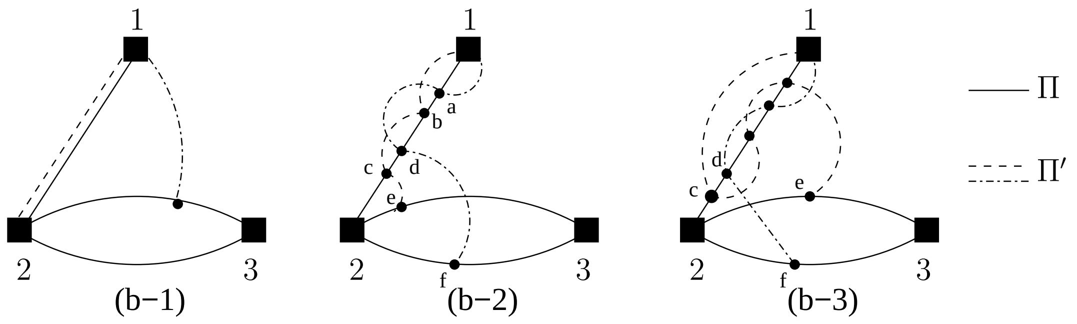


Figure 7.6: Two parallel paths of Π

is leaving node 1 and hence 1 is disconnected from 2 or 3 (depending on the considered path) and so no feasible solution can be obtained.
Now assume that the solution uses both the first reaching points, represented in case (b-2) by points e and f. This means that the path between terminals 1 and 2 is given by $(1, b, d, e, 2)$ (respectively $(1, a, c, 2)$) and the one between 1 and 3 is given by $(1, a, c, f, 3)$ (respectively $(1, b, d, e, 3)$) and hence no more path is connecting 2 and 3, which is infeasible for the MSOND problem. □

By lemma 7.3, to have a feasible solution for the MSOND problem, one must use both leaving points and only one reaching point. Two feasible solution can be considered $((1, b, d, 2); (1, a, c, f, 3); (2, e, 3))$ and
$((1, a, c, 2); (1, b, d, e, 3); (2, f, 3))$. A similar reasoning, applied for other possible configurations such as the case (b-3) of Figure 7.6, implies a feasible solution for the MSOND problem.

Overall, in this case a feasible solution of the MSOND problem can be easily obtained by the combination of paths Π and Π′.

3) Case (c): No parallel edges are considered
This means that the three paths of Π are connecting the different pairs of terminals. Clearly, these paths satisfy also inequalities (7.1)-(7.3). Moreover, it is obvious that Π define a solution for the MSOND problem

As a conclusion, we can deduce that every solution of the formulation (7.10) is a solution for the MSOND problem, and the result follows.

□

In this section, we have restricted ourselves to the case of 3-terminals' demands. In the following section, we will be interested to the case of 4 or more terminals.

7.1.3 Case of four terminals and more

In this section, we will prove that it is difficult to formulate the MSOND problem for a demand having 4 or more terminals.

We first, begin by showing that formulation (7.10) which is equivalent to the MSOND problem in the case of 3-terminal demands, is unfortunately no more sufficient to formulate the problem for demands with 4 terminals and more. For this purpose, consider the example of Figure 7.1.3. This figure 7.1.3 depicts a graph with seven vertices, 4 terminals (numbered $1, 2, 3$ and 4) and three Steiner nodes (nodes $5, 6$ and 7). The graph represents a feasible solution for formulation (7.10). In fact, it is not hard to verify that all the inequalities of this formulation are satisfied by the solution. However, this example does not define a feasible solution for the MSOND problem. In fact, all the combinations of the different possibilities of paths routing the demands' sections imply a violation of the disjunction requirement that must be guaranteed between these sections.

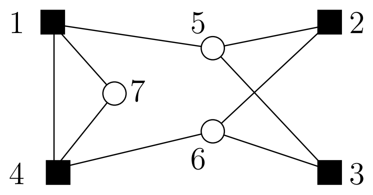


Figure 7.7: Counterexample for Sections Disjunction

Consequently, formulation (7.10) is not sufficient to formulate the MSOND problem. And this show, the difficulty of formulating the MSOND problem using only natural variables when demands have more than 4 terminals.

In the following proposition, we state a result detailing more this difficulty.

Proposition 7.4 *Formulating the single commodity MSOND problem using natural variables is as difficult as formulating the vertex disjoint paths problem.*

Proof Consider the case of single commodity MSOND (i.e. $|K| = 1$) and assume that this demand consists of 4 terminals, say s_1, s_2, t_2 and t_1, that must be visited in that order. Suppose also that we know two paths routing sections $(s_1\, s_2)$ and $(t_2\, t_1)$. In particular, we assume that these paths are given by the edges $s_1 s_2$ and $t_2 t_1$, respectively.

In this case, the MSOND problem is equivalent to the problem of looking for two node-disjoint paths between the pairs of nodes $(s_1\, t_1)$ and $(s_2\, t_2)$. Consequently, finding an integer linear programming formulation for the MSOND problem is equivalent to formulating the 2-vertex disjoint problem. Such a formulation can be easily found using flow or path variables for example. However, to the best of our knowledge, this remains a very hard task using only natural variables. $\square$

Motivated by this difficulty, we tried to find other possible formulations for the MSOND problem, mainly with thinking of new families of variables. We hence propose an integer linear programming formulation given in terms of cuts (see Chapter 4) and a path-based formulation having an exponential number of variables (see Chapter 6). We also think of adding other extra variables to the design variables yielding hence to an extended formulation for the problem. Thinking of extended formulations was motived by other reasons, mainly because these formulations have been proved very efficient in the resolution of huge-sized combinatorial optimization problems. In the next section, we propose an extended formulation for the MSOND problem.

7.2 Extended formulation

To tighten the linear relaxation of an integer linear programming formulation, one of the most efficient techniques is to identify and add new families of valid inequalities. These inequalities help cut fractional points and improve thus the resolution of the problem. Another easier way that generally provides high quality bounds for combinatorial optimization problems is to add extra variables to the original formulation, so as to obtain the so-called extended formulation. Moreover, when the addition of polynomial number of new variables allows a formulation with polynomially many linear inequalities, we say that the problem has a *compact extended formulation*. In this section, we propose a compact extended formulation for the MSOND problem and show that it performs well for its resolution.

7.2.1 The MSOND problem: a view in layers

The idea of the extended formulation came to us from the definition itself of the MSOND problem. As each demand is characterized by independent sections whose routing paths must be node-disjoint, we thought about a formulation across each section. The idea was then to view the single demand as a union of layers (called also levels) of demands. Each level is associated with a section of a demand and represents itself a demand that has to be routed between the extremities of the corresponding section.

In this section, we consider the asymmetric version of the MSOND problem, called *Asymmetric Multilayer Optical Network Design Problem*, AMSOND problem. The AMSOND problem can be defined as follows. Let $G = (V, E)$ be an edge weighted undirected graph and let A be the set of arcs corresponding to the edges' set E, such that with each $e = uv \in E$ are associated two arcs (u, v) and (v, u) in A. We suppose given a set K of demands such that for each demand $k \in K$, we know two dipaths routing it in $G' = (V', E')$ (the IP layer). These dipaths, denoted by $L'_{k,1} = (v^1_{k,1}, ..., v^j_{k,1}, ..., v^{l_{k,1}}_{k,1})$ and $L'_{k,2} = (v^1_{k,2}, ..., v^j_{k,2}, ..., v^{l_{k,2}}_{k,2})$, are node-disjoint.

The AMSOND problem consists in finding a subgraph of G that contains for each demand $k \in K$ two dipaths $L_{k,1}$ and $L_{k,2}$ routing it in G (the optical layer). $L_{k,1}$ and $L_{k,2}$ must be node-disjoint and respect the order of passing through the vertices of G that correspond to the ones visited in $L'_{k,1}$ and $L'_{k,2}$ in G'.

Proposition 7.5 *From every solution of the MSOND problem, we can construct a solution for the AMSOND problem and vice versa.*

Proof Easy. $\square$

In the sequel, we consider the AMSOND problem, and we propose an extended formulation using variables associated with arcs, so as to get better bounds for the MSOND problem. Before introducing the extended formulation, we briefly explain its principle.

The idea of the extended formulation is to consider each demand $k \in K$ as a union of subdemands. Each subdemand is associated with a section $q \in \mathcal{T}_k$ of demand k. Clearly, to route demand $k \in K$, one must route independently every subdemand subject to some constraints. This gives a layered view for the problem that can be explained as follows. For each section $q \in \mathcal{T}_k$ of demand k, we associate a layer (or level), corresponding to a reduced graph in which section q has to be routed. The routing of these subdemands is considered as flow problems and the aggregation of the different layers will then define a routing for demand k.

