UNIVERSITY OF SALERNO
Departments of Mathematics and Physics

PhD in Mathematics, Physics and Applications
Curriculum: Mathematics
CYCLE XXXII

PhD Thesis

Polyhedral Analysis and Branch and Cut Algorithms for Some NP-hard Spanning Subgraph Problems

Tutor:
Prof. Raffaele Cerulli

Doctoral dissertation of:
Federica Laureana

Co-Tutor:
Prof. Francesco Carrabs

Coordinator:
Prof. Carmine Attanasio

2018/2019

Alla mia famiglia e a Michele

Acknowledgements

I thank my tutor, Prof. Raffaele Cerulli, for being a point of reference, for having believed in me and for having guided and encouraged me with his valuable advices. I also thank the co-tutor of this thesis, Prof. Francesco Carrabs, for the time he has dedicated to me and for always being available to help me and spur me to pursue my goals.

Many thanks to all the members of my group, Ciro, Andrea, Rosa and Carmine: the time that I spent with you and your great support have been essential for me.

I thank my wonderful parents, who have always been by my side and have affectionately accompanied me through this journey. It is not possible to express in these few lines how lucky I feel for the family they have built for me and for my sister. I thank my beautiful sister, her support has never made me feel alone, not even miles away, filling my life with joy and affection.

Finally, I thank Michele, who never stopped believing in me, often more than I did myself. He made me understand how much strength love gives and this milestone is as mine as his.

Ringrazio il mio tutor, il Prof. Raffaele Cerulli, per essere un punto di riferimento, per aver creduto in me e avermi guidata e incoraggiata con i suoi preziosi consigli. Ringrazio inoltre il co-tutor di questa tesi, il Prof. Francesco Carrabs, per il tempo dedicatomi e per essere sempre stato disponibile ad aiutarmi e spronarmi a perseguire i miei obiettivi.

Ringrazio tutti i componenti del mio gruppo, Ciro, Andrea, Rosa e Carmine: il tempo trascorso con voi e il vostro sostegno sono stati per me di fondamentale importanza.

Ringrazio i miei meravigliosi genitori, che sono sempre stati al mio fianco e con amore mi hanno accompagnata in questo percorso. È impossibile esprimere in poche righe quanto io mi senta fortunata per la famiglia che hanno costruito per me e mia sorella. Ringrazio mia sorella, il cui sostegno non mi ha mai fatta sentire sola, neanche a miglia di distanza, riempiendo la mia vita di gioia e amore.

Infine, ringrazio Michele, che non ha mai smesso di credere in me, spesso più di quanto facessi io stessa. Mi ha fatto capire quanta forza dia l'amore e questo traguardo è mio quanto suo.

Contents

List of Figures

List of Tables

Introduction

There is a widespread need to use a quantitative approach for the solution of decision problems that arise in many different areas of real life. The goal is to choose which decisions to take to manage a real system as efficiently as possible using mathematical tools. *Operations Research* provides a scientific basis to try to analyze and understand situations even with very complex structures and then use the gathered information to predict the behaviour of a system and improve the performance of the system itself. The analysis of a real problem occurs in two phases: the representation of the problem through a *mathematical model* and the development of efficient *mathematical methods* to determine an optimal solution of the problem or a good approximation of it. Therefore, the Operations Research is the science that deals with giving a unitary context to mathematical and computer science concepts and that starting from theoretical bases arrives at the construction of concrete models and their solution. *Combinatorial optimization problems* are those in which mathematical techniques are applied to find optimal solutions within a discrete set of possible solutions. Many combinatorial optimization problems are defined on graphs and are *hard* to solve, which means that no polynomial time algorithm exists for them. To solve these problems it can be used *heuristic approaches*, which aim to return good solutions in a reasonable time, or *exact approaches*, which return the optimal solutions and are often based on implicit enumeration techniques.

This dissertation involves the study of two problems defined on graphs: the *Generalized Minimum Branch Vertices (GMBV) problem* and the *2-Edge-Connected Minimum Branch Vertices (2ECMBV) problem*. Both problems

aim to identify a subgraph of a given graph satisfying some feasibility conditions and for which is minimum the number of *branch vertices*, namely vertices with degree greater than two. Branch vertices have a very important role in the design of optical networks. Indeed, when in an optical network the signal enters a node having degree greater than two, it has to be split by a switch. For reasons related to cost containment, it is necessary to minimize the use of switches within the network, and thus minimize the number of branch vertices. Below are the definitions of the problems.

- Let $G = (V, E)$ be an undirected graph, where the set V is partitioned into k clusters, $V_1, ..., V_k$. The GMBV problem consists of finding a tree in G spanning exactly one vertex for each cluster and with the minimum number of branch vertices. This problem is NP-hard, indeed when each cluster is a singleton it reduces to the well-known *Minimum Branch Vertices problem*.
- Given an undirected graph $G = (V, E)$, the 2ECMBV problem consists of finding a spanning 2-edge-connected subgraph in G with the minimum number of branch vertices. Let us recall that a subgraph is 2-edge-connected if there exist at least two edge-disjoint paths between any pair of vertices. This problem is NP-hard, indeed finding an optimal solution to the 2ECMBV problem on a graph G in polynomial time is equivalent to establishing in polynomial time whether G is Hamiltonian.

This thesis is organized as follows.
Chapter 1 provides an overview of some basic concepts on combinatorial optimization, graph theory, polyhedral analysis and a brief description of the Branch and Cut algorithm.
Chapter 2 describes some Network Design Problems related to those handled in this thesis: the *Minimum Spanning Tree problem*, *Generalized Network Design Problems* and the *survivability requirements* in Network Design Problems.
In Chapter 3 we show some sufficient and necessary conditions for a graph to be Hamiltonian and we devise a procedure for the generation of a family

of 3-connected non-Hamiltonian graphs.
In Chapter 4 we introduce an integer linear programming formulation for the GMBV problem. Furthermore, we derive some properties regarding feasible GMBV solutions and we design a procedure to identify and remove useless vertices, namely vertices that do not belong to any feasible solution. We determine the dimension of the polyhedron of integer solutions as well as some valid inequalities and prove some facet results. We develop a Branch and Cut algorithm and computational tests are carried out on a set of 675 instances. Computational results show that the Branch and Cut algorithm optimally solve almost the 80% of the instances in 7 minutes.
In Chapter 5 we model the 2ECMBV problem as an integer linear program. We derive the dimension of the polyhedron of integer solutions, propose new classes of valid inequalities and we prove that some of them are facet-defining. We solve the 2ECMBV problem by a Branch and Cut algorithm. The computational tests are conducted over a set of 3-connected non-Hamiltonian graphs, generated as described in Chapter 3. Computational results show the effectiveness of the valid inequalities proposed in this dissertation.
Final remarks on the presented problems and future work projects are reported at the end of this thesis.

Chapter 1

Basic Concepts: Combinatorial Optimization and Polyhedral Theory

An *optimization problem* consists of maximizing or minimizing a function with respect a set representing the range of choices available in a certain situation. The function compares the possible choices to determine which is best. More formally, given a space X, a set $S \subseteq X$ and a function $f : S \to \mathbb{R}$, an optimization problem (OP) (in the minimum form) can be formulated as

$$min_{x \in S} f(x)$$

It consists of determining, if exists, $x^* \in S$ such that $f(x^*) \leq f(x)$, for any $x \in S$. The function f is called *objective function* while the set S is the *feasible set*. A point x in S is a *feasible solution* to the OP. Commonly, the space X coincides with the space $\mathbb{R}^n$ or with the space $\mathbb{Z}^n$, and the corresponding optimization problems are very different, both from the point of view of the theoretical characterizations and for the design of solution algorithms.

Combinatorial optimization studies optimization problems in which the feasible set is defined in terms of combinatorial structures, thus in such problems the set of feasible solution S is discrete. Examples of combinatorial optimization problems are the knapsack problem, the minimum spanning tree problem and the travelling salesman problem. In this thesis we deal with *integer linear programming (ILP)* problems, namely problems where the objective function is linear, the feasible set is expressed by linear

inequalities and the decisional variables are integer.
In this chapter we introduce some basic concepts that will be used in the following: an introduction to the graph theory, the main concepts of the polyhedral theory and a brief description of the Branch and Cut algorithm.

1.1 Graph Theory

A *graph* is a set of points, called *vertices*, connected by lines, named *edges*. The importance of this structure is due to the observation that several real-world problems can be modelled by graphs, such as the problems related to the transportation and communication networks. The historical origin of the theory is generally traced back to a work developed by Euler in 1736 concerning the answer to a famous mathematical question, known as the problem of the *Königsberg bridges*. Königsberg was a city in Prussia and one of the main cultural and political centers of Germany. The city is located on the banks of the Pregel river and includes two islands which were connected by seven bridges. The problem was to determine whether it was possible to plan a walk through the city that crossed each of those bridges once. Euler build a graph replacing each river bank and each island with a vertex, and each bridge with an edge, reducing the process of solving the problem into the analysis and topological study of a graph. Euler showed that the possibility of a walk through a graph traversing each edge exactly once depends on the number of edges touching the vertices: he proved that such a walk exists if the graph is connected and every vertex is touched by an even number of edges, and a graph that satisfies this property is called *Eulerian graph*.
A graph G is represented by a pair of sets (V,E), where $V = \{v_1,...,v_n\}$ is a non-empty discrete finite set and the elements in V are the *vertices* of the graph, while $E = \{e_1,...,e_m\}$ is the set of *edges* of the graph and contains pairs of vertices of G. If the pairs of vertices are ordered, the graph is said *directed*, if not the graph is said *undirected*. In what follows, we will refer to undirected graphs. Any edge $e_k \in E$ is a pair of vertices in V, namely $e_k = \{v_i, v_j\}$ with $v_i, v_j \in V$, and we say that e_k is *incident* on v_i and v_j. Two vertices $u, v \in V$ are *adjacent* if the edge $\{u,v\}$ belongs to E, while we say that two edges $e, f \in E$ are *adjacent* if they have a vertex in common. Given a vertex $v \in V$, the set $N(v) = \{u \in V : \{u,v\} \in E\}$ is the *neighborhood* of v in G. Furthermore, given $v \in V$, we denote by $\delta(v)$ the set of the edges incident on v,

and $|\delta(v)|$ is the *degree* of v. Given a graph $G = (V,E)$, the graph $G' = (V',E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$ such that if $\{u,v\} \in E'$ then $u,v \in V'$. A graph G is called *bipartite* if there exists a partition of V into sets V_1 and V_2 such that each edge in G joins a vertex in V_1 to a vertex in V_2. G is *complete* if it contains all the possible edges, namely $|\delta(v)| = m-1$, for any $v \in V$. A sequence of vertices $v_1,...,v_k$ such that $\{v_{i-1},v_i\} \in E$ for any $i = 2,...,k$, is a *path* of length k. A path is *simple* if each vertex is crossed exactly once. A simple path in which the first and last nodes coincide is called *cycle* of length k. A vertex $u \in V$ is *connected* to a vertex $v \in V$ if there exists a path in G between u and v. Finally, a graph is *connected* if all its vertices are connected to each other.

1.2 Polyhedral Theory

Let $\mathbb{R}^n$ be the set of n-dimensional vectors.

Definition 1.2.1. *A point $x \in \mathbb{R}^n$ is a linear combination of the points $x_1,...,x_k \in \mathbb{R}^n$ if there exist $\lambda_1,...,\lambda_k \in \mathbb{R}$ such that $x = \sum_{i=1}^k \lambda_i x_i$. A linear combination such that $\sum_{i=1}^k \lambda_i = 1$ is an affine combination. An affine combination such that $\lambda_1,...,\lambda_k \geq 0$ is a convex combination.*

A subset $A \subseteq \mathbb{R}^n$ is an *affine space* if it is closed under taking affine combinations. The inclusionwise minimal affine space containing a set $S \subseteq \mathbb{R}^n$ is called the *affine hull* of S and is denoted by $aff(S)$. Let us now introduce the notions of linear independent and affine independent vectors.

Definition 1.2.2. *A set of points $x_1,...,x_k \in \mathbb{R}^n$ is linear independent if the unique solution to $\sum_{i=1}^k \lambda_i x_i = 0$ is $\lambda_i = 0$, for any $i = 1,...,k$.*

Definition 1.2.3. *A set of points $x_1,...,x_k \in \mathbb{R}^n$ is affinely independent if the unique solution to $\sum_{i=1}^k \lambda_i x_i = 0$ and $\sum_{i=1}^k \lambda_i = 0$ is $\lambda_i = 0$, for any $i = 1,...,k$.*

Let us note that two distinct points are always affinely independent, while three points are affinely independent if they are contained in a line. Although the linear independence implies the affine independence, the inverse does not hold. For example the points $(2,1),(4,2) \in \mathbb{R}^2$ are affinely independent, but not linearly independent

since $(4,2) = 2(2,1)$.
Linear Programming is concerned with maximizing or minimizing a linear objective function with a finite number of variables and a finite number of linear inequalities. Thus the set of feasible solutions is the intersection of a finite number of half-spaces and it is called *polyhedron*.

Definition 1.2.4. *A polyhedron $P \subseteq \mathbb{R}^n$ is the set of points that satisfy a finite set of linear inequalities, then $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.*

A bounded polyhedron is called *polytope*.

Definition 1.2.5. *The dimension of a polyhedron P, denoted by $dim(P)$, is the maximum number of affinely independent points in P minus* 1.

Definition 1.2.6. *A polyhedron $P \subseteq \mathbb{R}^n$ is full-dimensional if $dim(P) = n$.*

Let $M = \{1, ..., m\}$ be the set of indices of the rows of the matrix (A, b). We consider a partition of M in the following subsets: $M^= = \{i \in M : a^i x = b_i, \text{ for all } x \in P\}$, which contains the indices corresponding to the inequalities satisfied by the equality from any point in P, and $M^\leq = M \setminus M^= = \{i \in M : a^i x < b_i, \text{ for some } x \in P\}$. Let $(A^=, b^=)$ and $(A^\leq, b^\leq)$ be corresponding rows of (A, b). There is a relation between the dimension of the polyhedron P and the rank of the matrix $(A^=, b^=)$. In what follows, we assume that $P \neq \emptyset$ and $dim(\emptyset) = -1$. The following result holds:

Proposition 1.2.1. *Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a nonempty polyhedron. Then*

$$aff(P) = \{x \in \mathbb{R}^n : A^= x = b^=\} = \{x \in \mathbb{R}^n : A^= x \leq b^=\}.$$

Furthermore $dim(P) = n - rank(A^=, b^=)$.

Given the description of a polyhedron by a set of linear inequalities, our purpose is to find a *minimal description* for it, determining which inequalities are necessary and which can be eliminated.

Definition 1.2.7. *An inequality $\pi x \leq \pi_0$ is valid for P if it is satisfied by all points in P.*

Definition 1.2.8. *Given $\pi x \leq \pi_0$, a valid inequality for P, the set $F = \{x \in P : \pi x = \pi_0\}$ is called a face of P. Moreover, if $F \neq \emptyset$ and $F \neq P$ then F is a proper face induced by $\pi x \leq \pi_0$.*

A face F represented by $\pi x \leq \pi_0$ is non-empty if and only if $max\{\pi x : x \in P\} = \pi_0$, and in such a case we say that F *supports* P. Obviously, all the inequalities inducing faces which are not supports of P can be eliminated by its description. By definition, all faces are polyhedra. The empty face is trivial, and the entire polyhedron P is the face which corresponds to the trivial valid inequality $0 \leq 0$. Non-trivial faces are those having dimension between 0 and $dim(P) - 1$. Faces of dimension 0 are called *extreme points* and faces F such that $dim(F) = dim(P) - 1$ are called *facets*. Facets are the maximal proper faces of a polyhedron and they are necessary and sufficient for the description of P, as stated by the following proposition:

Proposition 1.2.2.

1. *For each facet F of P, one of the inequalities representing F is necessary in the description of P.*

2. *Every inequality $a^i x \leq b_i$ with $i \in M^{\leq}$ representing a face of P of dimension less than $dim(P) - 1$ is redundant in the description of P.*

The following theorem states that in a *minimal description* of P we must have a set of linearly independent equalities together with precisely one inequality for each facet of P.

Theorem 1.2.3.