To better illustrate this concept, consider the example of Figure 7.8. On the left side of the figure is shown an example of a demand between the origin O and the destination

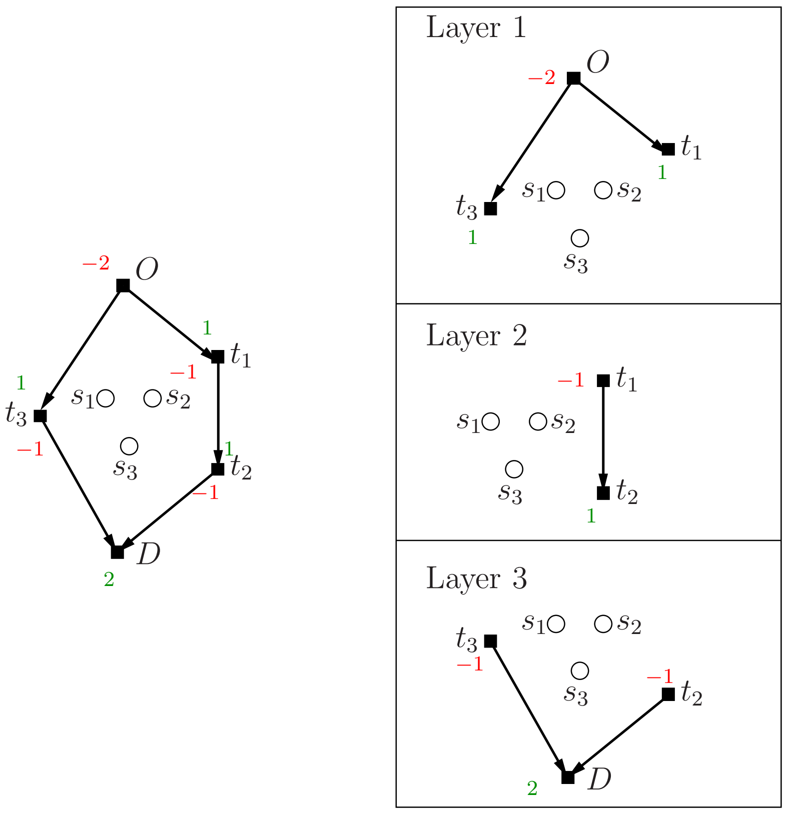


Figure 7.8: Examples of layers of a demand

D. This demand is routed through the dipaths (O, t_1, t_2, D) and (O, t_3, D). The nodes s_1, s_2 and s_3 are Steiner for the considered demand.

As a consequence, the sections of the demand are the following (O, t_1),(t_1, t_2),(t_2, D),(O, t_3) and (t_3, D). From the description given above, we conclude that we will associate 5 subdemands to the demand between O and D. Each subdemand represents a section and is treated as a flow problem. Remark however that the sections adjacent to the origin O and to the destination D can be considered together, allowing hence the diminution of the number of levels. This can be justified as follows. First, recall that each section's routing problem correspond to a flow problem. Note also that near to each terminal, we put two numbers corresponding to the quantities of flow entering and leaving the corresponding terminal, respectively. Normally, for the origin O (resp. the destination D), there is a quantity of 1 leaving it to t_1 (resp. entering to it from t_2) and a quantity of 1 leaving it to t_3 (resp. entering to it from t_3). This can be seen as a whole quantity leaving the origin O (resp. the destination D), of a value exactly equal to 2. Consequently, the sections adjacent to O (resp. D) will be combined in one layer, and we finally get the layered decomposition of the demand on the right side of Figure 7.8. Note, in addition, that for a demand k, each level associated with the section $q = (w_j, w_{j+1}) \in \mathcal{T}_k$ corresponds to the reduced graph $G^{k,j}$. Recall that this graph is obtained from G by deleting all the terminals of T_k except the extremities w_j and w_{j+1}.

Overall, the concept of the extended formulation's layers is the following. The first layer corresponds to the flow going from the origin to the first terminals of the first and second paths respectively. It is hence constituted by the origin O and the two first terminals of each path (here t_1 and t_3). The following layers corresponds to the flow between each two successive terminals of each path (here for instance t_1 and t_2), until arriving to the last layer. This layer is composed of the two last terminals of the paths and the destination D and corresponds to the flow leaving these two terminals and entering to the destination. Note that each layer has a copy of all the Steiner nodes of the corresponding demand.

Routing the demand on two paths is thus no more than a flow problem for each layer of the demand. To the flow problem (flow conservation constraint) we will add inequalities ensuring disjunction of the different sections of the demand (disjunction between layers) and inequalities coupling the flow variables and the design ones. This will be detailed in the next section.

7.2.2 Extended compact formulation

In this section, we propose to give a linear programming formulation of the AMSOND problem. This formulation is obtained using a large, yet polynomial, number of variables and constraints.

For this purpose, we define the following decision variables. Let y_e be the design variable associated with edge $e = ij \in E$. y_e is a binary variable equal to 1 if edge e is installed and 0 otherwise. Define also the binary variable $x_{i,j}^{k,l}$ for each demand $k \in K$, each layer of the demand $l \in L_k$ and for each arc $(i, j) \in A$. Variable $x_{i,j}^{k,l}$ indicates the quantity of flow of the demand k in layer l going from the node i to the node j.

Recall that T_k represents the set of terminals of the demand k, O^k and D^k respectively its origin and destination. We denote by L_k the set of layers of the demand k and by $T_{k,l}$ the set of terminals of demand k in the layer $l \in L_k$.

In addition, we define the flow quantity $b_v^{k,l}$ given as follows

$$b_v^{k,l} = \begin{cases} 0 & \text{if } v \in S_{k,l}, \\ -1 & \text{if } v \in T_{k,l}, v \notin \{O^k, D^k\} \text{ and } v \text{ is an origin}, \\ 1 & \text{if } v \in T_{k,l}, v \notin \{O^k, D^k\} \text{ and } v \text{ is a destination}, \\ -2 & \text{if } v = O^k, \\ 2 & \text{if } v = D^k. \end{cases}$$

Now consider the following ILP.

$$min \sum_{e \in E} c(e) y_e$$

$$\sum_{i:(i,v)\in A} x_{i,v}^{k,l} - \sum_{j:(v,j)\in A} x_{v,j}^{k,l} = b_v^{k,l} \qquad for\, all\; k \in K, l \in L_k, v \in V, \tag{7.11}$$

$$\sum_{l \in L_k} \sum_{j:(j,v)\in A} x_{j,v}^{k,l} \leq 1 \qquad for\, all\; k \in K, v \in S_k, \tag{7.12}$$

$$\sum_{l \in L_k} (x_{i,j}^{k,l} + x_{j,i}^{k,l}) \leq y_e \qquad for\, all\; k \in K, e = ij \in E, \tag{7.13}$$

$$0 \leq x_{i,j}^{k,l} \leq 1 \qquad for\, all\; k \in K, l \in L_k, i \in V, j \in V, \tag{7.14}$$

$$0 \leq y_e \leq 1 \qquad for\, all\; e \in E \tag{7.15}$$

$$x_{i,j}^{k,l} \in \{0,1\} \qquad for\, all\; k \in K, l \in L_k, i \in V, j \in V, \tag{7.16}$$

$$y_e \in \{0,1\} \qquad for\, all\; e \in E. \tag{7.17}$$

Inequalities (7.11) are the *flow conservation inequalities* for each layer and each demand. These inequalities ensure the routing of the demands through two paths. Inequalities (7.12) are the *disjunction inequalities* between the different layers. These inequalities ensure the elementariness and disjunction of the two paths of the demand. Inequalities (7.13) are the *coupling inequalities*, defining the relation between the design and flow variables. These inequalities force the flow variables to be equal to 0, if the design variables are equal to 0 as well. And finally, the inequalities (7.14), (7.25) are the *trivial inequalities* and (7.16) and (7.17) are *integrality inequalities* of the decision variables.

Theorem 7.6 *The integer linear program* (7.11)-(7.17) *is equivalent to the AMSOND problem.*

Proof By the development above, it is clear that the incidence vector of any solution of the AMSOND problem satisfies inequalities (7.11)-(7.17).

No let, $(x, y) \in \{0,1\}^{|K||L_k||V|^2, |E|}$ that does not induce a feasible solution for the AMSOND problem. Assume that (x, y) satisfies inequalities (7.11) and (7.13). In what follows, we prove that there is at least one inequality of type (7.12) that is violated by (x, y). Consider a demand k. By inequalities (7.11), we know that for each layer $l \in L_k$, we guarantee the constraint of flow conservation. This means that we route all the sections of demand k, which yields to the construction of two dipaths passing through the terminals of demand k. Moreover, as the layers are represented by reduced graphs, this implies that the calculated dipaths satisfy the order of passing through the terminals T_k. (x, y) is not feasible for the AMSOND problem implies that there is at least two sections of demand k whose routing paths are not node-disjoint. As each section is represented by a layer, we deduce that there is a Steiner node $v \in S_k$ that is visited in two different layers (or twice for the origin and destination layers grouping two sections). But this means that inequalities (7.12) are violated and the result follows. □

We also identify some valid equations that can strengthen the linear relaxation of the ILP given above.

$$x^{k,l}_{v,j} = x^{k,l}_{i,v} = 0 \quad for\, all\, k \in K, l \in L_k, v \in T_k \setminus T_{k,l}. \tag{7.18}$$

$$x^{k,l}_{v,j} = 0 \qquad \begin{array}{l} for\, all\, k \in K, l \in L_k\,;\, |L_k| = 3, \\ for\, all\, v \in T_{k,l};\, b^{k,l}_v = 1, \\ for\, all\, j \in V. \end{array} \tag{7.19}$$

$$x^{k,l}_{i,v} = 0 \qquad \begin{array}{l} for\, all\, k \in K, l \in L_k\, :\, |L_k| = 3, \\ for\, all\, v \in T_{k,l};\, b^{k,l}_v = -1, i \in V. \end{array} \tag{7.20}$$

Equations (7.18) ensure that the flow entering and leaving a terminal which is not a terminal of the layer is equal to 0. Equations (7.19) guarantee that the flow leaving a "destination" terminal of a layer is 0. And along the same line, equations (7.20) ensure that the flow entering to an "origin" terminal of a layer is equal to 0.

7.2.3 Experimental results

We tested the extended compact formulation introduced in the previous section on realistic instances generated from the SNDlib [5] as described in section 5.2.2.2. The implementation has been done on VBA/Excel using UFF-LP framework **??** as a modeller and Cplex 12.5 as the MIP solver. We set a time limit of execution equal to 3 hours, that is 10800 seconds. Results are reported in Tables 7.2,7.3, 7.4.

The entries of the tables are :

Instance	:	name of the instance;
V	:	number of node in graph G;
K	:	number of demands;
Terminals	:	average number of terminals;
Nsub	:	number of subproblems (nodes);
Opt	:	value of the best upper bound obtained at the root;
Relaxation	:	value of the lower bound;
Gap(%)	:	the relative error between the best upper bound (the optimal solution if the problem has been solved to optimality) and the lower bound obtained at the root;
TT	:	total CPU time (in seconds).

Tables 7.2, 7.3 and 7.4 represent the results obtained by testing the extended formulation on instances with graphs having at least 14 nodes and at most 54 nodes (corresponding to the original nodes of the SNDlib instances). For all the instances we generate demands whose number ranges from 6 to 50. Overall, we tested 110 instances.

It appears from these tables that almost all the tested instances have been solved to optimality within the time limit of execution. In fact, only 18 over 110 (corresponding to only 15%) of the instances did not reach the optimal solution within 3 hours. Moreover, the majority of the instances has a CPU time that does not exceed 1 hour. In particular, 96 over 110 were solved in less than 1 minute. And this proves the efficiency of the extended compact formulation for a fast resolution of instances that may take hours to be solved with other formulations. Furthermore, the tested formulation provides a good bound for linear relaxation, and this bound often coincides with the optimal solution of the AMSOND problem, mainly for "small" instances. In fact, 73 among the instances (about 70%) have been solved to optimality at the root of the Branch-and-Bound tree (i.e. $Nsub = 0$ and gap=0%). Besides, the remaining instances have a good gap between the best upper bound and the lower bound obtained at the root of the Branch-and-Bound tree. Observe that all the instances that have been solved to optimality within the time limit have a gap that does not exceed 6%. And this argue for the high quality of bounds the extended formulation is providing. In addition, we remark that the majority of the instances have been solved along a small Branch-and-Bound tree (i.e. $Nsub \leq 665$), and this also proves the efficiency of our formulation.

From tables 7.2, 7.3 and 7.4, we can also see that the difficulty of an instance is generally related to its size. This first depends on the size of the graph. In fact, the instances of 35 demands have been solved to optimality only for graphs whose number of nodes does not exceed 37. The difficulty of an instance depends also on the number of demands. Look for example to the instances polska and remark that as the number of demands increases, the time of execution grows as well (except instances plska with 20 and 50 demands).

As a consequence, we can deduce that this formulation gives very interesting results and can solve many instances that could not be solved with formulations tested in Chapter 5 and Chapter 6. However, even if the formulation behaves well with "small" instances, when the instances' sizes grow, the resolution become more difficult and sometimes impossible. This leads us to think about adding new valid inequalities to improve our linear relaxation, and this what we will develop in the next sections.

7.2.4 Fractional solutions and valid inequalities

In this section, we study the structure of some fractional points obtained from the resolution of random instances using formulation 7.11- 7.17.

7.2.4.1 First fractional solution

Consider the example illustrated in Figure 7.9. This figure shows a graph with 8 nodes on which are reported the values of a fractional solution. This solution is obtained from the resolution of the linear relaxation of formulation 7.11- 7.17 on an instance defined by 2 demands. The first demand is $(1, 4)$ and is routed by paths $(1\,3\,4)$ and $(1\,2\,0\,4)$. The second demand is $(1, 5)$ and is routed by $(1\,2\,5)$ and $(1\,4\,3\,5)$.

The example of Figure 7.9 can be simplified, mainly by doing some operations of contraction of edges. These operations lead to a simplified fractional point illustrated in Figure 7.10. In the following, we choose to restrict ourselves to the study of the first demand of Figure 7.9. Moreover, as a matter of simplification, we choose to renumber the nodes of the original graph. The equivalent demand of Figure 7.10 is hence between nodes 1 and 3, routed by $(1\,2\,3)$ and $(1\,4\,3)$. This is also equivalent to a demand defined by the circuit $(1\,2\,3\,4\,1)$. Note here that nodes $1, 2, 3$ and 4 are the terminal nodes. Nodes $5, 6$ and 7 are however Steiner nodes for the demand.

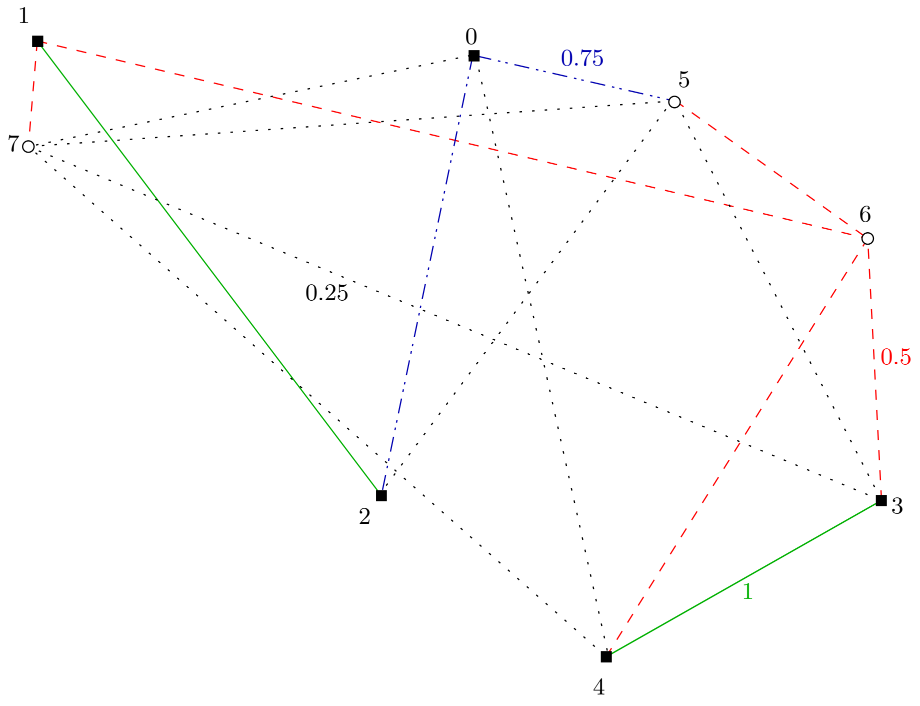


Figure 7.9: First fractional solution

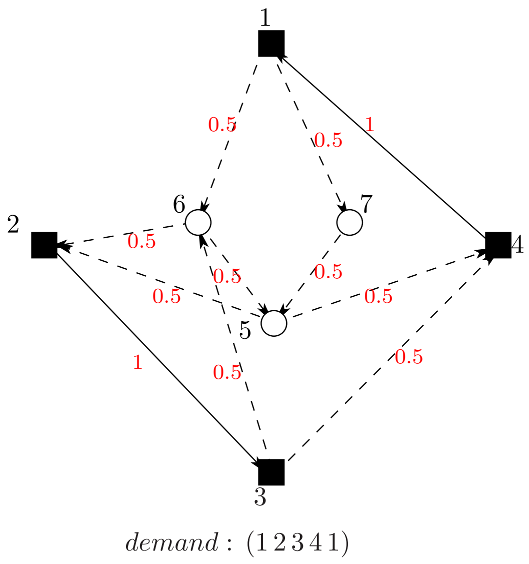


Figure 7.10: Simplification of the first fractional Solution

The fractional solution represented in Figure 7.10 shows an example that violates the sections' disjunction inequality. In fact, all the combinations of the different possibilities of the sections' routing paths generate infeasible solutions for the AMSOND problem.