1. *A full-dimensional polyhedron P has a unique (to within scalar multiplication) minimal representation by a finite set of linear inequalities. In more detail, for each facet F_i of P, with $i = 1, ..., t$, there is an inequality $a^i x \leq b_i$ (unique to within scalar multiplication) representing F_i and $P = \{x \in \mathbb{R}^n : a^i x \leq b_i$, for $i = 1, ..., t\}$.*

2. *If $dim(P) = n - k$, with $k > 0$, then $P = \{x \in \mathbb{R}^n : a^i x = b_i$, for $i = 1, ..., k, a^i x \leq b_i$, for $i = k+1, ..., k+t\}$. For $i = 1, ..., k$, $a^i x = b_i$ are a maximal set of linearly independent rows of $(A^=, b^=)$, and for $i = k+1, ..., k+t$, $a^i x \leq b_i$ is any inequality representing the facet F_i.*

Example 1.1. Let us consider the polyhedron P described by the following inequalities:

$$\begin{aligned} x_1 - x_2 &\leq 0 \\ -x_1 + x_2 &\leq 1 \\ 2x_2 &\geq 5 \\ 8x_1 - x_2 &\leq 16 \\ x_1 + x_2 &\geq 4 \\ x &\in \mathbb{R}^2 \end{aligned}$$

We want to derive the dimension of P and a minimal description for it. P is full-dimensional, then $dim(P) = 2$. The face $F_1 = \{x \in \mathbb{R}^2 : x_1 - x_2 = 0\} = \emptyset$, thus it is not a support for P and can be discarded. The face $F_2 = \{x \in \mathbb{R}^2 : -x_1 + x_2 = 1\}$ is a facet, indeed

$$(A^{=}_{F_2}, b^{=}_{F_2}) = \begin{pmatrix} -1 & 1 & 1 \end{pmatrix}$$

Thus, $(A^{=}_{F_2}, b^{=}_{F_2})$ has rank 1, then $dim(F_2) = 1 = dim(P) - 1$. Similarly inequalities $2x_2 \geq 5$ e $8x_1 - x_2 \leq 16$ are facet-defining. Finally, let us consider the face $F_5 = \{x \in \mathbb{R}^2 : x_1 + x_2 = 4\}$. It results that

$$(A^{=}_{F_5}, b^{=}_{F_5}) = \begin{pmatrix} 1 & 1 & 4 \\ 0 & 2 & 5 \\ -1 & 1 & 1 \end{pmatrix}$$

This matrix has rank 2, then $dim(F_5) = 0$. In more detail $F_5 = \{(\frac{3}{2}, \frac{5}{2})\}$ and it is not a facet for P. The minimal description for P is the following:

$$\begin{aligned} -x_1 + x_2 &\leq 1 \\ 2x_2 &\geq 5 \\ 8x_1 - x_2 &\leq 16 \end{aligned}$$

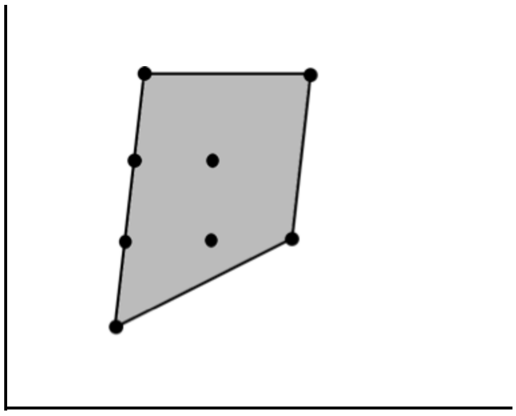

Figure 1.1: The black dots represent points in S while the grey area is the $conv(S)$.

Given the discrete optimization problem $min\{cx : x \in S\}$, where $S \subseteq \mathbb{Z}^n_+$, we formulate it as an integer program by specifying a polyhedron $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ such that $S = \mathbb{Z}^n_+ \cap P$. The aim is to find a linear inequality description of the set S. Let us introduce the following definition:

Definition 1.2.9. *Given $S \subseteq \mathbb{R}^n$, the convex hull of S, denoted by $conv(S)$, is the set of all points that are convex combination of points in S.*

The convex hull of a set S is the smallest convex set containing S. Figure 1.1 shows the convex hull of a set of integral points in $\mathbb{R}^2$. The following theorem holds:

Theorem 1.2.4. *If $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$,where (A,b) is an integer $m \times (n+1)$ matrix, and $S = P \cap \mathbb{Z}^n_+$, then $conv(S)$ is a polyhedron.*

Let us consider the following integer program, named (IP):

$$min\{cx : x \in S\}, S = P \cap \mathbb{Z}^n,$$

and the following linear program, named (LP):

$$min\{cx : x \in conv(S)\}.$$

The following theorem states that $min\{cx : x \in S\} = min\{cx : x \in conv(S)\}$:

Theorem 1.2.5. *Given $S = P \cap \mathbb{Z}^n \neq \emptyset$, $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ and $c \in \mathbb{R}^n$, it results that:*

1. *the objective value of (IP) is unbounded from below if and only if the objective value of (LP) is is unbounded from below;*

2. *If (LP) has a bounded optimal value, then it has an optimal solution that is an optimal solution to (IP);*

3. *If x_0 is an optimal solution to (IP), then x_0 is an optimal solution to (LP).*

Therefore, we can solve the integer program (IP) by solving the linear program (LP). The main issue concerns the representation of $conv(S)$ by a set of linear inequalities.
An inequality $\pi x \leq \pi_0$ is valid for the set S if $\pi x \leq \pi_0$, for any $x \in S$.

Proposition 1.2.6. *If $\pi x \leq \pi_0$ is valid for S, it is also valid for $conv(S)$.*

Two valid inequalities $\pi x \leq \pi_0$ and $\gamma x \leq \gamma_0$ are *equivalent* if $(\gamma, \gamma_0) = \mu(\pi, \pi_0)$, with $\mu > 0$. If there exists $\mu > 0$ such that $\gamma \geq \mu\pi$ and $\gamma_0 \leq \mu\pi_0$, then we say that $\pi x \leq \pi_0$ is *dominated* by $\gamma x \leq \gamma_0$. A *maximal* valid inequality is not dominated by any other valid inequality. It follows that any maximal valid inequality for S induces a non-empty face of $conv(S)$ and the set of maximal valid inequalities contains all of the facet-defining inequalities for $conv(S)$.
Further details on polyhedral theory can be found in [53], [59] and [43].

1.3 Branch and Cut Algorithm

The *Branch and Cut algorithm* is a very successful algorithm for solving a variety of integer programming problems. It was introduces in [45] and makes use of two techniques: the *Branch and Bound algorithm* and the *Cutting Plane method*.
The Branch and Bound algorithm is based on the idea of reducing the resolution of a difficult problem to that of simpler subproblems by performing a (recursive) partition of the feasible region. Computing upper and lower bounds of the optimal solution, only promising areas of the feasible region are explored, while are avoided part of it that cannot produce the optimal value. Therefore, *branching* is referred to the process of generating subproblems of the initial problem, while *bounding* refers to ignoring partial solutions that cannot be better than the current best solution. The Branch and

Bound algorithm is typically represented by a *tree*, where the root node is the initial integer linear programming and the other nodes are subproblems of it.
The Cutting Plane method iteratively refines a feasible set or objective function by means of linear inequalities. To solve an integer linear program, one considers the linear relaxation of the problem and repeatedly cuts out areas of the feasible region by adding new valid inequalities which are satisfied by the integer feasible solutions of the ILP but not by the optimal solution of its linear relaxation. Then, it is solved the linear relaxation of the problem obtained by adding the cuts and the procedure is iterated. The Cutting Plane methods often shows slow convergence, indeed by recursively adding cuts the resulting problem may become very large and there also may be a progressive loss of effectiveness of the cuts (tailing off).
The need to overcome the weaknesses of the Branch and Bound algorithm and the Cutting Plane method led to the development of the Branch and Cut algorithm. Given a node of the branch and bound tree, it is performed the search of new valid inequalities violated by the optimal solution of the current linear relaxation with the purpose of obtaining an optimal integer solution or a better bound, and if this is not possible a new branching is performed. To apply the Branch and Cut algorithm we need a structure to store the added cuts which globally valid and the definition of properly *separation procedures*, which allow to identify inequalities violated by the current relaxed solution.
The steps of the Branch and Cut algorithm for a minimization problem are summarized below:

1. Let L be the list of the active nodes, namely subproblems that still need to be solved. We initialize L with the initial integer linear program and we set the initial solution $x^* = NULL$ and the initial optimal value $z^* = \infty$.

2. While L is non empty, we select a subproblem P_i from L. If $L = \emptyset$, then x^* is the optimal solution and z^* is the optimal value.

3. Solve the linear relaxation of the subproblem P_i, named $RL(P_i)$.

4. If $RL(P_i)$ is infeasible node P_i can be pruned and go to Step 2.

5. If $RL(P_i)$ is feasible, let x_i and z_i be the optimal solution and the optimal value respectively.

(a) if $z_i \geq z^*$, then node P_i can be pruned and go to Step 2.

(b) if $z_i \leq z^*$ and x_i is integer, then we update x^* and z^*, namely $x^* = x_i$ and $z^* = z_i$. Node P_i can be pruned and go to Step 2.

(c) if $z_i \leq z^*$ and x_i is not integer, then we search for cutting planes that are violated by x_i. If one or more violated cuts are found, we add them to the linear programming relaxation of P_i and go to Step 3.

6. Branch to partition the subproblem P_i into two subproblems with restricted feasible regions. Add these subproblems to L and to to Step 2.

Chapter 2

Network Design Problems

In this chapter, we describe some network design problems, which are closely related to those addressed in this thesis. Many combinatorial optimization problems can be classified as network design problems: given an undirected graph $G = (V,E)$, a *Network Design Problem (NDP)* consists of identifying an optimal subgraph F of G, which satisfies some feasibility constraints. For instance, the Minimum Spanning Tree problem, which consists of identifying a minimum cost spanning tree of G, is one of the best known NDP, and it is a model for many real-world problems. In the literature, there are several variants of the Minimum Spanning Tree problem, where the addition of other constraints makes the problem difficult to solve. In designing telecommunication networks, the additional requirement regards the survivability, which refers to the restoration of services when a node or link failure occurs. Therefore, to achieve survivability it is necessary to introduce some connectivity constraints, which give rise to spanning subgraphs different from the tree.
In Generalized NDP, the set of vertices V is partitioned into *clusters*, and the feasibility conditions are expressed in terms of the clusters. The Generalized Minimum Spanning Tree problem is one of the most studied Generalized NDP, and it consists of determining a minimum cost tree, spanning exactly/at least one node for each cluster. In the remainder of the chapter, we describe in more detail these problems. In Section 2.1 we introduce the definition and some properties of the Minimum Spanning Tree problem, along with some of its NP-hard variants. Section 2.2 concerns the generalized version of NDPs. Finally, in Section 2.3 we present an overview of the most common survivability requirements for network design problems.

2.1 The Spanning Tree Problem

Let $G = (V, E)$ be an undirected connected graph, where V is the set of the vertices and E is the set of the edges. Given a subgraph $G_T = (V, E_T)$ of G, G_T is a tree if it satisfies at least one of the following:

(i) any two vertices in G_T can be connected by a unique path;

(ii) G_T is acyclic and $E_T = |V| - 1$;

(iii) G_T is connected and $E_T = |V| - 1$.

Let us introduce a cost function on G, $c : E \longrightarrow R^+$, which assigns a cost c_e to each edge $e \in E$. The *Minimum Spanning Tree (MST) problem* consists of finding a spanning tree $G_T = (V, E_T)$, with the minimum total cost, where the cost of a tree G_T is defined as the sum of the costs of the edges in E_T, and is denoted by $C(G_T)$. The MST problem has several applications in the field of network design, including computer, telecommunication networks, transportation, electrical grids, and so on. It can be formulated as an integer linear program (ILP), as follows. For each edge $e \in E$, let x_e be a binary variable equal to 1 if e is selected, and 0 otherwise. Given $S \subseteq V$, we denote by $E(S)$ the subset of E, containing any edge having both extremes in S. Using the Subtour Elimination Constraints (SECs), an ILP formulation for the MST problem is the following:

$$\min \sum_{e \in E} c_e x_e \tag{2.1}$$

subject to

$$\sum_{e \in E} x_e = |V| - 1, \tag{2.2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \qquad S \subset V, |S| \geq 3 \tag{2.3}$$

$$x_e \in \{0, 1\}. \qquad e \in E \tag{2.4}$$

The objective function (2.1) minimizes the total cost of the selected edges. Constraint (2.2) ensures that exactly the $|V| - 1$ edges are selected, while constraints (2.3) are the

SECs, and they guarantee that the optimal solution does not contain cycles. Let us note that this ILP formulation is based on property (ii). As an alternative, we could use property (iii), and obtain the following ILP formulation:

$$\min \sum_{e \in E} c_e x_e \tag{2.5}$$

subject to

$$\sum_{e \in E} x_e = |V| - 1, \tag{2.6}$$

$$\sum_{\substack{e=\{u,v\} \in E, \\ u \in S, v \in V \setminus S}} x_e \geq 1, \qquad S \subset V, S \neq \emptyset \tag{2.7}$$

$$x_e \in \{0,1\}. \qquad e \in E \tag{2.8}$$

Constraints (2.7) ensure that any feasible solution is connected. Although both models contain an exponential number of constraints, the MST problem can be solved in polynomial time, by Kruskal algorithm ([37]), or Prim algorithm ([50]).
Integer programming problems are in general hard to solve, then we usually deal with some type of more easily solved relaxation of the problem. The most common relaxation is the *linear programming relaxation*, obtained by eliminating the restriction that the decision variables need to be integer. For instance, for the MST problem constraints (2.4) and (2.8) are replaced by

$$0 \leq x_e \leq 1 \qquad e \in E \tag{2.9}$$

In general the optimal solution of the linear relaxation for a minimization problem provides a lower bound for the optimal solution, but for the MST problem the optimal solution of the formulation (2.2)-(2.4) has the same value of the optimal solution of its linear relaxation. This property was stated by Edmonds in [11], through the following theorem:

Theorem 2.1.1. *The extreme points of the polyhedron defined by the linear programming relaxation of the spanning tree model (2.2)-(2.4) are the 0-1 incidence vectors of*

spanning trees.

2.1.1 Related Problems

In this subsection, we introduce some problems related to the MST problem. Although the latter can be solved in polynomial time, most of its constrained versions are NP-hard.

Given an undirected graph $G = (V,E)$, and a cost function $c : E \rightarrow \mathbb{R}^+$, let us consider a subset of vertices $T \subseteq V$, named set of *terminal* vertices, while $V \setminus T$ is the set of *steiner* vertices. The *Steiner Tree (ST) problem* consists of finding a minimum cost tree spanning the set of terminal vertices, and, if necessary, some of the steiner vertices. It is easy to see that, when $T = V$, ST problem corresponds to the MST problem. The ST problem was first studied for Euclidean distance metric [34], and unlike the MST, it is NP-hard [24]. ST problem has application in any situation where the task is to minimize the cost of connection among some important locations, like very large scale integration design, computer networks, and so on.

Given a connected undirected graph G, the *Maximum Leaf Spanning Tree (MLST) problem*, consists of finding a spanning tree in G, having the maximum number of leaves, where a leaf is a vertex with degree 1 ([22], [1]). This problem has application in the field of communication networks and circuit layouts. Although it is easy to solve on complete graph, for the general case it is NP-hard, [23].

Given an undirected graph G, the *Minimum Branch Vertices (MBV) problem*, consists of finding a spanning tree in G with the minimum number of branch vertices, where a vertex is *branch* if its degree is greater than two. This problem arises in the context of optical network design, where minimizing the number of branch vertices corresponds to minimizing the number of switches deployed in the network. Indeed, when an optical signal traverses a vertex with degree greater than two, it is splitted through the usage of an electronic device, a switch. Thus, to reduce the cost of the network, the number of switches must be minimized. The MBV problem was introduced by Gargano et al. [25], and it is NP-hard.

Given a connected undirected graph $G = (V,E)$, and T a spanning tree of G, a vertex

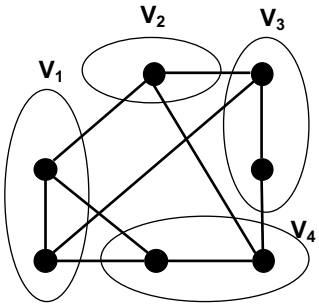

Figure 2.1: A graph $G = (V,E)$, where the set V is partitioned in clusters V_1, V_2, V_3, V_4.

$v \in V$, is said to be a *degree-preserving vertex*, if its degree in T is the same as its degree in G. The *Degree Preserving Spanning Tree (DPST) problem* [8] is to find a spanning tree T of G such that the number of degree-preserving vertices is maximized. This problem is NP-hard, and has application in water distribution networks.

The *Degree Constrained Minimum Spanning Tree (DCMST) problem* asks for a minimum cost spanning tree satisfying the condition that every vertex has degree no greater than a fixed value. By reducing it to an equivalent symmetric traveling salesman problem, Garey and Johnson [23] showed that the DCMST problem is NP-hard. This problem arises naturally in communication networks where the degree of a vertex represents the number of line interfaces available at a center.

2.2 Generalized Network Design Problems

Let $G = (V,E)$ be an undirected graph, with V the set of vertices, and E the set of edges. Moreover, a cost c_e is associated to each edge $e \in E$. In Generalized NDP (GNDP), the graph G is *clustered*, namely the set V is partitioned into k clusters, $V_1, ..., V_k$. Figure 2.1 shows an undirected graph $G = (V,E)$, where the set of vertices is partitioned into four clusters. It is easy to see that, if each cluster is a singleton, the GNDP reduces to the corresponding NDP. Formal definition for the GNDP have been introduced in [16]. Let us denote by $\mathcal{K}$ the set of indices of the clusters, namely $\mathcal{K} = \{1, ..., k\}$, and by *SP* a spanning NDP. Moreover, given $V' \subset V$, we denote by $G[V'] = (V', E(V'))$, the subgraph induced by V', and by $\mathcal{F}_{SP}(V')$, the set of the feasible solutions of the spanning problem *SP* on the graph $G[V']$. In the following, we give the definition of

the exactly generalization of the spanning problem *SP*.

Definition 2.2.1. *Given a clustered graph $G = (V,E)$, the "exactly" generalization of SP on G, consists of identifying a minimum cost subgraph $F = (V_F, E_F)$ of G, such that $|V_F \cap V_i| = 1$, for any $i \in \mathcal{K}$, and F belongs to $\mathcal{F}_{SP}\big(\cup_{i=1}^{k}(V_F \cap V_i)\big)$.*

This definition can be naturally extended to the "at least" and "at most" version of a GNDP, simply substituting the word "exactly" with "at least" or "at most". Obviously in the "exactly" and "at most" versions of a GNDP, intra-cluster edges are neglected.
In the remainder of this section, we introduce two Generalized Network Design Problems, which have been extensively studied in the literature: the Generalized Minimum Spanning Tree Problem and the Generalized Minimum Travelling Salesman Problem.

Given an undirected graph $G = (V,E)$, let us suppose that the set of vertices is partitioned into k clusters, $V_1, ..., V_k$, and on the edge of the graph is defined a cost function, $c : E \to \mathbb{R}^+$. The *Exactly Generalized Minimum Spanning Tree Problem (E-GMSTP)*, consists of determining a tree $G_T = (V_T, E_T)$, spanning exactly one vertex for each cluster, and with the minimum total cost. This problem arises in designing telecommunication networks, where it is required to interconnect metropolitan and regional areas by a tree, containing a gateway from each local area network. E-GMSTP was introduced by Myung et al. [42], and they proved that it is NP-hard. Feremans et al. [14] proposed several formulations for it, and made a comparison between them. Feremans et al. [15] derived some polyhedral results for the tightest formulation introduced in [14], and they developed a Branch and Cut algorithm. Pop et al. [48] designed a solution procedure, based on a new formulation for the problem. Some heuristic algorithms for the E-GMSTP can be found in [44], [18], [7], [49] and [26]. [47] provides a survey of the main known results for the E-GMSTP.
The *A least Generalized Minimum Spanning Tree Problem (L-GMSTP)*, consists of finding a tree $G_T = (V_T, E_T)$, spanning at least one vertex for each cluster, and with the minimum total cost. L-GMSTP has been proposed by Ihler et al. [35], and they proved that it is NP-hard. It has applications in agricultural settings, related to the construction of irrigation networks in desert environments. Dror et al. [10] proposed three mathematical formulations for the L-GMSTP. They also developed five heuristics, including a genetic algorithm. Feremans [13] designed an exact procedure and compared it to

the genetic algorithm in [10].

Given a graph $G = (V,E)$, and a partition of V into k clusters, the *Exactly Generalized Traveling Salesman Problem (E-GTSP)*, consists of finding a cycle passing once through exactly one node for each cluster, and having the minimum total cost. The symmetric E-GTSP has been introduced in [33], [57] and [52]. In the "at least" version of this problem, named L-GTSP, a feasible solution is a cycle visiting once at least a node for each cluster. These problems arises in location problems, logistics, postal routings, railway optimization, and so on. Fischetti et al. [19] studied the facial structure of both E-GTSP L-GTSP, and in [20] the same authors designed a Branch and Cut algorithm. Furthermore, several heuristic approaches have been proposed in the literature: an efficient composite heuristic [51], genetic algorithms [54][46], and a memetic approach [32].

2.3 Survivable Networks

Let us recall that a NDP consists of determining an optimal subgraph of a graph, taking into account some feasibility constraints. The latter are usually related to the real-world situations in which the problem arises. In the process of planning fiber-optic networks, the main issues are the *economy* and the *survivability*. Economy refers to the construction costs and is expressed as the sum of the costs of the edges of the network, while survivability refers to the restoration of services in case a node/edge fails. Thus, the aim is to determine a set of links connecting all the nodes with a minimum cost, ensuring some survivability criteria. Trees meet the primary goal of minimizing the total cost while connecting all the nodes. However, only one node/edge breakdown causes the failure of a tree network in its main purpose of enabling communication between all pairs of nodes. The construction of network topologies that protect against failures is one of the most important problems in the field of network design. A network is *survivable* if there exists a pre-specified number of node-disjoint or edge-disjoint paths between any two nodes/edges in the network. Given an undirected graph $G = (V,E)$, G is *k-edge-connected* (respectively *k-node-connected*) if for each pair of nodes $u, v \in V$, G contains at least k edge-disjoint (respectively, node-disjoint) paths between u and v. The *edge connectivity* (respectively, *node-connectivity*) of a graph G is the maximum

k for which G is k-edge-connected (respectively, k-node-connected). The notions of 1-edge-connectivity and 1-node-connectivity coincide and are reduced to the connectivity requirement. 2-connectivity concerns the protection of the network from a single link or node failure, and it has been found that 2-connected networks provide a sufficient level of survivability, indeed the probability of having multiple simultaneous failures is low. Therefore, a considerable amount of research has focused on the so-called *Low Connectivity Constrained Network Design problems*, where each node v is characterized by a requirement $r_v \in \{0,1,2\}$, and for all pairs of nodes u,v there must be at least $min\{r_u, r_v\}$ edge/node-disjoint paths.

Let us now introduce some definitions, which are useful in the context of survivable NDPs. A *connected component* of a graph G is a maximal connected subgraph of G. Given a graph G, and an edge $e \in E$, if $G \setminus \{e\}$ has more connected components than G, we call e a *bridge*. Similarly, given a node set $W \subseteq V$, if $G \setminus W$ has more connected components than G, we call W an *articulation set* of G. If a single node forms an articulation set, the node is called *articulation point*. Finally, given $W \subseteq V$, we define the subset of edges $\delta(W) = \{e = \{u,v\} \in E : u \in W, v \in V \setminus W\}$ as the *cut* induced by W. Menger's theorem [39] states the relation between edge-disjoint paths and cuts, and node-disjoint paths and articulation sets:

Theorem 2.3.1 (Menger,1927)**.**

1. *In a graph $G = (V,E)$, there is no cut of size $k-1$ or less disconnecting two given nodes s and t, if and only if there exist at least k edge-disjoint paths between s and t in G.*

2. *Let s and t be two non-adjacent nodes in G. Then, there is no articulation set W of size $k-1$ or less disconnecting s and t, if and only if there exist at least k node-disjoint paths between s and t in G.*

Grötschel, Monma and Stoer studied in detail NDPs with connectivity constraints, and in [31] there is a survey of their work. More recently, Kerivin and Mahjoub [36] and Fortz and Labbé [21] did a review of the polyhedral approaches for the survivable NDPs. In their first work on survivable NDP, Grötschel and Monma [29] introduced a general model mixing edge and node survivability constraints. They derived the dimension of the associated polytope, proved facet results for two classes of valid inequalities, and fully described the polytope of the 1-connected network problem.

2.3.1 Low-Connectivity Constrained Network Design Problems

As we mentioned in the previous section, much of the research has focused on Low-Connectivity Constrained NDPs. The latter have been introduced by Monma et al. [41] and Stoer [58]. The real-world problem from which Low-Connectivity Constrained NDPs arise can be stated as follows. Let us consider a set of telephone offices that have to be connected by a network. Moreover, offices are ranked in the following way: *special offices*, for which a "high" degree of survivability has to be ensured; *ordinary offices*, which simply require to be connected to the network; *optional offices*, that eventually could not be part of the network. It is assigned the set of possible connections between the offices, as well as the fiber cable management costs. The problem is to determine where to place fiber cables minimizing the construction cost and ensuring certain survivability constraints. This problem can be modeled by a Low-Connectivity Constrained NDP, where the set of offices and the connections between them, are represented by an undirected graph $G = (V,E)$. An integer $r_v \in \{0,1,2\}$ is associated to each vertex $v \in V$: $r_v = 2$, if v is a special office; $r_v = 1$, if v is an ordinary office; $r_v = 0$, if v is an optional office. Therefore, the aim is determining a minimum-cost subgraph, satisfying node/edge-connectivity requirements, namely there exist at least $min\{r_u, r_v\}$ node/edge-disjoint paths between every pair of vertices, $u, v \in V$.
Let us consider some particular Low-Connectivity Constrained NDPs:

- $r_v \in \{0,1\}$, for any $v \in V$: it reduces to the well-known Steiner Tree problem.
- $r_v = 1$, for any $v \in V$: the problem reduces to the well-known Minimum Spanning Tree problem.
- $r_s = 1 = r_t$, with $s,t \in V$, and $r_v = 0$, for any $v \in V \setminus \{s,t\}$: it reduces to the problem of determining the shortest path between s and t.
- $r_v = 2$, for any $v \in V$: it consists of determining a 2-connected network with the minimum total cost, and it is NP-hard even if the graph is complete and the function of the costs satisfies the triangle inequality. Indeed, solving this problem on a graph is equivalent to decide whether the graph is Hamiltonian or not ([12]).

Chapter 3

Generation of 3-Connected non-Hamiltonian Graphs

In this chapter, we introduce a procedure for the generation of a class of 3-connected non-Hamiltonian graphs. Given an undirected graph G, the problem of determining whether G is Hamiltonian is NP-complete. Nevertheless, sometimes there is an easily verifiable proof of the fact that G is non-Hamiltonian. In Section 3.1 we present some sufficient and necessary conditions for a graph to be Hamiltonian. A procedure for the generation of a family of 3-connected non-Hamiltonian graphs is proposed in Section 3.2.

3.1 Hamiltonian Graphs

Given an undirected graph $G = (V,E)$, an *Hamiltonian cycle* in G, is a cycle passing once through all the vertices in G. A graph G is said to be *Hamiltonian*, if there exists an Hamiltonian cycle in G. Decide whether a graph is Hamiltonian or not is a NP-complete problem. Trivially, if G is complete it is Hamiltonian. Given $n = |V|$, it is interesting to ask which is the minimum number of edges $f(n)$, so that G is Hamiltonian. For instance, a graph G with n vertices and $1 + \frac{(n-1)(n-2)}{2}$ edges is non-Hamiltonian. Indeed, G is obtained by adding to a complete graph with $n-1$ vertices, a vertex of degree one. The following theorem is due to Dirac [9]:

Theorem 3.1.1. *Given an undirected graph G with n vertices, if each vertex v has*

degree at least $n/2$, then G is Hamiltonian.

Therefore, one may ask when a sequence of positive integers $d_1, ..., d_n \leq n$ ensures that every graph G with vertices $v_1, ..., v_n$, such that the degree of v_i in G is d_i, for any $i = 1, ..., n$, is Hamiltonian. Such a sequence is called *forcibly Hamiltonian*, and Chvàtal [4] introduced the following theorem:

Theorem 3.1.2. *A sequence $d_1, ..., d_n$ is forcibly Hamiltonian if there is no k such that $0 < k < n/2$, at least k of the numbers d_i are at most k and at least $n-k$ of them are less than $n-k$.*

A later generalization of this theorem is due to Bondy and Chvàtal [2], and uses the notion of *closure* of a graph G, denoted by $[G]$. The closure of G is obtained by adding to G the edges connecting each pair of non-adjacent vertices u and v in G, such that $d(u) + d(v) \geq n$, where $d(v)$ is the degree of v in G. The Bondy-Chvàtal Theorem is the following:

Theorem 3.1.3. *A graph G is Hamiltonian if and only if its closure $[G]$ is Hamiltonian.*

Finally, Chvàtal and Erdös [6] stated a sufficient condition for a graph G to be Hamiltonian, using the *independence number* of G, $\alpha(G)$, and its *connectivity*, $\mathtt{k}(G)$. The independence number of G is the cardinality of the largest independent vertex set, that is a set of vertices in G, no two of which are adjacent. While the connectivity of G is defined as the largest k such that G is k-connected. We recall that a graph G is k-connected if there exist at least k disjoint paths between any pair of vertices.

Theorem 3.1.4. *If $\alpha(G) \leq \mathtt{k}(G)$ then G is Hamiltonian.*

Although establishing whether a given graph G is Hamiltonian is a NP-complete problem, there is a good characterization of certain non-Hamiltonian graphs. Given a graph G, it is *1-tough* if for any subset of vertices S in G, the graph $G \setminus S$, resulting from G by the removal of S, consists of at most $|S|$ connected components. The following theorem is due to Chvàtal [5]:

Theorem 3.1.5. *If G is not 1-tough, then G is non-Hamiltonian.*

For instance, consider the graph G shown in Figure 3.1. The subgraph $G \setminus \{v_1, v_9, v_{11}\}$ consists of four connected components, thus G is non-Hamiltonian. The inverse of

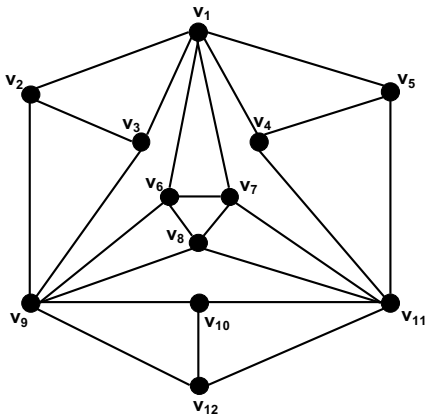

Figure 3.1: non-Hamiltonian

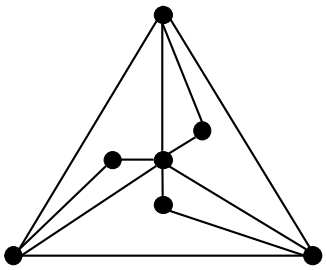

Figure 3.2: 1-tough and non-Hamiltonian

the theorem does not hold, indeed the subgraph in Figure 3.2 is 1-tough and non-Hamiltonian. Given a graph G, a *2-factor* is a subgraph of G in which all vertices have degree two, that is a collection of disjoint cycles which together cover all the vertices of G. It is easy to see that if G has no 2-factor, then G is non-Hamiltonian.

Let us mention an interesting class of non-Hamiltonian graphs, the *hypohamiltonian* graphs. Given a graph $G = (V,E)$, G is hypohamiltonian if G is non-Hamiltonian, but $G \setminus \{v\}$ is Hamiltonian for any $v \in V$. Hypohamiltonian graphs were first studied by Sousselier [56] and they arise in integer programming solutions to the Traveling Salesman problem, since certain kinds of hypohamiltonian graphs define facets of the traveling salesman polytope. Grötschel [28] observed that the computational complexity of determining whether a graph is hypohamiltonian, although unknown, is likely to be high. Every hypohamiltonian graph has the property of being 3-connected, since the removal of three vertices leaves a Hamiltonian path, which is connected. The smallest hypohamiltonian graph is the *Petersen graph*, which is shown in Figure 3.3.

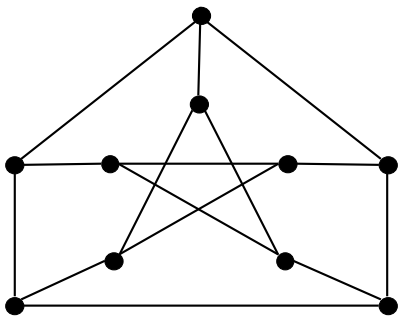

Figure 3.3: The Petersen graph.

3.2 A Class of 3-Connected non-Hamiltonian Graphs

In this section, we devise a procedure to generate a class of 3-connected non-Hamiltonian graphs. Let $G = (V,E)$ be an undirected graph, we denote by $\delta(v)$ the set of incident edges on v, by $d(v)$ the cardinality of $\delta(v)$, and by $N(v) = \{u \in V : \{u,v\} \in \delta(v)\}$. Let $G' = (V',E')$ be a complete graph with $|V'| \geq 4$, and let W be a subset of V', with $|W| = 3$. Given a set of vertices T, with $|T| \geq 3$, we build a new graph

$G(G';W;T) = (V,E)$, as follows. The set V is obtained by adding to V' the vertices in T, namely $V = V' \cup T$. The set of edges E is the following:

$$E = E' \cup \{\{u,v\} : u \in T, v \in W\}$$

It is easy to see that $G(G';W;T)$ is 3-connected, indeed whenever we remove two vertices it remains connected. We claim that $G(G';W;T)$ is also non-Hamiltonian.