As a consequence, we thought about cutting this fractional point and ameliorate thus our linear relaxation. Unfortunately, this is not an easy task. In fact, the solution of Figure 7.10 satisfy all the valid inequalities defined in Chapter 4, and that have been proved efficient to cut many fractional solutions (see results of Chapter 5).

This implies that to cut the fractional point of Figure 7.10, one must change the method of separation and make some strategic choices, mainly by answering the following questions:

- Which type of inequality should we generate to cut this fractional point ?
- How to detect easily such a type of violated inequality ?

In the next sections, we try to give some elements of answer to these questions.

7.2.4.2 Types of valid inequalities

The valid inequalities for the extended formulation can be classified according to two criteria. The first one concerns demands and the second is related to variables. Overall, we distinguish the following possible configurations of inequalities.

1) Single demand inequalities: inequalities that are written for each demand independently from the others,
2) Multiple demand inequalities: inequalities that result from considering more than one demand,
3) Extended variables inequalities: inequalities written in terms of x variables,
4) Design variables inequalities: inequalities written in terms of y variables.

In the coming sections, we will try to see if there exists a single demand / design variables violated inequality for the previous fractional solution. To achieve this, we will use a specific technique that we detail in the following paragraphs.

7.2.4.3 Cut finder LP and lifting procedure

Cut finder LP

To cut the fractional solution given above, we use the polarity theory [121]. This technique allows to find the most violated facet-defining inequality that cuts the considered fractional point. This can be done using a linear program defined as follows. Denote the fractional solution by $\overline{y}$. To detect the most violated facet-defining inequality, it suffices to look for an inequality having the form $\alpha y \geq 1$ and which is violated by $\overline{y}$. This can be seen as a linear program in whose decision variables are the components of vector α.

$$\begin{aligned}
&min\, \overline{y}.\alpha \\
&p.\alpha \geq 1 \qquad for\, all\, solution\, p \\
&\alpha \geq 0
\end{aligned} \tag{7.21}$$

The previous linear program 7.21 looks for coefficients α that allows to cut $\overline{y}$ and such that all the solutions p of the problem satisfy the inequality $\alpha y \geq 1$. The resolution of the linear program 7.21 using the values of the fractional point of Figure 7.10 gives the following violated constraint $y_{34} \geq 1$, represented in Figure 7.11.

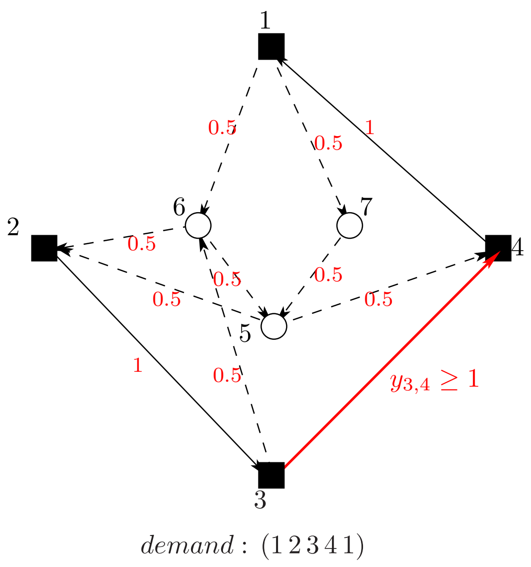


Figure 7.11: Violated constraint

Notice that the valid inequality illustrated in Figure 7.11 appears to be intuitive. In fact, if we remove the edge 34, and try to look for a cycle routing the demand, we can not find a solution satisfying the sections' disjunction. Moreover, this is completely consistent with the theorem of Thomassen and Seymour (see page 65 of [54]).

Theorem 7.7 (Thomassen,Seymour) *(see page 65 of [54]) Let G be a graph such that no single vertex separates s_{i_1} from s_{i_3} and s_{i_2} from s_{i_4}. There are no vertex-disjoint paths joining s_{i_1} to s_{i_3} and s_{i_2} to s_{i_4} if and only if G arises from a planar graph G' where these four vertices are on the outer face in order $s_{i_1}, s_{i_2}, s_{i_3}, s_{i_4}$, by placing an arbitrary graph into some faces of G' bounded by 3 edges.*

In the following, we will propose a lifting procedure to have the general form of this inequality. This enables finding the coefficients of the other edges in the identified constraint.

Lifting procedure

To generalize the valid inequality obtained in the previous section and obtain a cutting constraint, we use the sequential lifting procedure. The idea of this procedure can be described as follows. To the previous fractional solution, we add the missing edge one by one. Each time we insert a new edge, we check if when deleting the edge $(3,4)$ we can have a feasible solution or not. We apply this procedure by first adding edges $(6,7)$, $(1,3)$, $(2,4)$, $(3,7)$ and then the other missing edges of the graph. We therefore obtain the following violated inequality which cut the fractional solution of Figure 7.10.

$$y_{34} + y_{12} + y_{27} + y_{35} + y_{46} + y_{47} \geq 1$$

This lifting procedure allows us hence to detect a violated inequality for the previous fractional solution.
However, detecting and generalizing such an inequality seems not to be an easy task. This lead us to think about the quality of this kind of inequalities. More precisely, one can ask if it is really interesting to investigate on the single demand/design variables cuts. To answer this question, we develop a column generation linear program that will be detailed in the following section.

7.2.4.4 Impact of the different types of valid inequalities

In order to see the impact of the different types of possible valid inequalities on the extended formulation, we implement a column generation process that can be described as follows. The idea of this process is based on the decomposition of the problem having k demands solved by the extended formulation to k subproblems solved each one by the extended formulation and managed by a master problem optimizing the all. In the sequel, in order to simplify notations, we refer to the demands of K as circuits .

The master problem

Let us denote by λ_j^k the circuit j routing demand k ($k \in K, j \in J_k$) and by y_e the design variable ($e \in E$). We will assume that every circuit $j \in J_k$ generated to route demand $k \in K$ satisfy all the constraints of the AMSOND problem for a single demand. That is to say it passes well in order through the terminals T_k of demand k and that it guarantees the sections' disjunction requirement.

The master program is given by the following LP.

$$min \sum_{e \in E} c(e) y_e$$

$$\sum_{j \in J_k} \lambda_j^k = 1 \qquad for\, all\, k \in K, \tag{7.22}$$

$$\sum_{j \in J_k} a_j^k(e) \lambda_j^k \leq y_e \qquad for\, all\, e \in E,\ k \in K, \tag{7.23}$$

$$\lambda_j^k \geq 0 \qquad for\, all\, k \in K,\ j \in J_k, \tag{7.24}$$

$$y_e \geq 0 \qquad for\, all\, e \in E. \tag{7.25}$$

Inequalities (7.22) are the called *circuit inequalities* and they guarantee that for each demand we select a circuit through which this demand will be routed. Inequalities (7.22) are the *linking inequalities*. These inequalities define the relationship that exists between the two types of variables. In particular they say that if an edge e will not be installed (i.e. $y_e = 0$) then it is not possible to select any circuit using this edge to route the demands. Finally, inequalities (7.24) and (7.25) are the *trivial inequalities* corresponding to the decision variables.
It is not hard to see that the linear program defined by (7.22)- (7.25) is equivalent to the AMSOND problem.

The subproblems

Denote by π^k the dual variables associated with inequalities (7.22) and β_e^k the dual variables associated with inequalities (7.23). The reduced cost for a variable λ_j^k, $k \in$

$K, j \in J_k$ is given by the following formula.

$$ReducedCost(\lambda_j^k) = -\pi^k - \sum_{j \in J_k} a_j^k(e)\beta_e^k. \tag{7.26}$$

The subproblem consists hence in finding an interesting circuit variable for a demand k whose reduced cost is negative and subject to some constraints. These constraints should translate the fact that the circuit $j \in J_k$ must pass in order through the terminals of demand k and that the paths routing all the sections of the demand are node-disjoint.

As a consequence, the subproblem associated with a demand $k \in K$ is given by the following ILP.

$$min \sum_{e \in E} \beta_e^k y_e \tag{7.27}$$

$$\sum_{i:(i,v) \in A} x_{i,v}^l - \sum_{j:(v,j) \in A} x_{v,j}^l = b_v^l \quad for\, all\, l \in L_k, v \in V, \tag{7.28}$$

$$\sum_{l \in L_k} \sum_{j:(j,v) \in A} x_{j,v}^l \leq 1 \quad for\, all\, v \in S_k, \tag{7.29}$$

$$0 \leq x_{i,j}^l \leq 1 \quad for\, all\, l \in L_k, i \in V, j \in V, \tag{7.30}$$

$$0 \leq y_e \leq 1 \quad for\, all\, e \in E, \tag{7.31}$$

$$x_{i,j}^l \in \{0,1\} \quad for\, all\, l \in L_k, i \in V, j \in V, \tag{7.32}$$

$$y_e \in \{0,1\} \quad for\, all\, e \in E. \tag{7.33}$$

Observe that given a current solution represented by the primal vector $\overline{y}$ and the dual vector $(\overline{\pi}, \overline{\beta})$, the pricing problem consists in minimizing the function (7.26). As the term $\overline{\pi}^k$ is fixed, the pricing reduces to the minimization of (7.27). Moreover, the variable (circuit) that price out (with a negative reduced cost) must satisfy some constraints. These constraints are reproduced in inequalities (7.28)- (7.33), which correspond to the inequalities of the extended formulation written for a single demand. Recall that inequalities (7.28) ensure the routing of demand k passing in order through its terminals and that inequalities (7.29) guarantee the sections' disjunction constraint.

In the following section, we use this decomposition to show some interesting results.

Computational results

We carry out an experimental study using the extended formulation on one hand, and the formulation described in the previous section on the other hand. Our goal is to see the impact of using the extended formulation to route each demand on its own in a first step, and compare the result to the one we obtain when we apply the extended formulation to the aggregation of all the demands. Experimentations are hold on random instances generated from the TSPlib **??**. The extended formulation and column generation formulation are both implemented on VBA/Excel using UFF-LP framework **??** as modeller and Ceplex 12.5 **??** as a LP solver. Results are reported in Table 7.1. The entries of Table 7.1 are the following, the instance name and size, the values of the linear relaxations of the extended formulation and the column generation respectively, the optimum solution and the gaps of the extended formulation and the column generation relatively to the optimum.

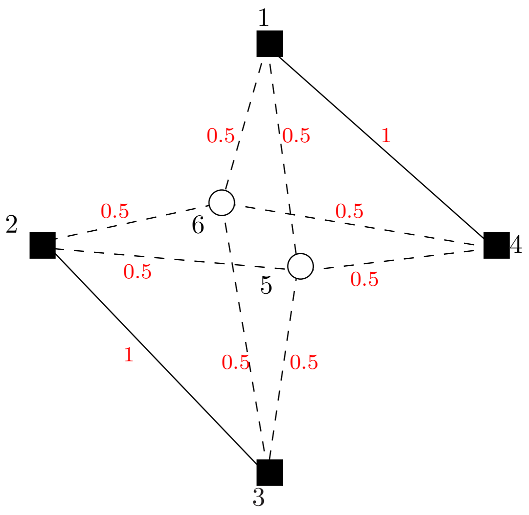
instance	LR-EF	LR-CG	Opt	gap-EF(%)	gap-CG(%)
berlin6_4_2	4180.02	4180.02	4221.71	0.99	0.99
berlin9_6_2	5120.24	5120.86	5175.28	*1.06*	*1.05*
berlin11_6_2	3697.04	3815.61	3936.16	*6.07*	*3.06*
berlin6_4_4	5076.06	5076.06	5101.13	0.49	0.49
berlin9_6_4	4683.19	4683.19	4945.58	5.31	5.31
berlin11_6_4	5801.60	5830.97	5904.04	*1.74*	*1.24*
berlin9_6_6	4531.69	4531.69	4840.85	6.39	6.39
berlin11_6_6	6097.49	6100.29	6379.63	*4.42*	*4.38*
berlin13_8_6	4985.67	4991.96	5428.73	*8.16*	*8.05*

Table 7.1: Column generation experimentations

The comparison of the two gaps shows that for almost all the instances (except for some instances reported in italic in the table), the gaps of the two tested formulations are equal. This implies that the linear relaxation of the extended formulation for the aggregation of $|K|$ demand is very close to the linear relaxation of the formulation considering each demand separately. And hence we can conclude that to cut the fractional points of the extended formulation, it is better to investigate on the *multiple demand/design variables* inequalities, instead of *multiple demand/design variables* ones.

7.2.4.5 Second fractional solution

In the previous section, we proved that the multiple demand/design variables inequalities are more efficient to cut fractional solutions than the single demand/design variables ones. This observation can be endorsed by the example of Figure 7.12. This example illustrates a fractional solution obtained by the linear relaxation of the extended formulation to route two demands.



$demands:$ (1 2 3 4 1), (1 3 2 4 1)

Figure 7.12: Second fractional Solution

Experimentations show that the value of this linear relaxation is exactly the same of the one obtained by the column generation LP previously described. Hence, example of Figure 7.12 can not be cut using the single demand/design variables inequalities. And as a consequence, to cut it one should look for violated multiple demand/design variables inequalities.

Such inequalities are not easy to detect and a deeper investigation of their form is one of the interesting perspectives to this work.

7.3 Concluding remarks

In this chapter, we have studied two linear programming formulations for the MSOND problem. The first one is called natural formulation. We have proved its validity for 3-terminal demands. We have also shown that finding a natural formulation of the MSOND problem become harder with demands having 4 and more terminals. Moreover, finding such a formulation is as difficult as the formulating the notoriously hard vertex disjoint problem. The second formulation is an extended compact one. Based on an experimental study, we have proved that this formulation gives a high-quality bound for the linear relaxation of the MSOND problem. We have also shown by a column generation procedure that new classes of valid inequalities relating many demands at a time are in many cases needed to cut fractional solutions.

Instance	V	K	Terminals	Nsub	Opt	Relaxation	Gap(%)	TT
dfnbwin	10	6	3.29	1	13.134	13.134	0.00	0.04
dfnbwin	10	8	3.33	1	19.878	19.878	0.00	0.06
dfnbwin	10	10	3.31	1	21.249	21.249	0.00	0.05
dfnbwin	10	12	3.39	1	23.214	23.214	0.00	0.06
dfnbwin	10	14	3.50	1	23.214	23.214	0.00	0.09
dfnbwin	10	15	3.45	1	23.766	23.766	0.00	0.20
dfnbwin	10	16	3.48	1	23.766	23.766	0.00	0.40
dfnbwin	10	18	3.50	1	25.699	25.699	0.00	0.39
dfnbwin	10	20	3.25	1	28.535	28.535	0.00	1.33
dfnbwin	10	25	3.53	65	31.04	30.111	2.99	3.53
dfnbwin	10	30	3.56	27	31.04	29.907	3.65	3.04
dfnbwin	10	35	3.20	1	34.038	34.038	0.00	1.93
dfnbwin	10	40	3.56	69	34.844	33.592	3.59	5.13
dfnbwin	10	45	3.53	25	34.844	33.984	2.47	5.19
dfnbwin	10	50	3.17	121	35.315	34.356	2.72	9.13
polska	12	6	3.71	1	24.382	24.382	0.00	0.29
polska	12	8	3.67	1	29.857	29.857	0.00	0.39
polska	12	10	3.69	31	30.791	29.917	2.84	2.13
polska	12	12	3.72	61	32.781	31.375	4.29	3.12
polska	12	14	3.00	63	32.781	31.487	3.95	3.40
polska	12	15	3.65	63	32.781	31.409	4.19	4.96
polska	12	16	3.56	83	33.825	32.487	3.96	5.24
polska	12	18	3.53	65	33.825	33.192	1.87	5.51
polska	12	20	3.75	35	33.825	33.126	2.07	4.34
polska	12	25	3.53	49	33.825	33.187	1.89	9.56
polska	12	30	3.53	167	35.643	34.031	4.52	12.93
polska	12	35	3.80	225	35.643	34.142	4.21	16.96
polska	12	40	3.60	243	37.042	35.268	4.79	22.20
polska	12	45	3.55	253	37.042	34.974	5.58	29.60
polska	12	50	3.67	169	42.711	40.26	5.74	19.45
nobelus	14	6	3.64	1	70.415	70.415	0.00	0.10
nobelus	14	8	3.60	1	70.415	70.415	0.00	0.12
nobelus	14	10	3.63	1	70.415	70.415	0.00	0.15
nobelus	14	12	3.67	1	71.158	71.158	0.00	0.21
nobelus	14	14	3.50	1	85.998	85.998	0.00	1.05
nobelus	14	15	3.75	19	85.998	83.298	3.14	1.71
nobelus	14	16	3.76	11	92.625	89.515	3.36	1.23
nobelus	14	18	3.70	1	129.708	129.708	0.00	2.30

Table 7.2: Extended Formulation results (1)