Proposition 3.2.1. *The graph $G(G';W;T)$ is non-Hamiltonian.*

Proof. To prove the assert we use Theorem 3.1.5, mentioned in the previous section. Therefore, it is sufficient to show that $G(G';W;T)$ is not 1-tough to prove that it is non-Hamiltonian. Let us recall that a graph G is 1-tough if for any subset of vertices S in G, $G \setminus S$ consists of at most $|S|$ connected components. Choosing $S = W$, it results that $G(G';W;T) \setminus S$ is composed by the following connected components: $\{u_1\}$,...,$\{u_p\}$, $G' \setminus S$, where $T = \{u_1,...,u_p\}$. Since $p \geq 3$, $G(G';W;T) \setminus S$ consists of at least four connected components, while $|S| = 3$. This implies that $G(G';W;T)$ is not 1-tough, and then non-Hamiltonian. □

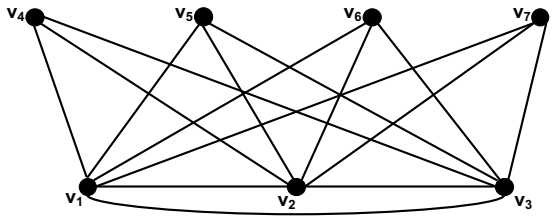

Figure 3.4: A graph $G(G';W;T)$ with $V' = \{v_1,v_2,v_3,v_4\}$, $W = \{v_1,v_2,v_3\}$ and $T = \{v_5,v_6,v_7\}$.

Let us note that the graph shown in Figure 3.4 is the graph $G(G';W;T)$, where $G' = (V',E')$ is the complete graph with $V' = \{v_1,v_2,v_3,v_4\}$, $W = \{v_1,v_2,v_3\}$ and $T = \{v_5,v_6,v_7\}$.
This construction can be further generalized as described below. Let $G' = (V',E')$ be a complete graph such that $|V'| \geq 3q+1$, where $q \geq 1$. Let us consider q disjoint subsets of vertices $W_1,...,W_q \subseteq V'$, such that $|W_i| = 3$, for any $i = 1,...,q$. Given q disjoint sets of vertices $T_1,...,T_q$, with $|T_i| \geq 3$, for any $i = 1,...,q$, we build a new

graph $G(G', W_1, ..., W_q, T_1, ..., T_q) = (V, E)$, where $V = V' \cup T_1 \cup ... \cup T_q$ and the set of edges E is the following:

$$E = E' \cup \{\{u, v\} : u \in T_i, v \in W_i\}_{i=1,...,q}$$

The graph $G(G', W_1, ..., W_q, T_1, ..., T_q)$ is 3-connected, and also non-Hamiltonian.

Proposition 3.2.2. *The graph $G(G', W_1, ..., W_q, T_1, ..., T_q)$ is non-Hamiltonian.*

Proof. The proof is analogous to that of the Theorem 3.2.1, since we show that the graph $G(G', W_1, ..., W_q, T_1, ..., T_q)$ is not 1-tough. Indeed, choosing $S = W_i$, with $i = 1, ..., q$, the graph $G(G', W_1, ..., W_q, T_1, ..., T_q) \setminus S$ is made of $|T_i| + 1$ connected components, where $|T_i| \geq 3$, while $|S| = 3$. □

Therefore, graphs $G(G', W_1, ..., W_q, T_1, ..., T_q)$ constitute a family of 3-connected non-Hamiltonian graphs. A procedure for the generation of such graphs is described in Algorithm 1. The inputs of the procedure are three integers: q, $n' \geq 3q + 1$, and $\bar{n} \geq 3q$. The function $BuildComplete(n')$ builds a complete graph with n' vertices (line 1). Then, the function $3DisjointSubset(V')$ returns q disjoint subsets of vertices $W_1, ..., W_q \subseteq V'$, with $|W_i| = 3$. For any $i = 1, ..., q - 1$ we compute t_i as a random integer between 3 and $\bar{n}/q$, and by the function $Vertices(t_i)$ we create a set T_i with t_i vertices (lines 3-5). Finally, we set t_q to $\bar{n} - \sum_{i=1}^{q} t_i$, and the set of vertices T_q is obtained by the function $Vertices(t_q)$. The algorithm returns the graph $G(G', W_1, ..., W_q, T_1, ..., T_q) = (V, E)$, where V is equal to $V \cup T_1 \cup ... \cup T_q$, while the set of edges E is initialized to E'. Then, for any $i = 1, ..., q$, we add to E the edges connecting each vertex in T_i to any vertex in W_i (lines 10-13).

Algorithm 1: Generation of $G(G', W_1, ..., W_q, T_1, ..., T_q)$

Input: q, $n' \geq 3q+1$, $\bar{n} \geq 3q$
Output: $G(G', W_1, ..., W_q, T_1, ..., T_q)$

$G' = (V', E') \leftarrow BuildComplete(n')$;
$(W_1, ..., Wq) \leftarrow 3DisjointSubset(V')$;
for $i = 1$ *to* $q-1$ **do**
    $t_i \leftarrow random(3, \bar{n}/q)$;
    $T_i \leftarrow Vertices(t_i)$;
$t_q \leftarrow \bar{n} - \sum_{i=1}^{q-1} t_i$;
$T_q \leftarrow Vertices(t_q)$;
$V \leftarrow V' \cup T_1 \cup ... \cup T_q$;
$E \leftarrow E'$;
for $i = 1$ *to* q **do**
    for $u \in T_i$ **do**
        for $v \in W_i$ **do**
            $E \leftarrow E \cup \{u, v\}$;
$G(G', W_1, ..., W_q, T_1, ..., T_q) \leftarrow (V, E)$;

Chapter 4

The Generalized Minimum Branch Vertices Problem

4.1 Introduction

In this chapter, we introduce the *Generalized Minimum Branch Vertices (GMBV) problem*: given an undirected graph $G = (V, E)$, where V is partitioned into clusters $V_1, ..., V_k$, the aim is determining a subgraph spanning exactly one vertex for each cluster, and with the minimum number of branch vertices, namely vertices with degree greater than two. The GMBV problem is NP-hard, indeed when each cluster is a singleton it reduces to the Minimum Branch Vertices (MBV) problem. The MBV problem was introduced by Gargano et al. [25]. Carrabs et al. [3] proposed four mathematical formulations, while Silvestri et al. [55] derived some valid inequalities and proposed a hybrid formulation with both undirected and directed variables, which was solved through a Branch & Cut algorithm. Landete et al. [38] investigated decomposition methods for degree dependent spanning tree problems. Finally, Merabet et al. [40] proposed a generalization of the MBV problem, introducing the definition of k-branch vertex, a vertex with degree greater than $k+2$.
The GMBV problem arises in the context of optical network. When designing Metropolitan Area Network (MAN) we need to interconnect several Local Area Network (LAN), by selecting a hub for each LAN, and then connecting hubs through transmission links. If more than two links entering a hub are chosen, the optical signal has to be split us-

ing a dedicated network device, a switch. Then, the minimization of the number of switches in the network is required to minimize the costs. This problem can be modelled as a GMBV problem, where each LAN is a cluster, each hub is a vertex, and hubs where a switch is deployed are branch vertices. To the best of our knowledge, the GMBV problem has never been introduced before. However, in the literature, the generalized version of other Network Design Problems have been extensively studied. Feremans et al. [17] provided a definition of the Generalized Network Design Problem, as a problem defined over clustered graph and where the feasibility conditions are expressed in terms of the clusters. Myung et al. [42] introduced the Generalized Minimum Spanning Tree problem, and Feremans et al. [14] proposed several mathematical formulations for it. Moreover in [15], they developed new valid inequalities and designed a Branch & Cut algorithm. Fischetti et al. [19] conducted a polyhedral analysis for the Generalized Travelling Salesman Problem, and in [20] they proposed a Branch & Cut algorithm.
The remainder of the chapter is organized as follows. In Section 4.2 we introduce the definition of the problem and some notations. In Section 4.3, we propose an integer linear programming formulation for the GMBV problem. Section 4.4 reports some properties about clustered graphs. In Section 5.3, we derive the dimension of the polyhedron as well as some facet related results, and we introduce some valid inequalities. A Branch and Cut algorithm is described in Section 4.6, and the computational results are summarized in Section 4.7.

4.2 Definition of the Problem and Notation

Let $G = (V, E)$ be an undirected graph, where V is the set of vertices, and E is the set of edges. Moreover, G is *clustered*, which means that V is partitioned into k clusters, $V_1, ..., V_k$, see Figure 5.1(a). A *generalized spanning tree (gst)* of G is a subgraph $G_T = (V_T, E_T)$ of G, such that G_T is a tree and $|V_T \cap V_i| = 1$, for any $i = 1, ..., k$. A vertex $v \in V_T$ is a *branch vertex* if its degree is greater than two in G_T. The *Generalized Minimum Branch Vertices (GMBV) problem*, consists of finding a *gst* in G, with the minimum number of branch vertices. Since exactly one vertex for each cluster can be selected, edges among vertices of the same cluster are neglected. Let us consider the graph shown in Figure 5.1(a). An optimal solution to the GMBV problem in G, is the

tree in Figure 5.1(b), having only one branch vertex.

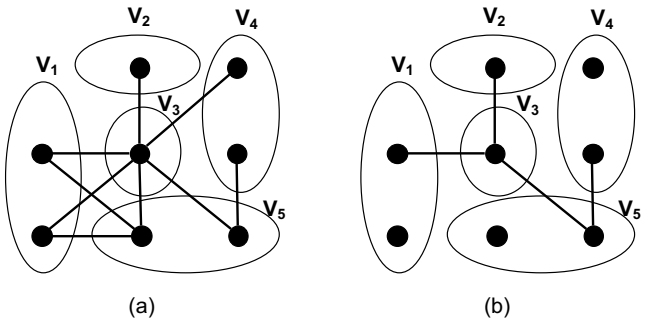

Figure 4.1: (a) A graph G with five clusters. (b) A *gst* of G with one branch vertex.

4.3 Mathematical Formulation

The GMBV problem can be formulated as an integer linear program (ILP) as follows. The binary variables are:

- x_e, $\forall e \in E$, is equal to 1 if e is selected, and 0 otherwise;
- y_v, $\forall v \in V$, is equal to 1 if v is selected, and 0 otherwise;
- z_v, $\forall v \in V$, is equal to 1 if v is a branch vertex, and 0 otherwise.

Given $E' \subseteq E$ and $V' \subseteq V$, we use the notations $x(E') = \sum_{e \in E'} x_e$, and $y(V') = \sum_{v \in V'} y_v$. For $S, T \subseteq V$, we define $E(S:T) = \{\{u,v\} \in E : u \in S, v \in T\}$, and $E(S) = E(S:S)$ the set of the edges having both extremes in S. We denote by $\delta(S) = E(S : V \setminus S)$ the set of edges incident to vertices in S, and by $N(S) = \{u \in V \setminus S : \exists \{u,v\} \in \delta(S), v \in S\}$. When $S = \{v\}$, $\delta(\{v\})$ and $N(\{v\})$ become $\delta(v)$ and $N(v)$ respectively, and we denote by $d(v)$ the cardinality of $\delta(v)$. Let us denote by $\mathcal{K}$ the set of indices of the clusters, $\mathcal{K} = \{1, ..., k\}$. Given $S \subseteq V$, we define $\mu(S) = |\{i \in \mathcal{K} : V_i \subseteq S\}|$, that is the number of clusters included in S.
The ILP formulation is the following:

$$\text{Minimize } z = \sum_{v \in V} z_v \tag{4.1}$$

subject to

$$x(E) = k - 1 \tag{4.2}$$

$$y(V_i) = 1 \qquad i \in \mathcal{K} \tag{4.3}$$

$$x(E(S)) \leq y(S) - 1 \qquad S \subset V : |S| \geq 2, \mu(S) > 0 \tag{4.4}$$

$$x(\delta(v)) - 2y_v \leq (d(v) - 2)z_v \qquad v \in V \tag{4.5}$$

$$x_e \in \{0,1\} \qquad e \in E \tag{4.6}$$

$$y_v \in \{0,1\} \qquad v \in V \tag{4.7}$$

$$z_v \in \{0,1\} \qquad v \in V \tag{4.8}$$

The objective function (4.1) minimizes the number of branch vertices. Constraint (4.2) requires that the number of selected edges coincides with the number of clusters minus one. Constraints (4.3) guarantee that exactly a vertex is selected for each cluster. Constraints (4.4) are the Generalized Subtour Elimination Constraints (GSECs), introduced in [15]. Constraints (4.5) ensure that a selected vertex v is branch if at least three edges in $\delta(v)$ are selected. The objective function forces variables z_v to zero when $x(\delta(v)) \leq 2$ holds. However, to ensure that variables z_v fully describe a branch vertex, the following constraints can be introduced for any $v \in V$:

$$2z_v \leq x(\delta(v)) - y_v \tag{4.9}$$

Constraints (4.9) ensure that z_v is equal to zero if in $\delta(v)$ are selected less than three edges. Given a vertex $v \in V$, by $h(v)$ we denote the index of the cluster containing v, and then by $V_{h(v)}$ the cluster containing v. Let us consider the set $W = \{v \in V : |V_{h(v)}| = 1\}$, containing vertices which belong to clusters that are singleton. It is worth noting two particular cases of the GSECs:

$$x(E(\{v\} : V_i)) \leq y_v \qquad i \in \mathcal{K}, v \in V \setminus (W \cup V_i) \tag{4.10}$$

$$x(\delta(v)) \geq y_v \qquad v \in V \tag{4.11}$$

Constraints (4.10) are obtained from constraints (4.4), when $S = V_i \cup \{v\}$, while (4.11) are obtained choosing $S = V \setminus \{v\}$. Let us note that constraints (4.10) ensure that y_v is equal to 1 if at least one edge incident to v is selected, while constraints (4.11) ensure

that y_v is equal to 0 if no edge in $\delta(v)$ is selected. Furthermore, constraints (4.11)

Remark 1. *Constraints (4.11) is dominated by constraints (4.9).*

4.4 Properties of the Clustered Graphs

In this section, we show some properties that any feasible solution to the GMBV problem satisfies. Some of these properties can be used to determine useless vertices or edges, since they do not belong to any feasible solution. These elements could be identified and removed to reduce the size of the graph. Moreover, they will be used in Section 5.3 to obtain some polyhedral results about the GMBV polyhedron.
Given a subset of vertices $V' \subseteq V$, we denote by $G[V'] = (V', E(V'))$, the subgraph induced by V'. When $V' = \{a_1, ..., a_l\}$, instead of $G[\{a_1, ..., a_l\}]$ we use simply $G[a_1, ..., a_l]$.

4.4.1 v-Connection

Definition 4.4.1. *Given a vertex $v \in V$, G is v-connected if there exist vertices $a_1 \in V_1, ..., a_k \in V_k$, with $a_{h(v)} = v$, such that $G[a_1, ..., a_k]$ is connected.*

Given the graph $G = (V, E)$, shown in Figure 4.2(a), it is easy to see that it is v_1-connected, since the subgraph $G[v_1, v_4, v_5, v_6]$ is connected (see Figure 4.2(b)). On the contrary, G is not v_2-connected, because neither $G[v_2, v_3, v_5, v_6]$ nor $G[v_2, v_4, v_5, v_6]$ are connected. If the subgraph $G[a_1, ..., a_k]$ is not connected, let us denote by $c(a_1, ..., a_k)$ the number of connected components of $G[a_1, ..., a_k]$. Then, we define

$$c_v = min\{c(a_1, ..., a_k) : a_1 \in V_1, ..., a_k \in V_k, a_{h(v)} = v, \\ G[a_1, ..., a_k] \text{ is not connected}\}.$$

For example, for the graph shown in Figure 4.2(a), it results that $c(v_2, v_3, v_5, v_6) = c(v_2, v_4, v_5, v_6) = 2$, then $c_{v_2} = 2$.

Remark 2. *If each cluster is a singleton, given a vertex $v \in V$, G is v-connected if and only if G is connected.*

Indeed, since each cluster is a singleton, we have that $V_1 = \{a_1\}, ..., V_k = \{a_k\}$, and $G[a_1, ..., a_k] = G$. Therefore, the v-connection is an extension of the connection property to clustered graphs.

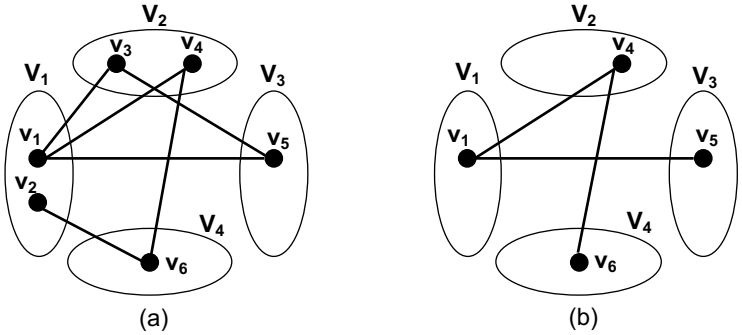

Figure 4.2: (a) A graph G, such that G is v_1-connected, but not v_2-connected. (b) A connected subgraph $G[v_1, v_4, v_5, v_6]$.