Instance	V	K	Terminals	Nsub	Opt	Relaxation	Gap(%)	TT
nobelus	14	20	4.00	11	134.125	132.46	1.24	3.22
nobelus	14	25	3.63	1	134.468	134.468	0.00	1.4
nobelus	14	30	3.64	205	155.204	150.456	3.06	15.44
nobelus	14	35	3.80	169	176.383	171.318	2.87	26.18
nobelus	14	40	3.68	53	176.922	172.34	2.59	38.33
nobelus	14	45	3.67	235	177.456	171.847	3.16	50.85
nobelus	14	50	3.67	27	178.633	172.935	3.19	49.19
newyork	16	6	3.36	1	1999.857	1999.857	0.00	0.13
newyork	16	8	3.40	249	2305.955	2156.418	6.48	4.07
newyork	16	10	3.38	113	2316.952	2195.958	5.22	6.82
newyork	16	12	3.44	161	2316.952	2194.895	5.27	9.1
newyork	16	14	3.00	41	2374.92	2282.748	3.88	4.52
newyork	16	15	3.50	81	2416.135	2326.154	3.72	9.32
newyork	16	16	3.56	139	2416.135	2316.453	4.13	10.32
newyork	16	18	3.63	1	2441.615	2441.615	0.00	2.83
newyork	16	20	3.00	73	2510.208	2433.66	3.05	12.19
newyork	16	25	3.60	21	2570.391	2533.927	1.42	23.31
newyork	16	30	3.56	65	2830.762	2797.525	1.17	16.44
newyork	16	35	3.00	11	2830.762	2783.958	1.65	21.57
newyork	16	50	3.52	11	2830.762	2777.96	1.87	66.55
cost266	37	6	3.36	1	38.731	38.731	0.00	1.09
cost266	37	8	3.40	1	38.791	38.791	0.00	1.05
cost266	37	10	3.38	1	47.242	47.242	0.00	2.24
cost266	37	12	3.33	1	47.425	47.425	0.00	3.12
cost266	37	15	3.50	1	65.032	65.032	0.00	6.79
cost266	37	16	3.35	1	65.032	65.032	0.00	10.9
cost266	37	18	3.44	1	84.706	84.706	0.00	15.08
cost266	37	20	3.57	1	84.706	84.706	0.00	27.96
cost266	37	30	3.25	425	105.424	103.368	1.95	2316.31
cost266	37	35	3.60	371	113.084	109.33	*3.32*	10800
cost266	37	50	3.53	81	119.061	109.313	*8.19*	10800
pioro40	40	6	3.43	1	1887.779	1887.779	0.00	2.68
pioro40	40	8	3.53	1	2233.211	2233.211	0.00	4.93
pioro40	40	10	3.50	1	2365.621	2365.621	0.00	20.97
pioro40	40	12	3.50	47	2573.226	2519.782	2.08	204.09
pioro40	40	14	3.50	429	2786.349	2700.343	3.09	1225.47
pioro40	40	15	3.45	33	2791.43	2752.096	1.41	634.06

Table 7.3: Extended Formulation results (2)

Instance	V	K	Terminals	Nsub	Opt	Relaxation	Gap(%)	TT
pioro40	40	16	3.56	453	2893.117	2852.252	1.41	1738.25
pioro40	40	18	3.57	199	3006.622	2978.419	0.94	1686.38
pioro40	40	20	3.75	29	3006.622	2979.993	0.89	1988.64
pioro40	40	25	3.57	163	3661.68	3197.421	*12.68*	10800
pioro40	40	30	3.51	53	4026.504	3492.195	*13.27*	10800
pioro40	40	35	3.60	49	4108.156	3653.95	*11.06*	10800
pioro40	40	40	3.52	21	4287.488	3658.278	*14.68*	10800
pioro40	40	45	3.47	3	4398.286	3777.991	*14.10*	10800
pioro40	40	50	3.50	11	4466.091	3804.168	*14.82*	10800
germany50	50	6	3.67	1	16.724	16.724	0.00	3.65
germany50	50	10	3.68	1	20.643	20.643	0.00	30.26
germany50	50	14	3.61	11	28.765	28.477	1.00	382.37
germany50	50	15	3.50	0	28.958	28.958	0.00	213.52
germany50	50	16	3.60	0	30.747	30.747	0.00	794.19
germany50	50	18	3.64	11	31.276	31.222	0.17	1200.15
germany50	50	20	3.57	45	33.576	33.459	0.35	3234.87
germany50	50	25	3.75	101	36.03	34.247	*4.95*	10800
germany50	50	30	3.60	49	42.131	36.736	*12.81*	10800
germany50	50	35	3.56	21	45.872	42.302	*7.78*	10800
germany50	50	40	3.60	17	51.256	44.549	*13.09*	10800
germany50	50	45	3.54	7	48.105	44.918	*6.63*	10800
zib54	54	6	3.43	1	634.364	634.364	0.00	2.13
zib54	54	8	3.47	1	1392.408	1392.408	0.00	15.93
zib54	54	10	3.56	1	1886.05	1886.05	0.00	20.13
zib54	54	12	3.61	1	1895.839	1895.839	0.00	24.68
zib54	54	14	4.00	1	2063.327	2063.327	0.00	40.94
zib54	54	15	3.60	1	2127.207	2127.207	0.00	48.62
zib54	54	16	3.52	1	2237.833	2237.833	0.00	98.1
zib54	54	18	3.50	39	2407.716	2395.114	0.52	642.33
zib54	54	20	3.50	483	2495.751	2453.184	1.71	9409.76
zib54	54	25	3.60	665	2907.574	2821.139	*2.97*	10800
zib54	54	30	3.62	65	3319.497	2931.859	*11.68*	10800
zib54	54	35	3.4	11	3696.209	3187.12	*13.77*	10800
zib54	54	40	3.62	3	3851.323	3271.347	*15.06*	10800
zib54	54	45	3.58	3	3711.031	3324.196	*10.42*	10800
zib54	54	50	3.50	27	49.659	46.219	*6.93*	10800

Table 7.4: Extended Formulation results (3)

Conclusion

In this thesis, we have studied a survivability problem in the multilayer IP-over-WDM networks. The problem, called the Multilayer Survivable Optical Network Design problem (MSOND problem), consists in determining a survivable topology for the optical WDM layer networks given a secure topology for the logical IP layer.

First, we have considered the Single Commodity Multilayer Survivable Optical Network Design problem (SC-MSOND problem), that is the MSOND problem with only one demand. We have shown that the problem in this case is closely related to the Steiner Travelling Salesman Problem, and is equivalent to the well known k-Vertex Disjoint Paths Problem (k-VDPP). We have then investigated the complexity of the SC-MSOND problem and proved that it is NP-hard when the number of terminals as well as that of Steiner nodes are part of the input. As a consequence, we have obtained that the MSOND problem is NP-hard except for some special cases.

Afterwards, through the thesis, we have studied four integer programming formulations for the MSOND problem. First, we have presented a cut formulation. We have studied the associated polytope, introduced new classes of valid inequalities and given necessary and sufficient conditions under which these inequalities define facets for the polytope. We have also devised separation routines for some of these inequalities. Using these results, we have devised a Branch-and-Cut algorithm. We have then presented a substantial computational study using random and realistic instances. The experimental results show the efficiency of the inequalities and the separation algorithms used in the Branch-and-cut algorithm.

After that, we have discussed a path formulation for the problem. We have shown that the linear relaxation of this formulation is tighter than the one of the cut formulation. We have also shown that the pricing problem reduces to a Shortest Path Problem and can be solved in polynomial time. A suitable branching rule was also discussed and a primal heuristic was then proposed to enhance the resolution of the problem. Using this, we have devised a Branch-and-Price algorithm. The experimental results show that this algorithm performs very well for the tested instances, especially the realistic ones.

Furthermore, we have discussed two further integer linear programming formulations. The first one uses only the design variables. We prove that this formulation is valid when all the demands have exactly 3 terminals. In the case of 4 terminals and more, we have given some insight on how it is difficult to model the sections' disjunction requirements using only the design variables. The second one is an extended formulation. This formulation considers every demand of the MSOND problem as the union of levels. Each level is associated with a section of the demand, and constitutes a demand to be routed as a flow problem. The experimental results show that this formulation gives a tighter linear relaxation for the MSOND problem and that new classes of valid inequalities, involving several demands at a time, are needed to cut fractional solutions.

Now, some questions related to this work deserve to be considered.

From an algorithmic point of view, it would be interesting to improve the separation procedures introduced in this work. One may also investigate the separation problems associated with the TSP-related valid inequalities, i.e., the generalized disjunction inequalities and the Steiner comb inequalities.

From a polyhedral point of view, it would be interesting to better exploit the strong relationship between the MSOND problem and the Steiner TSP. In fact, as the polytope associated with the latter has been completely described for series-parallel graphs [12], one can expect that such a result remains valid also for the MSOND problem. Moreover, thanks to the equivalence with the k-VDPP, an interesting perspective of the current work is to propose an integer linear programming formulation for the k-VDPP and k-EDPP (k-Edge Disjoint Paths Problem). In fact, in spite of being widely studied in terms of complexity and approximation, to the best of our knowledge, these interesting problems have never been studied from a polyhedral point of view. Hence, finding a valid natural formulation for the MSOND problem could be, in this context, exploited to model both the k-VDPP and the k-EDPP and lead a polyhedral investigation on the associated polytopes.

From a practical point of view, many interesting extensions can be considered for the problem. First, it would be interesting to consider the dimensioning of equipments and capacities on the network's nodes and links, respectively. Also, as our problem concerns IP-over-WDM networks, additional technical constraints, dealing with the multilayer aspect, may be considered in further versions of the problem. Moreover, other engineering requirements, related mainly to the quality of service (QoS) and to a better usage of the networks' resources could be interesting to consider in future works.

Bibliography

[1] http://www.informatik.uni-koeln.de/abacus/.

[2] http://www.ilog.com/products/cplex/.

[3] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[4] http://www.tageo.com/index.htm.

[5] http://sndlib.zib.de/home.action.

[6] C. Alves and J. M. Valério de Carvalho. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, 35(4):1315–1328, 2008.

[7] V. Alwayn. *Optical network design and implementation.* Cisco Systems, 2004.

[8] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The traveling salesman problem: a computational study.* Princeton University Press, 2007.

[9] N. Ascheuer, M. Jnger, and G. Reinelt. A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Computational Optimization and Applications*, 17(1):61–84, 2000.

[10] M. Baïou, F. Barahona, and A. R. Mahjoub. Separation of partition inequalities. *Mathematics of Operations Research*, 25(2):243–254, 2000.

[11] M. Baïou, F. Barahona, and A. R. Mahjoub. Progress in combinatorial optimization. In A. R. Mahjoub, editor, *Partition inequalities: Separation, Extensions, and Network Design*, pages 1–39. ISTE, Wiley, 2011.

[12] M. Baïou and A. R. Mahjoub. Steiner 2-edge connected subgraph polytopes on series-parallel graphs. *SIAM Journal on Discrete Mathematics*, 10(3):505–514, 1997.

[13] C.P. Bajaj. Some constrained shortest-route problems. *Mathematical Methods of Operations Research*, 15(1):287–301, 1971.

[14] E. Balas, M. Fischetti, and W. R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68(1-3):241–265, 1995.

[15] F. Barahona and A. R. Mahjoub. On two-connected subgraph polytopes. *Discrete Mathematics*, 147(1-3):19–34, 1995.

[16] C. Barnhart, C. A. Hane, E. L. Johnson, and G. Sigismondi. A column generation and partitioning approach for multi-commodity flow problems. *Telecommunication Systems*, 3(3):239–258, 1994.

[17] C. Barnhart, C. A. Hane, and P. H. Vance. Using Branch-and-Price-and-Cut to solve origin-destination integer multicommodity Flow problem. *Operations Research*, 48(2):318–326, 2000.

[18] C. Barnhart, E. L. Johnson, G. L. Nemhauser, G. L. Savelsberg, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[19] R. Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[20] P. Belotti, A. Capone, G. Carello, and F. Malucelli. Multi-layer MPLS network design: The impact of statistical multiplexing. *Computer Networks*, 52(6):1291–1307, 2008.

[21] W. Ben-ameur, A. R. Mahjoub, and J. Neto. Paradigms of combinatorial optimization. In V. Th. Paschos, editor, *The Maximum Cut Problem*, pages 131–164. ISTE-WILEY, 2010.

[22] F. Bendali, I. Diarrassouba, A.R. Mahjoub, M. Didi Biha, and J. Mailfert. A branch-and-cut algorithm for the k-edge connected subgraph problem. *Networks*, 55(1):13–32, 2010.

[23] S. Borne. *Sécurisation et dimensionnement de réseaux multicouches : modèles et polyèdres*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2006.

[24] S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8(1):25 – 29, 1989.

[25] S. Chopra. The k-edge connected spanning subgraph polyhedron. *SIAM Journal on Discrete Mathematics*, 7(2):245–259, 1994.

[26] S. Chopra and M. R. Rao. The steiner tree problem II: Properties and classes of facet. *Mathematical Programming*, 64(1-3):231–246, 1994.

[27] S. Chopra and M.R. Rao. The steiner tree problem I: Formulations, compositions and extension of facets. *Mathematical Programming*, 64(1-3):209–229, 1994.

[28] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973.

[29] M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *4OR*, 8(1):1–48, 2010.

[30] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[31] G. Cornuéjols, J. Fonlupt, and D. Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33(1):1–27, 1985.

[32] F. R. B. Cruz, G. R. Mateus, and J. M. Smith. A branch-and-bound algorithm to solve a multi-level network optimization problem. *Journal of Mathematical Modelling and Algorithms*, 2(1):37–56, 2003.

[33] W. H. Cunningham. On submodular function minimization. *Combinatorica*, 5(3):185–192, 1985.

[34] G. Dahl, A. Martin, and M. Stoer. Routing through virtual paths in layered telecommunication networks. *Operations Research*, 47(5):693–702, 1999.

[35] G. Dahl and M. Stoer. A Cutting Plane Algorithm for Multicommodity Survivable Network Design Problems. *INFORMS Journal on Computing*, 10(1):1–11, 1998.

[36] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[37] C. De Verdière and A. Schrijver. Shortest vertex-disjoint two face paths in planar graphs. *in STACS*, pages 181–192, 2008.

[38] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*, volume 5. Springer-Verlag New York Incorporated, 2005.

[39] H. Diarrassouba, I.and Kutucu and A. R. Mahjoub. Two node-disjoint hop-constrained survivable network design and polyhedra. Technical report, Cahier de Recherche LAMSADE 332, France, 2013.

[40] M. Didi Biha and A. R. Mahjoub. k-edge connected polyhedra on series-parallel graphs. *Operations Research Letters*, 19(2):71–78, 1996.

[41] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[42] S.E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395–412, 1969.

[43] I. Dumitrescu, S. Ropke, J.F. Cordeau, and G. Laporte. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, 121(2):269–305, 2010.

[44] J. Edmonds. Covers and packings in a family of sets. *Bulletin of the American Mathematical Society*, 68(5):494–499, 1962.

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

[46] T. Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998.

[47] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR: Quartery Journal of Operational Research*, 8(4):407–424, 2010.

[48] B. Feng, A-G. Karasen, P. T. Huth, and B. Slagsvold. State-of-the-art of IP routing. *Telektronikk*, 97(2/3):130–144, 2001.

[49] L. Fleischer. Recent progress in submodular function minimization. *Optima 64. Mathematical Programming Society Newletter*, 2000.

[50] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

[51] S. Fortune, J. E. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[52] B. Fortz, T. McCormick, A. R. Mahjoub, and P. Pesneau. Two-edge connected subgraphs with bounded rings: Polyhedral results and Branch-and-Cut. *Mathematical Programming*, 105(1):85–111, 2006.

[53] B. Fortz and M. Poss. An improved benders decomposition applied to a multi-layer network design problem. *Operations Research Lettrs*, 37(5):359–364, 2009.

[54] András Frank. *Packing paths, circuits and cuts: a survey.* Forschungsinst. für Diskrete Mathematik, 1988.

[55] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

[56] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman New York, 1979.

[57] N. Ghani, S. Dixit, and T. Wang. On IP-over-WDM integration. *Communications Magazine, IEEE*, 38(3):72–84, 2000.

[58] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[59] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem-part II. *Operations Research*, 11(6):863–888, 1963.

[60] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery*, 35(4):921–940, 1988.

[61] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[62] L. Gouveia, P. Patrício, and A. de Sousa. Hop-constrained node survivable network design: An application to MPLS over WDM. *Networks and Spatial Economics*, 8(1):3–21, 2008.

[63] L. Gouveia, P. Patrício, A. de Sousa, and R. Valadas. MPLS over WDM network design with packet level QoS constraints based on ILP models. In *INFOCOM*, 2003.

[64] L. Gouveia and P. Pesneau. On extended formulations for the precedence constrained asymmetric traveling salesman problem. *Networks*, 48(2):77–89, 2006.

[65] M. Grötschel. On the symmetric travelling salesman problem: solution of a 120-city problem. *Mathematical Programming Study*, 12:61–77, 1980.

[66] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[67] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization.* Berlin [u.a]: Springer4060 XII, 362 S, 1988.

[68] M. Grötschel and C. L. Monma. Integer polyhedra arising from certain networks design problems with connectivity constraints. *SIAM Journal of Discrete Mathematics*, 3(4):502–523, 1990.