Lemma 4.4.1. *Given a vertex $v \in V$, it results that:*

1. *if G is v-connected, then there exists a feasible solution to the GMBV problem containing v.*
2. *if G is not v-connected, then v does not belong to any feasible solution to the GMBV problem.*

Proof.

1. If G is v-connected, there exist $a_1 \in V_1, ..., a_k \in V_k$, with $a_{h(v)} = v$, such that $G[a_1, ..., a_k]$ is connected. Therefore, there exists a *gst* G_T in $G[a_1, ..., a_k]$, that is a feasible solution to the GMBV problem containing v.
2. We prove it by contradiction. Let G_T be a feasible solution to the GMBV problem containing v. By definition G_T is a *gst*. Therefore, the subgraph induced by vertices in G_T is connected, and then G is v-connected, which is a contradiction.

□

Thanks to Lemma 4.4.1 it is possible to identify in G vertices that are useless because they do not belong to any feasible solution. To this end, for any $v \in V$, we have to

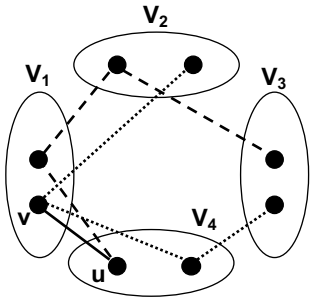

Figure 4.3: A v-connected (dotted line) and u-connected (dashed line) graph, such that the edge $\{u,v\}$ does not belong to any feasible solution.

check if G is v-connected, and if this condition does not hold v can be removed from G. In what follows we assume that G is v-connected, for any $v \in V$, to guarantee that any vertex may belong to a feasible solution to the GMBV problem. It is worth noting that in G there could be useless edges too. Indeed, given an edge $\{u,v\} \in E$, even if G is both u-connected and v-connected, it may not be contained in any feasible solution, as shown by the following remark:

Remark 3. *Given an edge $e = \{u,v\} \in E$, even if G is u-connected and v-connected, this does not imply that there exists a feasible solution to the GMBV problem in G, containing e.*

Let us consider the graph shown in Figure 4.3: it is u-connected and v-connected. It is easy to see that edge $\{u,v\}$ cannot belong to any feasible solution, since the contemporary selection of u and v does not allow to reach cluster V_3. Given a vertex $v \in V$, we denote by $G(v)$ the subgraph of G obtained by removing all vertices in $V_{h(v)}$, except for v, namely $G(v) = G[V \setminus \{V_{h(v)} \setminus \{v\}\}]$.

Lemma 4.4.2. *Given an edge $e = \{u,v\} \in E$, if $G(u)$ is v-connected, then there exists a feasible solution to the GMBV problem in G, containing e.*

Proof. Since $G(u)$ is v-connected, there exist $a_1 \in V_1, ..., a_k \in V_k$, with $a_{h(v)} = v$, such that $G[a_1, ..., a_k]$ is connected. Moreover, $V_{h(u)}$ in $G(u)$ is a singleton containing only vertex u, thus $a_{h(u)} = u$. Therefore, in $G[a_1, ..., a_k]$ there exists a *gst*, $G_T = (V_T, E_T)$, spanning u and v. If edge e belongs to E_T, the lemma is proved. Otherwise, we can build a new *gst*, $G'_T = (V_T, E'_T)$, where $E'_T = E_T \cup \{e\} \setminus \{e'\}$, with e' one of the edges in E_T belonging to the cycle generated by the introduction of e in G_T. □

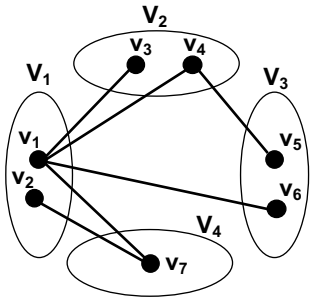

Figure 4.4: A graph G, such that vertex v_1 is a generalized cut vertex.

4.4.2 Generalized Cut Vertex

In this section, we extend the notion of *cut vertices* to clustered graphs, where a cut vertex is a vertex which removal disconnects the graph.

Definition 4.4.2. *A vertex $v \in V$ is a generalized cut vertex in G, if there exists a vertex $u \in N(v)$, such that $G[V \setminus V_{h(v)}]$ is not u-connected.*

In the graph depicted in Figure 4.4, v_1 is a generalized cut vertex, since $G[V \setminus V_{h(v_1)}]$ is not u-connected for any $u \in N(v_1)$, with $N(v_1) = \{v_3, v_4, v_6, v_7\}$. If v is a generalized cut vertex, let us denote by $c(v) = min\{c_u : u \in N(v), G[V \setminus V_{h(v)}]$ is not u-connected$\}$. In the previous example, we have that $c(v_1) = min\{c_{v_3}, c_{v_4}, c_{v_6}, c_{v_7}\}$, where $c_{v_3} = c_{v_6} = 3$, $c_{v_4} = min\{c(v_4, v_5, v_7), c(v_4, v_6, v_7)\} = 2$, and $c_{v_7} = 2$. Therefore, $c(v_1) = 2$.

Remark 4. *If each cluster is a singleton, a generalized cut vertex is exactly a cut vertex.*

Indeed, if v is a generalized cut vertex, there exists $u \in N(v)$ such that $G[V \setminus V_{h(v)}]$ is not u-connected. Without loss of generality, let us suppose that $V_{h(v)} = V_1$, and whenever we take $a_2 \in V_2, ..., a_k \in V_k$, with $a_{h(u)} = u$, $G[a_2, ..., a_k]$ is disconnected. Furthermore, it is straightforward to note that if each cluster contains exactly one vertex, then $G[a_2, ..., a_k] = G[V \setminus \{v\}]$, and it is disconnected. Therefore v is a cut vertex.

Lemma 4.4.3. *Given a vertex $v \in V$, if $G[V \setminus V_{h(v)}]$ is not u-connected, for any $u \in N(v)$, and $c(v) \geq 3$, then v is branch in any GMBV feasible solution which contains v.*

Proof. Let us suppose that $G[V \setminus V_{h(v)}]$ is not u-connected, for any $u \in N(v)$, and $c(v) \geq 3$. Whenever in cluster $V_{h(v)}$ we select v, to guarantee connectivity we have to select at

least three edges in $\delta(v)$. Therefore, v is branch in any feasible solution which contains v. □

Lemma 4.4.4. *If a vertex $v \in V$ is not a generalized cut vertex, then there exists a feasible solution to the GMBV problem containing v and exactly one edge incident on v.*

Proof. Since v is not a generalized cut vertex, for any $u \in N(v)$, $G[V \setminus V_{h(v)}]$ is u-connected. Without loss of generality let us assume $V_{h(v)} = V_1$. According to the definition, there exist $a_2 \in V_2, ..., a_k \in V_k$, with $a_{h(u)} = u$, such that $G[a_2, ..., a_k]$ is connected. Therefore, there is a spanning tree $G_{T_u} = (V_{T_u}, E_{T_u})$ in $G[a_2, ..., a_k]$, that is a *gst* in $G[V \setminus V_{h(v)}]$ containing u. Finally, if we consider $G_T = (V_T, E_T)$, where $V_T = V_{T_u} \cup \{v\}$ and $E_T = E_{T_u} \cup \{u, v\}$, it is a feasible solution to the GMBV problem in G, containing v and exactly one edge in $\delta(v)$. □

Notice that, if $v \in V$ is not a generalized cut vertex, we have $d(v)$ feasible solutions to the GMBV problem containing v and exactly one edge incident on it, one for each edge in $\delta(v)$.

Lemma 4.4.5. *If there are no generalized cut vertices in G, then for any $v \in V$ there exists a feasible solution to the GMBV problem containing v and at least two edges in $\delta(v)$.*

Proof. Since G does not contain any generalized cut vertex, $d(v) \geq 2$ for any $v \in V$. Given $v \in V$, let us consider $u \in N(v)$. Since u is not a generalized cut vertex, according to Lemma 4.4.4, there exists a feasible solution G_T to the GMBV problem containing u and exactly one edge in $\delta(u)$. In more detail, let us assume that $\{u, v\}$ belongs to G_T. Therefore, G_T contains $\{u, v\}$ and at least another edge incident on v, thus it is a feasible solution to the GMBV problem containing v, and with at least two edges belonging to $\delta(v)$. □

4.5 Polyhedral Analysis

Let us denote by $P(G)$ the polytope described by the constraints (4.2)-(4.8), that is:

$$P(G) = conv\{(x, y, z) \in \mathbb{R}^{|E|+2|V|} : (x, y, z) \text{ satisfies } (4.2) - (4.8)\} \tag{4.12}$$

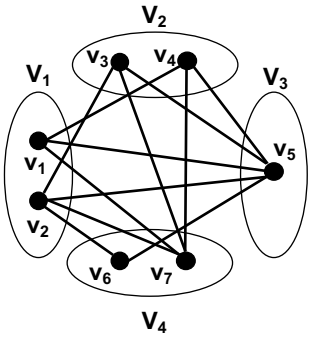

Figure 4.5: A graph $G = (V,E)$, with $k = 4$, $t = 6$ and $s = 3$, satisfying assumptions (A1) and (A2).

In this section, we examine some properties of the polytope $P(G)$. In order to assure that each vertex could be a branch vertex in a GMBV solution, we assume that $k \geq 4$, and $N(v)$ contains at least three vertices belonging to three different clusters, for any $v \in V$. We denote by $t = |V \setminus W|$, and by s the number of clusters that are not singleton, namely $s = |\{i \in \mathcal{K} : |V_i| > 1\}|$. We assume that:

(A1) G does not contain any generalized cut vertex;

(A2) if $t > 0$, there exist $S_1 \subset S_2 \subset ... \subset S_{t-s} \subset V$, with $\mu(S_i) = 0$ and $|S_i| = i$, for any $i = 1,...,t-s$, such that $G[V \setminus S_i]$ does not contain any generalized cut vertex.

Let us consider the graph $G = (V,E)$ shown in Figure 4.5. Here we have, $k = 4$, $t = 6$ and $s = 3$. G satisfies assumption (A1). Moreover, it satisfies assumption (A2), with $S_1 = \{v_1\}$, $S_2 = \{v_1, v_4\}$ and $S_3 = \{v_1, v_4, v_7\}$. Let us introduce some results, which were proposed by Fischetti et al. [19] for the Generalized Travelling Salesman problem, and here they are adapted to the GMBV problem.

Definition 4.5.1. *Let $\alpha x \leq \beta y + \gamma z + \delta$ be a valid inequality for $P(G)$. We denote by $\mathcal{H}(\alpha,\beta,\gamma,\delta)$ the face of $P(G)$ induced by $\alpha x \leq \beta y + \gamma z + \delta$. Given a vertex $v \in V \setminus W$, the v-restriction of $\alpha x \leq \beta y + \gamma z + \delta$ is the inequality obtained through the deletion of the variables y_v, z_v and x_e, for all $e \in \delta(v)$.*

Lemma 4.5.1. *Given a valid inequality $\alpha x \leq \beta y + \gamma z + \delta$, for any $v \in V \setminus W$, the dimension of $\mathcal{H}(\alpha,\beta,\gamma,\delta)$ is greater than or equal to the sum of the following quantities:*

(i) the dimension of the face of the polyhedron $P(G[V \setminus \{v\}])$ induced by the v-restriction of $\alpha x \leq \beta y + \gamma z + \delta$;

(ii) the rank of the matrix containing the coordinates of the extreme points of $\mathcal{H}(\alpha, \beta, \gamma, \delta)$ with $y_v = 1$, restricted to y_v, z_v and x_e, for any $e \in \delta(v)$.

Proof. Let us consider the matrix X, where each row is an extreme point of the face $\mathcal{H}(\alpha, \beta, \gamma, \delta)$. Let us note that, since the polyhedron does not contain the origin, the dimension of $\mathcal{H}(\alpha, \beta, \gamma, \delta)$ is the rank of X minus 1. Given $v \in V \setminus W$, matrix X can be partitioned in this way:

$$X = \begin{bmatrix} X_{11} & 0 & 0 & 0 \\ X_{21} & X_{22} & 1 & 0 \\ X_{31} & X_{32} & 1 & 1 \end{bmatrix}$$

The last two columns are associated to variables y_v and z_v, respectively, while the submatrix $\begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix}$ contains variables x_e, with $e \in \delta(v)$. It results that $rank\ X \geq rank\ X_{11} + rank \begin{bmatrix} X_{22} & 1 & 0 \\ X_{32} & 1 & 1 \end{bmatrix}$, where $rank\ X_{11}$ minus 1 is exactly the dimension of the face of $P(G[V \setminus \{v\}])$, induced by the v-restriction of $\alpha x \leq \beta y + \gamma z + \delta$. □

Given $V' \subseteq V$, we represent V' by its characteristic vector, $\nu^{V'} \in \mathbb{B}^n$, with $\nu_v^{V'} = 1$ if $v \in V'$, and $\nu_v^{V'} = 0$ otherwise. Analogously, given $E' \subseteq E$, let $\pi^{E'} \in \mathbb{B}^m$ be its characteristic vector, with $\pi_e^{E'} = 1$ if $e \in E'$, and $\pi_e^{E'} = 0$ otherwise. Moreover, we denote by $\mathbb{0}$ and $\mathbb{1}$, the vectors of all zeros and all ones, respectively.

Proposition 4.5.2. *$dim(P(G)) = m + 2n - k - 1$.*

Proof. Inequality $dim(P(G)) \leq m + 2n - k - 1$ is trivial, indeed constraints (4.2) and (4.3) are valid for $P(G)$, and they are linearly independent. We show the inverse inequality by induction on t. We recall that $t = |V \setminus W|$, namely t is the number of vertices which belong to clusters that are not singleton. When $t = 0$, the GMBV problem reduces to the MBV problem and the claim is true (see [55]). Let us assume that the claim holds for t, we prove it for $t + 1$. Since $t > 0$, then there exists at least a vertex $v \in V \setminus W$. According to Lemma 5.2.1, given the valid inequality $0 \leq 0$, $dim(P(G)) \geq$

$rank\ X_{11} + rank \begin{bmatrix} X_{22} & 1 & 0 \\ X_{32} & 1 & 1 \end{bmatrix}$. Furthermore, assumption (A2) implies that we can choose $v \in V \setminus W$ such that $G[V \setminus \{v\}]$ satisfies the same conditions as G, thus, from the induction hypothesis, it follows that $rank\ X_{11} = |E \setminus \delta(v)| + 2|V \setminus \{v\}| - k - 1$. Therefore, we need to show that $rank \begin{bmatrix} X_{22} & 1 & 0 \\ X_{32} & 1 & 1 \end{bmatrix} = d(v) + 2$. This matrix contains the columns associated to y_v, z_v and x_e, $e \in \delta(v)$, of the incidence vectors of feasible GMBV solutions containing vertex v. We have to exhibit $d(v) + 2$ incidence vectors belonging to $P(G)$, such that when restricted to y_v, z_v and x_e, $e \in \delta(v)$ are linearly independent. Due to assumption (A1) and Lemma 4.4.4, for any $u \in N(v)$ there exists a feasible solution $G_{T^u} = (V_{T^u}, E_{T^u})$ to the GMBV problem where $y_v = 1$, $x_{\{u,v\}} = 1$, and $x(\delta(v)) = 1$. Therefore, for any $u \in N(v)$, $(\pi^{E_{T^u}}, \nu^{V_{T^u}}, \mathbb{1} \setminus \nu^{\{v\}})$ belongs to $P(G)$, and matrix X_{22} is the identity matrix $I_{d(v)}$. Moreover, $(\pi^{E_{T^u}}, \nu^{V_{T^u}}, \mathbb{1})$, with $u \in N(v)$, belongs to $P(G)$ too. Due to Lemma 4.4.5, there exists a feasible solution $G_T = (V_T, E_T)$, with $y_v = 1$ and $x(\delta(v)) > 1$. Thus, $(\pi^{E_T}, \nu^{V_T}, \mathbb{1})$ belongs to $P(G)$. Introducing in the matrix the incidence vectors of these solutions, we have:

$$\begin{bmatrix} X_{22} & 1 & 0 \\ X_{32} & 1 & 1 \end{bmatrix} = \begin{bmatrix} I_{d(v)} & 1 & 0 \\ 1\,0\,0\,0\,...\,0 & 1 & 1 \\ 1\,...1\,0\,...\,0 & 1 & 1 \end{bmatrix}$$

The rows of this matrix are linearly independent, thus its rank is $d(v) + 2$. □

Lemma 4.5.3. *Given a valid inequality $\alpha x \leq \beta y + \gamma z + \delta$ inducing a proper face of $P(G)$, if there exists a vertex $v \in V \setminus W$ such that:*

(i) the v-restriction of this inequality is facet-defining for $P(G[V \setminus \{v\}])$;

(ii) there are $d(v) + 2$ incidence vectors of GMBV solutions containing v and belonging to $\mathcal{H}(\alpha, \beta, \gamma, \delta)$, such that their restriction to x_e, $e \in \delta(v)$, y_v and z_v, are linearly independent;

then, $\alpha x \leq \beta y + \gamma z + \delta$ is facet-defining for $P(G)$.