[69] M. Grötschel, C. L. Monma, and M. Stoer. Polyhedral approaches to network survivability. In F. Hwang F. Roberts and C. Monma, editors, *Reliability of Computer and Communication Networks*, volume 5 of *Series Discrete Mathematics and Computer Science*, pages 121–141. AMS/ACM, 1991.

[70] M. Grötschel, C. L. Monma, and M. Stoer. Facets for polyhedra arising in the design of communication with low-connectivity constraints. *SIAM Journal on Optimisation*, 2(3):474–504, 1992.

[71] M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem i: inequalities. *Mathematical Programming*, 16(1):265–280, 1979.

[72] Q.P. Gu and S. Peng. Algorithms for node disjoint paths in incomplete star netwoks. In *Proceedings of the 1994 International Conference on Parallel and Distributed System*, pages 296–303, 1994.

[73] D. Gusfield. Very simple algorithms and programs for all pairs network flow analysis. Technical report, Computer Science Division, University of California, Davis, 1987.

[74] D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal of Computing 19*, 19(1):143–155, 1990.

[75] D. Huygens, M. Labbé, A. R. Mahjoub, and P. Pesneau. The two-edge connected Hop-constrained network design problem: Valid inequalities and branch-and-cut. *Networks*, 40(1):116–133, 2007.

[76] D. Huygens, A. Mahjoub, and P. Pesneau. Two edge-disjoint hop-constrained paths and polyhedra. *SIAM Journal on Discrete Mathematics*, 18(2):287–312, 2004.

[77] T. Ibaraki. Algorithms for obtaining shortest paths visiting specified nodes. *SIAM*, 15(2):309–317, 1973.

[78] S. Iwata. Submodular function minimization. *Mathematical Programming*, 112(1):45–64, 2008.

[79] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[80] S. Iwata and J. B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237. Society for Industrial and Applied Mathematics, 2009.

[81] T. Jensen. Internet protocol and transport protocols. *Telektronikk*, 97(2/3):20–38, 2001.

[82] R. Kalaba. On some communication network problems. Technical report, DTIC Document, 1959.

[83] R. M. Karp. *Reducibility among combinatorial problems.* Springer, 1972.

[84] R. M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[85] K. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424 – 435, 2012.

[86] H. Kerivin and A. R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[87] A. Knippel and B. Lardeux. The multi-layered network design problem. *European Journal of Operational Research*, 183(1):87–99, 2007.

[88] Y. Kobayashi and C. Sommer. On shortest disjoint paths in planar graphs. *Discrete Optimization*, 7(4):234–245, 2010.

[89] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[90] G. Laporte, H. Mercure, and Y. Nobert. Optimal tour planning with specified nodes. *RAIRO, Recherche Opérationnelle*, 18(3):203–210, 1984.

[91] A. N. Letchford, S. D. Nasiriy, and D. O. Theisz. Compact formulations of the steiner traveling salesman problem and related problems. *Online Draft*, March 2012.

[92] C. L. Li, T. McCormick, and D. Simchi-Levi. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26(1):105–115, 1990.

[93] K. H. Liu. *IP Over WDM*. Wiley, 2003.

[94] M. E Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[95] A. R. Mahjoub. Two edge connected spanning subgraphs and polyhedra. *Mathematical Programming*, 64(1-3):199–208, 1994.

[96] A. R. Mahjoub. On perfectly two-edge connected graphs. *Discrete Mathematics*, 170(1-3):153–172, 1997.

[97] A. R. Mahjoub. Polyhedral approaches. In *Concepts of Combinatorial Optimization, Volume 1*, pages 261–324. Wiley Online Library, 2010.

[98] A. R. Mahjoub and C. Nocq. On the linear relaxation of the 2-node connected subgraph polytope. *Discrete Applied Mathematics*, 95(1-3):389–416, 1999.

[99] A. R. Mahjoub and P. Pesneau. On the steiner 2-edge connected subgraph polytope. *RAIRO - Operations Research*, 42(3):259–283, 2008.

[100] E. Mannie. Generalized multi-protocol label switching (GMPLS) architecture. *Interface*, 501:19, 2004.

[101] S. T. McCormick. Submodular function minimization. In *The Handbook on Discrete Optimization, Elsevier, K; Aardal, G; Nemhauser, and R. Weismantel, eds., 321-391*. 2008.

[102] K. Menger. Zur allgemeinen kurventheorie. *Fundamanta Mathematicae*, 10(1):96–115, 1927.

[103] C. St. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, s1-36(1):445–450, 1961.

[104] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[105] S. Orlowski and M. Pióro. Complexity of column generation in network design with path-based survivability mechanisms. *Networks*, 59(1):132–147, 2012.

[106] S. Orlowski, C. Raack, A. M. C. A. Koster, G. Baier, T. Engel, and P. Belotti. Branch-and-cut techniques for solving realistic two-layer network design problems. In Arie Koster and Xavier Muoz, editors, *Graphs and Algorithms in Communication Networks*, Texts in Theoretical Computer Science. An EATCS Series, pages 95–118. Springer Berlin Heidelberg, 2010.

[107] M. W. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review 33*, pages 60–100, 1991.

[108] IETF. 1981. J. Postel. *Internet Protocol. DARPA Internet Program Protocol Specification*, (RFC791).

[109] IETF. 1981. J. Postel. *Transmission Control Protocol. DARPA Internet Program Protocol Specification*, (RFC793).

[110] R. C. Prim. Shortest connection networks and some generalization. *Bell System Technical Journal*, 36:1389–1401, 1957.

[111] S. Raghavan and D. Stanojevic. Branch and price for wdm optical networks with no bifurcation of flow. *INFORMS Journal on Computing*, 23(1):56–74, 2011.

[112] S. Ramamurthy and B. Mukherjee. Survivable WDM mesh networks. II. restoration. In *Communications, 1999. ICC'99. 1999 IEEE International Conference on*, volume 3, pages 2023–2030. IEEE, 1999.

[113] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee. Survivable WDM mesh networks. *Journal of Lightwave Technology*, 21(4):870, 2003.

[114] N. Robertson and P. D. Seymour. Graph minors. XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[115] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. 2001. Network Working Group. RFC 3031.

[116] K.S. Ruland and E.Y. Rodin. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & amp; Mathematics with Applications*, 33(12):1 – 13, 1997.

[117] D.M. Ryan and B.A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling.* North-Holland, Amsterdam, 1981.

[118] J.P. Saksena and S. Kumar. The routing problem with 'k' specified nodes. *Operations Research*, 14(5):909–913, 1966.

[119] J. J. Salazar-González. The steiner cycle polytope. *European Journal of Operational Research*, 147(3):671–679, 2003.

[120] S. Salsano, F. Ricciato, M. Listanti, and A. Belmonte. Off-line configuration of a MPLS over WDM network under time-varying offered traffic. In *INFOCOM*, 2002.

[121] A. Schrijver. *Theory of linear and integer programming.* Wiley, 1998.

[122] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[123] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Verlag, 2003.

[124] S. Sengupta, V. Kumar, and D. Saha. Switched optical backbone for cost-effective scalable core ip networks. *Communications Magazine, IEEE*, 41(6):60–70, 2003.

[125] M. Steinová. Approximability of the minimum steiner cycle problem. *Computing And Informatics*, 29(6):1349–1357, 2010.

[126] M. Stoer. *Design of survivable networks.* Lectures Notes in Mathematics 1531. Springer-Verlag, 1992.

[127] T. Tholey. Improved algorithms for the 2-vertex disjoint paths problem. *M. Nielsen et al. (Eds.): SOFSEM, LNCS 5404*, pages 546–557, 2009.

[128] E. Uchoa. Progress in combinatorial optimization. In A. R. Mahjoub, editor, *Cuts over Extended Formulations by Flow Discretization*, pages 255–279. ISTE, Wiley, 2011.

[129] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200, 1997.

[130] F. Vanderbeck. *Decomposition and column generation for integer programming.* PhD thesis, Université Catholique de Louvain, Belgium, 1994.

[131] F. Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128, 2000.

[132] F. Vanderbeck. Implementing mixed integer column generation. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 331–358. Springer US, 2005.

[133] T. Volgenant and R. Jonker. On some generalizations of the traveling-salesman problem. *Journal of Operational Research Society*, 38(11):1073–1079, 1987.

[134] J. Y. Wei. Advances in the management and control of optical internet. *Selected Areas in Communications, IEEE Journal on*, 20(4):768–785, 2002.

[135] P. Winter. Generalized Steiner problem in series-parallel networks. *Journal of Algorithms*, 7(4):549–566, 1986.

[136] P. Zhang and W. Zhao. On the complexity and approximation of the min-sum and min-max disjoint paths problems. *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies. LNCS 4614*, pages 70–81, 2007.