Proof. Inequality $\alpha x \leq \beta y + \gamma z + \delta$ induces a proper face of $P(G)$, then $dim(\mathcal{H}(\alpha, \beta, \gamma, \delta)) \leq m + 2n - k - 2$. Therefore, we have to prove the inverse inequality. By Lemma 5.2.1, $dim(\mathcal{H}(\alpha, \beta, \gamma, \delta)) \geq dim(v - restriction) + rank \begin{bmatrix} X_{22} & 1 & 0 \\ X_{32} & 1 & 1 \end{bmatrix}$.

Hypothesis 1 implies that the v-restriction of the inequality is a facet of $P(G[V \setminus \{v\}])$, then, for the Proposition 5.3.2, $dim(v-restriction) = |E \setminus \delta(v)| + 2|V \setminus \{v\}| - k - 2 = m + 2n - k - d(v) - 4$. Moreover, hypothesis 2 implies that $rank \begin{bmatrix} X_{22} & 1 & 0 \\ X_{32} & 1 & 1 \end{bmatrix} = d(v) + 2$. Therefore, $dim(\mathcal{H}(\alpha, \beta, \gamma, \delta)) \geq m + 2n - k - d(v) - 4 + d(v) + 2 = m + 2n - k - 2$. □

Proposition 4.5.4.

1. *Inequality $y_v \geq 0$, $v \in V$, is not facet-defining for $P(G)$.*
2. *Inequality $y_v \leq 1$, $v \in V$, is not facet-defining for $P(G)$.*

Proof.

1. Inequality $y_v \geq 0$ is not facet-defining for $P(G)$, since the face $P(G) \cap \{(x,y,z) : y_v = 0\}$ is properly contained in the proper face $P(G) \cap \{(x,y,z) : x(E(\{v\} : V_i)) = 0\}$, for any $i \neq h(v)$.
2. The inequality $y_v \leq 1$, $v \in V$, does not define a facet of $P(G)$, because of the constraints (4.3).

□

Proposition 4.5.5. *Inequality $z_v \leq 1$, $v \in V$, is a facet of $P(G)$, if assumption (A2) holds for $S_1, ..., S_{t-s} \subset V \setminus \{v\}$.*

Proof. We prove the proposition by induction on t. When $t = 0$, all clusters are singleton, so the GMBV problem reduces to the MBV problem, and the claim is true. Let us assume that the claim holds for t, we prove it for $t + 1$. Since $t > 0$, then there is a vertex $u \in V \setminus W$, with $u \neq v$, such that $G[V \setminus \{u\}]$ satisfies assumptions (A1) and (A2). By the induction hypothesis, the first request of Lemma 5.2.2 is satisfied. If we prove that the second hypothesis of Lemma 5.2.2 is satisfied too, we are done. To this end, we exhibit $d(u) + 2$ feasible solutions containing u, satisfying $z_v = 1$, and such that the restrictions of their incidence vectors to y_u, z_u and x_e, $e \in \delta(u)$, are linearly independent. By Lemma 4.4.4, for any $w \in N(u)$ there exists a feasible solution $G_{T^w} = (V_{T^w}, E_{T^w})$ to the GMBV problem with $y_u = 1$ and $x(\delta(u)) = 1$. Note that, $(\pi^{E_{T^w}}, \nu^{V_{T^w}}, \mathbb{1} \setminus \nu^{\{u\}})$ belongs to $P(G)$ and satisfies $z_v = 1$, for any $w \in N(u)$.

Furthermore, $(\pi^{E_{T^w}}, \nu^{V_{T^w}}, \mathbb{1}) \in P(G)$ too. By Lemma 4.4.5, there exists a feasible solution $G_T = (V_T, E_T)$ having $y_u = 1$ and $x(\delta(u)) > 1$, thus $(\pi^{E_T}, \nu^{V_T}, \mathbb{1})$ belongs to $P(G)$. These are $d(u)+2$ feasible solutions containing u, that satisfy $z_v = 1$, and their restriction to the variables associated to u gives rise to a full rank matrix. □

Proposition 4.5.6. *Inequality $z_v \geq 0, v \in V$, is a facet of $P(G)$ if*

1. *$G[V \setminus V_{h(v)}]$ does not contain any generalized cut vertex;*
2. *there exists $i \in \mathcal{K}$ such that $V_i \subseteq N(v)$;*
3. *assumption (A2) holds for $S_1, \ldots, S_{t-s} \subseteq V \setminus \{N(v) \cup \{v\}\}$.*

Proof. We prove the assert by induction on t. For $t = 0$, each cluster is a singleton, thus the GMBV problem reduces to the MBV problem and the claim is true. Let us assume that the claim is true for t, we prove it for $t+1$. Since $t > 0$, and according to hypothesis 2, it is always possible to choose a vertex $u \in V \setminus W$, with $u \neq v$ and $u \notin N(v)$, such that the graph $G[V \setminus \{u\}]$ satisfies (A1) and (A2). Therefore, by the induction hypothesis, the first request of Lemma 5.2.2 holds. Thus, to complete the proof, we have to exhibit $d(u)+2$ feasible solutions to the GMBV problem, satisfying $y_u = 1$ and $z_v = 0$, such that the restrictions of their incidence vectors to y_u, z_u and x_e, with $e \in \delta(u)$, are linearly independent. We have to consider two cases:

- $h(u) = h(v)$:
  whenever we consider a feasible solution with $y_u = 1$, we have $y_v = 0$, thus we can set z_v to 0. From assumption (A1), u is not a generalized cut vertex, and by Lemma 4.4.4, there exists a feasible solution $G_{T^w} = (V_{T^w}, E_{T^w})$, with $y_u = 1$ and $x(\delta(u)) = 1$, for any $w \in N(u)$. Therefore, $(\pi^{E_{T^w}}, \nu^{V_{T^w}}, \mathbb{1} \setminus \nu^{\{u,v\}})$ belongs to $P(G)$, for any $w \in N(u)$. Moreover, $(\pi^{E_{T^w}}, \nu^{V_{T^w}}, \mathbb{1} \setminus \nu^{\{v\}})$ belongs to $P(G)$ too. Finally, Lemma 4.4.5 assures that there exists a feasible solution $G_T = (V_T, E_T)$, with $y_u = 1$ and $x(\delta(u)) > 1$, thus $(\pi^{E_T}, \nu^{V_T}, \mathbb{1} \setminus \nu^{\{v\}})$ belongs to $P(G)$. Obviously, the restrictions of these solutions to y_u, z_u and x_e, with $e \in \delta(u)$, are $d(u)+2$ linearly independent vectors.

- $h(u) \neq h(v)$:
  u is not a generalized cut vertex in $G[V \setminus V_{h(v)}]$, then for any $w \in N(u)$, there exists

$G_{\bar{T}^w} = (V_{\bar{T}^w}, E_{\bar{T}^w})$, a feasible solution to the GMBV problem in $G[V \setminus V_{h(v)}]$. To obtain a feasible solution in G, it is sufficient to add an edge in $\delta(v)$, and this can always be done thanks to hypothesis 2. Thus, for any $w \in N(u)$, we have a feasible solution to the GMBV problem on G, $G_{T^w} = (V_{T^w}, E_{T^w})$, where $V_{T^w} = V_{\bar{T}^w} \cup \{v\}$ and $E_{T^w} = E_{\bar{T}^w} \cup \{e\}$, with $e \in \delta(v)$. This solution contains exactly one edge in $\delta(v)$, and its characteristic vector $(\pi^{E_{T^w}}, \nu^{V_{T^w}}, \mathbb{1} \setminus \nu^{\{u,v\}})$ belongs to $P(G)$ and satisfies $z_v = 0$. In such a way we have built $d(u)$ feasible solutions. The remaining two are the followings: one is that represented by $(\pi^{E_{T^w}}, \nu^{V_{T^w}}, \mathbb{1} \setminus \nu^{\{v\}})$, for a given $w \in N(u)$, and the other one is $(\pi^{E_T}, \nu^{V_T}, \mathbb{1} \setminus \nu^{\{v\}})$, where $G_T = (V_T, E_T)$ is obtained by adding to a feasible solution with $y_u = 1$ and $x(\delta(u)) > 1$ in $G[V \setminus V_{h(v)}]$, one edge in $\delta(v)$. These solutions are such that their restrictions to the variables related to u are linearly independent.

□

Proposition 4.5.7. *Given $v \in V$ and $H \subseteq \delta(v)$, with $|H| \geq 3$, inequality*

$$\sum_{e \in H} x_e - 2y_v \leq (|H| - 2)z_v \tag{4.13}$$

is valid for $P(G)$.

Proof. This inequality is derived from constraints (4.5). It ensures that whenever at least three edges in $H \subseteq \delta(v)$, with $|H| \geq 3$, are selected, then v is branch. □

4.6 Branch and Cut Algorithm

We designed a Branch and Cut algorithm for the GMBV problem, based on the ILP formulation introduced in Section 4.3. The steps of the algorithm are summarized in Algorithm 2. We initialize the linear program (LP) model by removing the exponential number of constraints GSECs, and relaxing the integrality constraints on the variables of the original formulation. Hence, the initial LP model is (4.2), (4.3), (4.5) and (4.9), and the continuous relaxation of (4.6)-(4.8). Given a subproblem L', we compute the optimal LP solution $(x^*_{LP}(L'))$: if it is feasible for the ILP, and it is better than the incumbent solution, then the incumbent is updated (line 14). Otherwise, if the LP

Algorithm 2: Branch and Cut algorithm for the GMBV problem

Input: integer linear program ILP
Output: optimal solution of ILP

```
1  L = ∅;
2  x' ← null;                    //incumbent
3  z(x') ← ∞;                    //value of the incumbent
4  L_0 ← first subproblem;
5  L ← L_0;
6  while L ≠ ∅ do
7      found ← true;
8      L' ← subproblem from L;
9      while found == true do
10         found ← false;
11         x*_LP(L') ← optimal LP solution of the subproblem L';
12         if z(x*_LP(L')) < z(x') then
13             if x*_LP(L') is feasible then
14                 x' ← x*_LP(L');          //update incumbent
15             else
16                 search for violated constraints (4.10);
17                 if violated constraints (4.10) are identified then
18                     add them to the model;
19                     found ← true;
20                 else if violated constraints (4.10) are not identified then
21                     search for violated constraints (4.4);
22                     if violated constraints (4.4) are identified then
23                         add them to the model;
24                         found ← true;
25                 search for violated constraints (4.13);
26                 if violated constraints (4.13) are identified then
27                     add them to the model;
28                     found ← true;
29                 else
30                     do the branching → subproblems L_1, L_2;
31                     L ← L_1, L_2;
```

solution is not feasible, we search for violated constraints (4.4) and (4.13) (lines 16-28). Since constraints (4.4) are exponentially many, we first check the violations of constraints (4.10) (lines 16-19); if no violated inequalities (4.10) are found, then we look for violations of constraints (4.4) (lines 20-24). We repeat this procedure until inequalities violated by the current relaxed solution are identified. When improvements are no longer possible, we branch on the variables using the default parameters of CPLEX (lines 29-31).

4.6.1 Preprocessing Phase

Before applying the Branch and Cut algorithm, a preprocessing phase is carried out to reduce the size of the instances, when possible, exploiting the properties introduced in Section 4.4. In that section, we introduced the definition of v-connection for a graph G and a vertex $v \in V$, and we showed that if G is not v-connected, then v does not belong to any feasible solution (see Lemma 4.4.1). Therefore, if G is not v-connected, v is useless and can be removed from G. In order to remove from G all the useless vertices, we verify if G is v-connected, for any $v \in V$. To this end, we build an auxiliary graph $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V}$ is obtained by adding to V k dummy vertices $t_1, ..., t_k$, such that each t_i is connected to all and only the vertices in V_i. Accordingly, $\bar{E} = E \cup \{\{u, t_i\} : \forall u \in V_i\}_{i \in \mathcal{K}}$. Furthermore, we introduce a cost function, $c : \bar{E} \to \mathbb{R}^+$, where $c(u,w) = 1$, for any $\{u,w\} \in \bar{E}$. The cost of a subgraph $G' = (V', E')$ of $\bar{G}$, is denoted by $C(G')$ and is the sum of the costs of the edges in E'. In Figure 4.6 is shown a graph $\bar{G}$ for a given graph G. Given a vertex $v \in V$, let us consider the Steiner Tree problem in $\bar{G}$ with terminal set $\{v, t_1, ..., t_k\}$, we call it $v - ST$ problem. The following lemma holds:

Lemma 4.6.1. *The value of any feasible solution to the $v - ST$ problem in $\bar{G}$ is greater than or equal to $2k - 1$.*

Proof. To prove the assert, let $\bar{G}'$ be a feasible solution to the $v - ST$ problem in $\bar{G}$. Let us note that $t_1, ..., t_k$ belong to the terminal set, then $\bar{G}'$ contains at least an edge incident on t_i, for any $i \in \mathcal{K}$. Moreover, $N(t_i) = V_i$, for any $i \in \mathcal{K}$, thus to reach t_i we need to select at least a vertex for each V_i. Thus, to ensure the connection between the terminal vertices, the clusters $V_1, ..., V_k$ must be connected by a tree, namely by $k - 1$ edges. Hence, $C(\bar{G}') \geq k + k - 1 = 2k - 1$. □

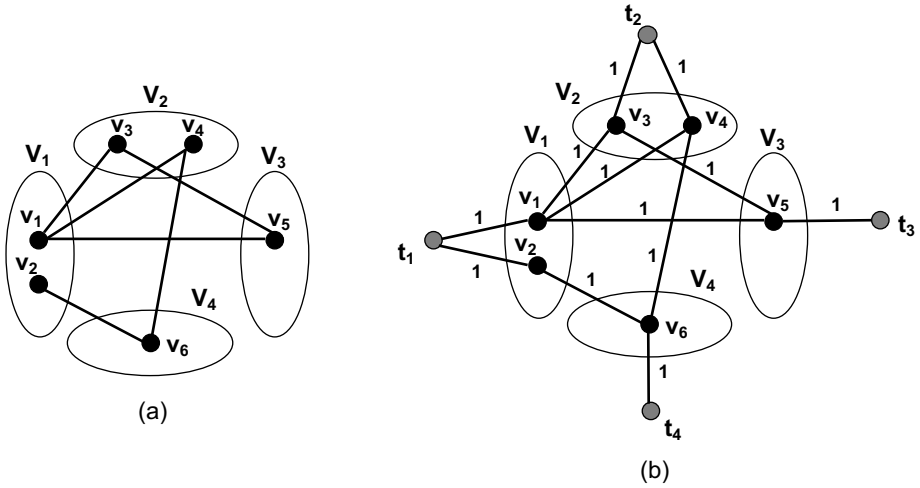

Figure 4.6: (a) A clustered graph $G = (V,E)$, with $k = 4$. (b) The auxiliary graph $\bar{G} = (\bar{V},\bar{E})$, associated to G.

Proposition 4.6.2. *G is v-connected if and only if the value of the optimal solution to the $v-ST$ problem in $\bar{G}$ is $2k-1$.*

Proof. Let us suppose that G is v-connected. According to Definition 4.4.1, there exist vertices $a_1 \in V_1, ..., a_k \in V_k$, with $a_{h(v)} = v$, such that $G[a_1, ..., a_k]$ is connected. Therefore, there exists a spanning tree G_T in $G[a_1, ..., a_k]$. A Steiner Tree $\bar{G}_T$ in $\bar{G}$ with terminal set $\{v, t_1, ..., t_k\}$, can be obtained by adding to G_T the edges $\{a_1, t_1\}, ..., \{a_k, t_k\}$. Clearly, $C(\bar{G}_T) = 2k-1$. Thanks to Lemma 4.6.1, $\bar{G}_T$ is an optimal solution to the $v-ST$ problem.
On the contrary, let us assume that the optimal solution to the $v-ST$ problem in $\bar{G}$ is a tree $\bar{G}_T = (\bar{V}_T, \bar{E}_T)$, with $C(\bar{G}_T) = 2k-1$. Vertices t_i belongs to the terminal set and $t_1, ..., t_k \in \bar{V}_T$. Furthermore, each vertex t_i can be reached only through vertices in V_i, then $|\bar{V}_T \cap V_i| \geq 1$, for any $i = 1, .., k$. Let us note that, since $C(\bar{G}_T) = 2k-1$, then $\bar{G}_T$ is a tree spanning $2k$ vertices, k of which are $t_1, ..., t_k$. This implies, that $\bar{G}_T[\bar{V}_T \setminus \{t_1, ...t_k\}]$ is a connected subgraph containing exactly one vertex for each cluster, $a_1 \in V_1, ..., a_k \in V_k$. Since v belongs to the terminal set, then $a_{h(v)} = v$, and G is v-connected. □

As a consequence of Proposition 4.6.2, to establish if G is v-connected, it is sufficient to optimally solve the $v-ST$ problem on the auxiliary graph $\bar{G}$. Nevertheless, since

Steiner Tree problem is NP-hard, this cannot be done in reasonable computational time. Therefore, we need another approach to determine in polynomial time if G is v-connected.

Let us consider a capacitated network, $\tilde{G} = (\tilde{V}, \tilde{A})$, built as follows. The set of vertices $\tilde{V}$ is obtained by adding to V, k sink vertices, $t_1, ..., t_k$, each of them requiring 1 unit of flow. Every edge $\{u, w\} \in E$ is replaced in $\tilde{G}$, by two arcs (u, w) and (w, u). Every vertex $u \in V_i$, with $i \in \mathcal{K}$, is connected to the corresponding sink vertex t_i, by arc (u, t_i). Finally, we introduce a capacity function on the arcs of $\tilde{G}$, $p : \tilde{A} \to \mathbb{R}^+$, where $p(u, w) = k$, for any $(u, w) \in \tilde{A}$ such that $u, v \in V$, while $p(u, t_i) = 1$, for any $u \in V_i$, $i \in \mathcal{K}$. For example, in Figure 4.7(b) is shown the auxiliary graph $\tilde{G}$, for the graph G depicted in Figure 4.7(a). Given $u \in \tilde{V}$, we denote by $\delta^-(u)$ the set of the ingoing arcs

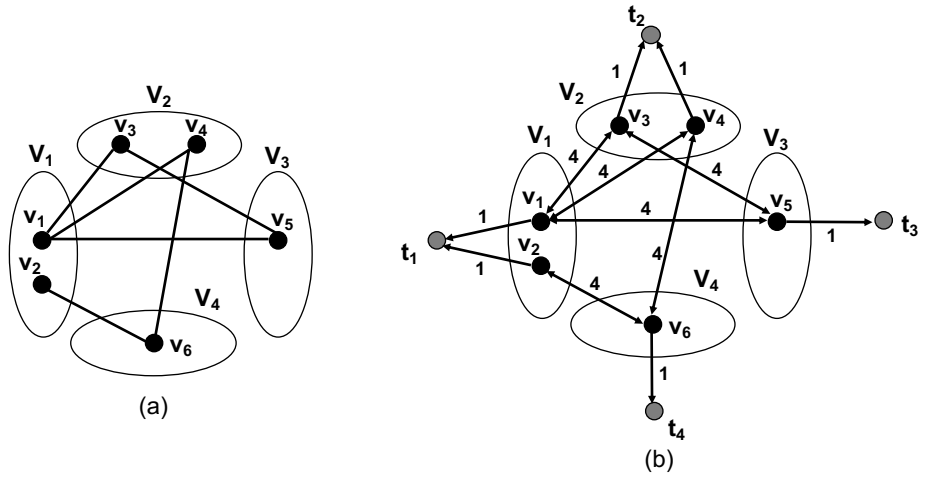

Figure 4.7: (a) A clustered graph $G = (V, E)$, with $k = 4$. (b) The auxiliary graph $\tilde{G} = (\tilde{V}, \tilde{A})$, associated to G.

in u, while $\delta^+(u)$ is the set of the outgoing arcs from u.

Given $v \in V$, we introduce on $\tilde{G}$ a decision problem, named $v - flow$, that consists of verifying if it is possible to satisfy the demands of the sink vertices $t_1, ..., t_k$, by sending k units of flow from vertex v, and using exactly one vertex for each cluster. For instance, the answer to the $v_1 - flow$ problem for the graph in Figure 4.7(b) is "yes", because it is possible to satisfy the demands of t_1, t_2, t_3 and t_4, by using the arcs in the subgraph induced by vertices v_1, v_4, v_5 and v_6. On the contrary, the answer to the $v_2 - flow$ problem on the same graph is "no", because it is not possible to satisfy the

demand of vertex t_3, having v_2 as the source of the flow and using exactly one vertex for each cluster.
The following proposition holds:

Proposition 4.6.3. *Given $v \in V$, G is v-connected if and only if the answer to the $v-flow$ problem on $\tilde{G}$ is "yes".*

Proof. If G is v-connected, according to Definition 4.4.1, there exist vertices $a_1 \in V_1, ..., a_k \in V_k$, with $a_{h(v)} = v$, such that $G[a_1, ..., a_k]$ is connected. Let us consider the subgraph of $\tilde{G}$, $G^* = (V^*, A^*)$, where $V^* = \{a_1, ..., a_k, t_1, ..., t_k\}$, while A^* is obtained by considering any directed arc in $\tilde{G}$ having both extremes in V^*. Since $G[a_1, ..., a_k]$ is connected in G, by the construction of $\tilde{G}$, in G^* there exists a path from v to any other node. For this reason, sending k unit of flow from vertex v, we are able to satisfy the demands of vertices $t_1, ..., t_k$, using the arcs in G^*. Thus, the answer to the $v-flow$ problem is "yes".
On the contrary, if the answer to the $v-flow$ problem is "yes", it is possible to satisfy the demands of $t_1, ..., t_k$, sending k unit of flow from v, and using exactly one vertex for each cluster, $a_1 \in V_1, ..., a_k \in V_k$, with $a_{h(v)} = v$. It is easy to see that the subgraph of G, $G[a_1, ..., a_k]$ is connected. Indeed, each sink node t_i is connected only to the vertices belonging to V_i and if $G[a_1, ..., a_k]$ is not connected, then the demand of some sink nodes is not satisfied, which is a contradiction. Therefore, G is v-connected. □

Let us consider the optimization problem, $Min(v-flow)$, corresponding to the the $v-flow$ problem, where the objective function is that of minimizing the number of vertices used to ship flow from v to $\{t_1, ..., t_k\}$. We can formulate it as a mixed-integer linear programming (MILP), as follows. Let $\tilde{x}_{uw}$, with $(u,w) \in \tilde{A}$, represents the amount of flow passing through arc (u,w). Moreover, let $\tilde{y}_u$, with $u \in V$, be a binary variable equal to 1 if vertex u is used to ship flow (in other words, if there exist at least an arc in $\delta^-(u)$ traversed by flow), and 0 otherwise. Given $A' \subseteq \tilde{A}$ and $V' \subseteq \tilde{V}$, we use the notations $\tilde{x}(A') = \sum_{(u,w)\in A'} \tilde{x}_{uw}$, and $\tilde{y}(V') = \sum_{u\in V'} \tilde{y}_u$. The MILP formulation is the following:

$$\text{Minimize } y = \sum_{u \in V} \tilde{y}_u \tag{4.14}$$

subject to

$$\tilde{y}_v = 1 \tag{4.15}$$

$$\tilde{y}(V_i) = 1, \qquad i \in \mathcal{K} \tag{4.16}$$

$$\tilde{x}(\delta^+(v)) - \tilde{x}(\delta^-(v)) = k, \tag{4.17}$$

$$\tilde{x}(\delta^+(u)) - \tilde{x}(\delta^-(u)) = 0, \qquad u \in V \setminus \{v\} \tag{4.18}$$

$$\tilde{x}(\delta^+(t_i)) - \tilde{x}(\delta^-(t_i)) = -1, \qquad i \in \mathcal{K} \tag{4.19}$$

$$k\tilde{y}_u \geq \tilde{x}(\delta^-(u)), \qquad u \in V \tag{4.20}$$

$$0 \leq \tilde{x}_{uw} \leq p(u,w), \qquad \{u,w\} \in \tilde{A} \tag{4.21}$$

$$\tilde{y}_u \in \{0,1\}. \qquad u \in V \tag{4.22}$$

The objective function (4.14) minimizes the number of vertices used to ship flow. Constraints (4.15) and (4.16) require that vertex v is used to ship flow and exactly one vertex for each cluster belongs to the network flow, respectively. Constraint (4.17) requires that v is the source of the flow. Constraints (4.18) state the conservation of flow. Constraints (4.19) require that vertices $t_1, ..., t_k$ are the sink nodes of the network flow. Constraints (4.20) establish the link between variables $\tilde{x}$ and $\tilde{y}$: variable $\tilde{y}_u$ is equal to 1 if at least an arc in $\delta^-(u)$ is traversed by flow. Finally, constraints (4.21) are the capacity constraints. It is easy to see that the following corollary of Proposition 4.6.3 holds:

Corollary 4.6.4. *The $Min(v-flow)$ problem is feasible if and only if G is v-connected.*

Let us note that the value of any feasible solution to the $Min(v-flow)$ problem is always equal to the number of clusters k.
The preprocessing procedure follows from Corollary 4.6.4: given the graph G, we build the directed graph $\tilde{G}$, as described before, and for any vertex $v \in V$, we check if the $Min(v-flow)$ problem on $\tilde{G}$ is feasible, and if not we remove v from G. It is worth noting that, thanks to the following remark, we do not need to solve the $Min(v-flow)$ problem for any $v \in V$.

Remark 5. *Given a vertex $v \in V$, if the $Min(v-flow)$ problem on $\tilde{G}$ is feasible, then for any vertex $u \in V$ belonging to this feasible solution, it results that the $Min(u-flow)$ problem is feasible too, and G is u-connected.*

Hence, the number of times the procedure is applied is less than the number of vertices of the graph. The $Min(v-flow)$ problem is solved by CPLEX, and since the value of any feasible solution is equal to the number of clusters, we set the lower cutoff tolerance parameter to k.

4.6.2 Separation Procedures

Since the number of constraints (4.13) is polynomial, no particular separation procedures are needed: we just check if $x(E(\{v\} : V_i)) \leq y_v$ is violated, for any $v \in V$ and $i \in \mathcal{K}$. The separation procedure for constraints (4.4) consists of solving a maximum flow problem on an auxiliary graph, built according to the current LP solution, as described in [15]. Finally, for constraints (4.13), the separation procedure is the one proposed by Lucena et al. [1]. In more detail, let $(\bar{x}, \bar{y}, \bar{z})$ be a feasible solution to the integer relaxation of the problem. Given a vertex $v \in V$ with $d(v) \geq 4$, we order variables $\bar{x}_e$, with $e \in \delta(v)$, in a decreasing way according to their values. For every $p = 3, ..., d(v) - 1$, we compute $\sum_{i=1}^{p} \bar{x}_{e_i} - (p-2)\bar{z}_v - 2\bar{y}_v$: if this value is greater than a certain tolerance, we have identified a subset $H \subseteq \delta(v)$, with $|H| = p$, for which constraints (4.13) is violated.

4.7 Computational Results

The Branch and Cut algorithm was coded in C++ on an OSX platform, running on an Intel Core i7 3.4 GHz processor with 8 GB of RAM. For the model the Concert library of IBM ILOG CPLEX 12.8 was used (default parameters and single thread mode). Furthermore, all CPLEX Cuts were disabled, because useless and wasteful.

4.7.1 Instances Generation

Since no benchmark instances for the GMBV problem are available in the literature, then we generated them to evaluate the performance of the Branch and Cut algorithm. It is worth noting that if the density of the graph is too high, in most cases the optimal solution to the GMBV problem is zero. Therefore, to guarantee a significant number of branch vertices in the optimal solution, we generated graphs

with a low density. The instances are grouped in three sets: Small instances, with $k \in \{12, 16, 20\}$, Medium instances, with $k \in \{30, 40, 50\}$, and Large instances, with $k \in \{60, 70, 80\}$. The number of vertices is chosen as a multiple of the number of clusters, $n \in \{3k, 4k, 6k, 8k, 10k\}$, and vertices are randomly distributed among the clusters. Finally, the number of edges has been generated according to the following formula, [3]: $m = \lfloor n - 1 + \delta \times 0.5 \times \lceil \sqrt{n} \rceil \rfloor$, where $\delta \in \{1, 3, 5\}$. For each combination of k, n and δ, we have a different scenario, and for each scenario we generated five instances, thus the total number of instances is 675. Therefore, each line in the tables represents a scenario composed by 5 instances with the same characteristics but different topologies, and the results reported in each line are the average values on these 5 instances. From the computational tests, it turns out that Small instances are easy to solve, because the optimal solution is always found within 20 seconds. For this reason, in the following we present computational results only for Medium and Large instances. The detailed computational results for Small instances can be found in Appendix A.

4.7.2 Preprocessing

To evaluate the effectiveness of the preprocessing procedure, we look over three parameters: the number of removed vertices, the computational time it requires, and the time reduction gained applying the Branch and Cut algorithm after preprocessing the instances. These parameters for Medium and Large instances are summarized in Table 4.1. In the first three columns of the tables, there are the informations about the instances: the number of clusters (k), the number of vertices (n), and the number of edges (m). In column $\%RV$ is reported the percentage of vertices removed by the preprocessing procedure, computed as $\%RV = \frac{RV \times 100}{n}$, where RV is the number of vertices removed by the preprocessing procedure. In column $\%TR$ there is the percentage of time reduction obtained comparing the computational time required by the Branch and Cut algorithm with the preprocessing phase (TP) and the by Branch and Cut algorithm without it (TNP), and it is computed as $\%TR = \frac{(TNP - TP) \times 100}{TNP}$. Finally, in column $time$ there is the computational time, in seconds, required by the preprocessing phase. The preprocessing procedure is quite fast, because it requires at most $29,6$ seconds on the Medium instances, and at most 87 seconds on the Large instances. The parameters that mostly affect the preprocessing phase, both in terms of effectiveness

k	n	m	%RV	%TR	time
30	90	93	20,7	-141,4	0,1
		103	10,9	-67,6	0,2
		112	7,3	-43,1	0,2
	120	124	18,7	-88,3	0,3
		135	16,0	-4,2	0,6
		146	0,8	28,5	0,5
	180	185	8,1	3,4	0,9
		199	2,7	20,9	1,4
		212	1,8	52,9	2,1
	240	246	12,5	27,1	2,2
		262	5,9	45,3	3,8
		277	2,8	57,1	4,6
	300	307	13,8	59,3	3,9
		324	4,5	63,3	5,8
		342	2,1	67,2	8,5
40	120	124	15,3	-59,4	0,3
		135	6,5	24,9	0,3
		146	3,5	12,5	0,4
	160	165	14,4	-14,8	0,5
		177	4,5	20,5	0,9
		190	1,8	37,9	1,1
	240	246	10,7	38,3	1,8
		262	3,4	74,2	3,1
		277	0,6	89,1	3,6
	320	327	13,7	16,5	4,8
		345	2,2	71,9	6,3
		363	1,5	76,6	9,4
	400	409	8,0	52,1	8,7
		429	4,0	80,5	11,8
		449	3,0	13,8	15,1
50	150	155	14,3	-43,8	0,4
		167	6,9	25,4	0,5
		179	3,6	52,5	0,7
	200	206	11,9	35,1	1,0
		220	5,9	24,8	1,3
		234	0,9	83,6	2,1
	300	307	7,7	38,9	3,8
		324	4,2	61,9	5,3
		342	1,1	71,6	6,6
	400	409	7,5	60,5	8,2
		429	3,4	83,1	11,4
		449	0,8	48,2	17,0
	500	510	7,3	71,4	16,2
		532	2,6	26,4	18,7
		554	1,8	-0,8	29,6

k	n	m	%RV	%TR	time
60	180	185	15,0	-35,2	0,5
		199	7,4	31,9	0,7
		212	2,7	65,5	1,0
	240	246	9,2	31,3	1,5
		262	3,8	60,7	2,1
		277	2,5	54,1	2,4
	360	368	8,3	66,9	4,2
		387	1,9	73,4	7,6
		406	1,4	33,0	10,9
	480	489	8,3	88,3	12,0
		511	1,9	13,8	18,0
		533	1,0	0,0	27,4
	600	611	6,4	73,7	22,9
		635	2,4	0,0	34,8
		660	1,0	0,0	39,8
70	210	216	14,6	16,8	0,7
		230	9,6	74,6	1,0
		245	6,2	55,7	1,1
	280	287	12,8	36,3	1,8
		304	7,0	66,7	2,7
		320	2,2	68,0	3,6
	420	429	6,2	77,0	7,2
		449	2,3	44,3	10,4
		470	0,7	4,0	13,3
	560	570	5,4	75,0	16,7
		594	1,5	0,0	22,8
		618	0,5	0,0	29,4
	700	712	5,2	66,8	31,7
		738	1,3	0,0	43,9
		765	1,0	0,0	78,1
80	240	246	16,9	47,6	0,9
		262	5,0	32,0	1,3
		277	6,1	64,1	1,6
	320	327	11,8	52,7	2,4
		345	7,1	70,8	3,8
		363	2,2	59,8	5,3
	480	489	10,2	71,9	9,4
		511	1,8	11,0	12,8
		533	1,3	0,0	15,8
	640	651	4,4	60,9	22,8
		676	2,2	0,0	33,6
		702	0,8	0,0	38,9
	800	813	4,3	27,2	45,0
		841	1,6	0,0	68,4
		869	1,1	0,0	87,0

Table 4.1: Results of the preprocessing phase on the sets of Medium and Large instances.

and efficiency, are the number of vertices and the number of edges. For the set of Medium instances, as the number of vertices increases, the percentage of time reduction increases too. This does not happen for the set of Large instances, where there are instances which were not optimally solved within the time limit, not even with the introduction of the preprocessing phase. Given the number of clusters and the number of vertices, the percentage of removed vertices decreases as the number of edges increases. Indeed, the more sparse is the graph, the higher is the probability to identify useless vertices. For some of the Medium instances the time reduction is negative, which denotes an increase of the computational time with the introduction of the preprocessing phase. This is not surprising, since those scenarios were optimally solved within 1 second even without the preprocessing phase. For the Large instances, the time reduction is greater than or equal to zero for all the scenarios, except the one with $k = 60$, $n = 180$ and $m = 185$, which is the smallest scenario in the set of Large instances. Finally, it is equal to zero for the scenarios with the highest density, which are the ones not optimally solved within the time limit.
The introduction of the preprocessing phase causes the $6,4\%$ of additional instances optimally solved within the time limit. Furthermore, it leads to a reduction of the computational time for almost all the instances that were optimally solved even without the preprocessing phase. In Figure 4.8, we reported the percentage of the removed vertices (green) and the percentage of the time reduction (blue), computed over Medium and Large instances. It is easy to see that as the number of clusters increases, both the percentage of removed vertices and the percentage of time reduction, decreases, except for the group of instances with $k = 70$, where there is a slight increase in the percentage of the time reduction with respect the instances with $k = 60$.

4.7.3 Medium and Large Instances

Computational results of the Branch and Cut algorithm for Medium instances are shown in Table 4.2. In column *Opt* is reported the value of the optimal solution, when available, or the value of the best solution computed within the time limit. In column *Nodes* there is the number of nodes of the Branch and Bound tree. In the next three columns there are the informations about the added cuts: column $GSEC(4.10)$ reports the number of inequalities (4.10), column $GSEC(4.4)$ reports the number of

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
30	90	93	20,7	4,2	7,8	41,0	88,2	47,0	0,2
		103	10,9	2,4	17,0	72,0	279,6	72,0	0,5
		112	7,3	1,6	18,4	78,4	363,0	81,4	0,6
	120	124	18,7	3,4	9,6	70,2	117,6	79,4	0,5
		135	16,0	2,8	14,8	94,4	217,4	84,8	1,0
		146	0,8	1,0	55,8	150,4	846,6	148,6	2,2
	180	185	8,1	2,2	14,6	138,2	260,6	124,4	1,7
		199	2,7	1,8	53,0	182,2	877,2	176,0	5,0
		212	1,8	1,2	161,6	241,4	2118,2	225,6	16,4
	240	246	12,5	2,2	18,4	173,4	348,6	155,0	3,8
		262	5,9	1,4	61,2	257,4	1064,6	208,0	10,5
		277	2,8	1,2	291,4	324,2	2925,8	268,6	43,6
	300	307	13,8	1,8	25,6	217,8	548,6	156,4	7,0
		324	4,5	1,0	132,8	305,4	1728,2	256,8	30,4
		342	2,1	1,0	146,2	395,0	3393,4	328,5	334,8
40	120	124	15,3	5,6	6,0	58,4	144,4	77,0	0,4
		135	6,5	4,2	25,0	99,6	639,4	115,8	1,1
		146	3,5	2,2	37,6	123,4	1023,4	143,0	1,9
	160	165	14,4	4,8	11,6	94,0	176,8	108,6	0,9
		177	4,5	3,0	54,8	156,0	1130,2	190,2	3,8
		190	1,8	2,0	80,2	199,6	2272,8	219,8	7,3
	240	246	10,7	4,0	26,2	175,2	455,8	187,2	3,9
		262	3,4	2,6	104,0	257,6	2338,0	262,4	16,4
		277	0,6	1,8	163,2	298,4	3708,2	294,0	95,7
	320	327	13,7	3,0	135,6	248,4	1266,2	282,2	12,3
		345	2,2	2,0	208,8	341,0	3846,2	373,4	84,0
		363	1,5	1,4	310,6	401,8	7153,6	398,6	252,3
	400	409	8,0	2,2	99,0	337,4	1648,4	403,2	27,3
		429	4,0	1,8	225,8	406,0	4787,8	364,8	135,2
		449	3,0	$1,4^{(2)}$	659,4	509,8	10555,6	555,6	2673,5
50	150	155	14,3	7,0	19,8	81,2	553,6	105,0	1,0
		167	6,9	5,2	44,4	123,6	1306,4	161,4	2,6
		179	3,6	3,2	54,4	156,8	1970,4	176,8	4,5
	200	206	11,9	5,4	32,2	128,6	568,8	170,4	2,3
		220	5,9	3,8	66,8	182,0	1885,4	188,8	7,8
		234	0,9	3,4	118,0	243,6	3830,0	258,2	25,7
	300	307	7,7	4,6	94,6	234,8	1450,4	296,8	12,6
		324	4,2	3,2	168,0	304,6	4171,8	389,0	59,0
		342	1,1	2,6	545,0	382,8	14284,2	494,2	547,0
	400	409	7,5	4,2	218,8	330,8	2921,8	459,8	47,0
		429	3,4	2,6	265,6	421,8	7489,0	449,6	223,8
		449	0,8	$2,4^{(2)}$	662,0	497,2	16335,2	511,2	1870,9
	500	510	7,3	3,4	150,0	432,2	3757,0	494,0	84,3
		532	2,6	2,2	672,8	503,2	12824,2	722,0	1190,2
		554	1,8	$2,8^{(5)}$	691,0	627,2	26013,8	765,4	3610,6

Table 4.2: Computational results for Medium instances.

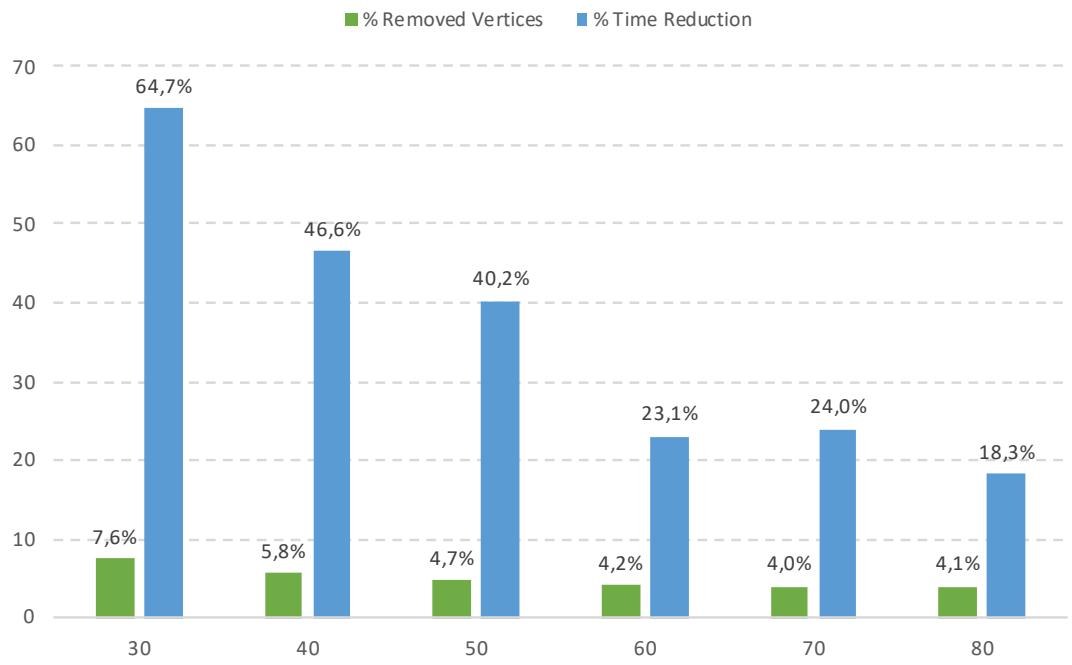

Figure 4.8: Bar chart reporting the percentage of removed vertices and the percentage of time reduction for instances with $k = 30, 40, 50, 60, 70, 80$.

inequalities (4.4), and column (4.13) contains the number of constraints (4.13) added. Finally, column *time* reports the computational time in seconds. If in a scenario there are a instances that were not optimally solved within the time limit, (a) appears close to the solution value, while we use the symbol " $-$ " if for that scenario there are instances for which no feasible solution has been found.

Only 9 over the 225 Medium instances were not optimally solved within the time limit, and they are the ones corresponding to the scenario with the highest density and the highest number of vertices with 40 clusters, and the ones corresponding to the scenarios with $k = 50$, $n = 400$ and $n = 500$, with the highest density. At the same k, as the number of vertices increases the computational time increases too. For instance, let us consider the instances with $k = 50$ and the maximum density: when n goes from 150 to 300 there is an increase of the computational time of the $12055,5\%$. Furthermore, the computational time is directly proportional to the density of the instances. For example, for the scenarios with $k = 40$ and $n = 400$, passing from $m = 409$ to $m = 429$ we have an increase of the computational time of the $395,2\%$, and when $m = 449$ there are

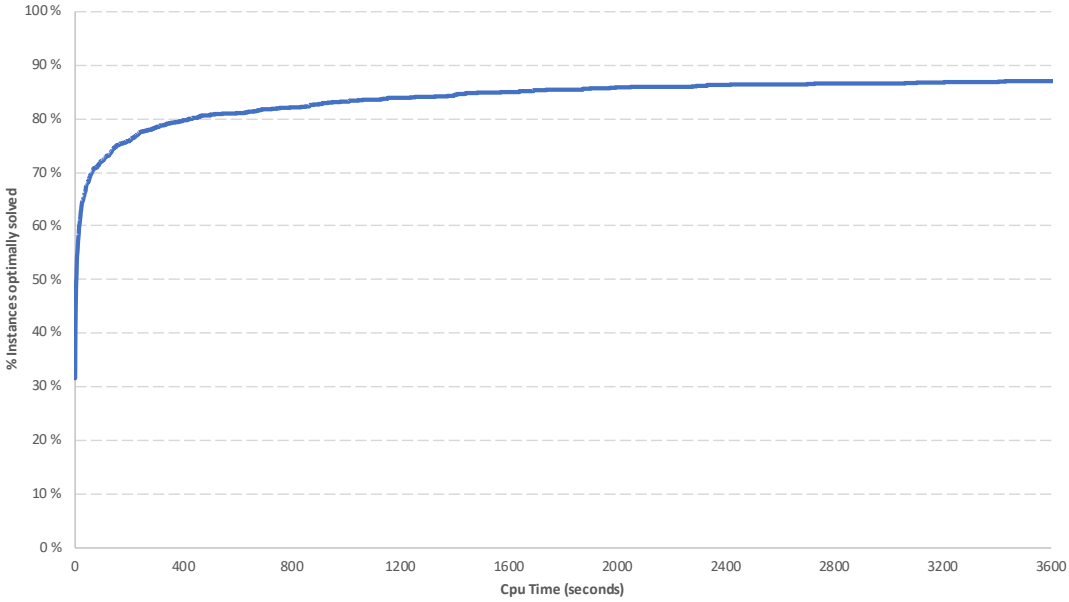

Figure 4.9: Percentage of optimally solved instances within the Cpu time.

two instances that were not optimally solved within the time limit. It is worth noting that GSEC(4.4) are in all cases the most generated inequalities, while the number of GSEC(4.10) and (4.13) inequalities generated are comparable.

Table 4.3 reports the computational results for the set of Large instances. As expected, the number of instances that were not optimally solved within the time limit is bigger for the set of Large instances, with respect the set of Medium instances. Indeed, 76 over 225 instances were not optimally solved, and in the scenario with the $k = 80$, $n = 800$ and $m = 869$, there is an instance for which we were not even able to find a feasible solution within one hour of computation. Even for the set of Large instances, the computational time is strictly related to the number of vertices and to the number of edges. Indeed, at the same k and the same n, the increase of the density implies the increase of the computational time. Analogously, at the same k and the same density, as the number of vertices increases the computational time increases too. The inequalities that were mostly generated are the GSEC(4.4).

Finally in Figure 4.9 we represent the percentage of optimally solved instances within the Cpu time. In more detail, the horizontal axis represents the Cpu time in seconds,

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
60	180	185	15,0	8,2	13,0	85,0	290,4	123,8	1,0
		199	7,4	5,6	36,6	125,0	1929,6	183,8	4,3
		212	2,7	3,6	89,8	192,0	4300,4	232,6	12,7
	240	246	9,2	6,8	52,0	170,0	1091,8	212,8	4,6
		262	3,8	5,0	195,0	242,2	4835,2	300,0	31,4
		277	2,5	3,2	216,2	276,2	9012,0	363,6	95,0
	360	368	8,3	5,4	153,4	283,8	2406,6	362,6	27,6
		387	1,9	3,6	538,4	391,8	13565,6	536,2	552,6
		406	1,4	3,2	546,0	471,4	25445,0	618,4	1473,3
	480	489	8,3	4,8	119,2	379,2	3214,4	507,6	63,1
		511	1,9	$3,8^{(2)}$	848,8	532,2	25330,0	824,8	2608,6
		533	1,0	$3,8^{(5)}$	682,4	598,6	38760,6	841,6	3610,6
	600	611	6,4	4,6	526,4	535,2	7561,6	675,6	532,5
		635	2,4	$4,4^{(5)}$	794,0	660,0	29471,6	908,6	3610,6
		660	1,0	$5,4^{(5)}$	584,2	730,0	32235,6	904,2	3610,6
70	210	216	14,6	9,2	26,6	101,6	879,8	158,4	2,0
		230	9,6	8,0	35,0	150,8	1862,6	190,4	5,1
		245	6,2	5,8	98,8	199,6	6038,2	279,2	19,5
	280	287	12,8	8,2	73,6	175,4	1137,8	246,2	6,4
		304	7,0	6,4	178,0	261,2	7880,0	364,0	66,1
		320	2,2	4,6	514,4	338,6	21344,4	474,0	589,4
	420	429	6,2	6,2	444,8	346,4	4168,0	525,0	99,1
		449	2,3	$5,2^{(1)}$	764,2	445,0	24362,0	730,2	1598,1
		470	0,7	$4,6^{(4)}$	667,0	532,0	41717,8	722,0	3308,1
	560	570	5,4	5,2	414,0	495,8	9252,6	698,0	482,5
		594	1,5	$5,2^{(5)}$	750,2	605,2	37025,8	978,4	3610,6
		618	0,5	$5,4^{(5)}$	548,0	701,0	39218,0	922,8	3610,6
	700	712	5,2	4,6	961,0	604,2	10274,6	768,8	1111,4
		738	1,3	$5,2^{(5)}$	612,0	747,2	34372,6	1101,6	3610,6
		765	1,0	$7,2^{(5)}$	177,8	758,8	30594,2	653,6	3610,6
80	240	246	16,9	11,6	8,4	103,4	663,8	141,0	1,8
		262	5,0	7,8	139,0	209,8	6263,2	319,2	28,4
		277	6,1	7,0	229,2	247,8	14087,0	363,6	95,1
	320	327	11,8	9,2	73,4	193,6	1936,4	282,8	11,0
		345	7,1	7,8	194,6	301,8	9133,4	395,0	102,3
		363	2,2	$5,6^{(1)}$	295,2	367,2	25444,2	437,6	862,1
	480	489	10,2	7,6	509,0	379,2	9058,4	612,2	280,1
		511	1,8	$6,2^{(4)}$	824,8	516,2	39485,2	783,0	3211,6
		533	1,3	$5,2^{(5)}$	554,6	564,6	50008,2	783,4	3610,6
	640	651	4,4	$6,2^{(1)}$	665,4	572,2	15514,4	803,0	1377,5
		676	2,2	$6,2^{(5)}$	488,4	690,6	39200,0	956,6	3610,6
		702	0,8	$6,6^{(5)}$	310,4	753,2	40553,8	884,8	3610,6
	800	813	4,3	$5,6^{(3)}$	854,2	720,8	21752,2	1045,4	2627,9
		841	1,6	$6,8^{(5)}$	374,8	809,6	34377,2	978,6	3610,6
		869	1,1	–	149,0	866,5	29370,5	811,3	3610,6

Table 4.3: Computational results for Large instances.

while the vertical axis represents the percentage of optimally solved instances within a fixed Cpu time. This means that as faster is the curve growth, as better is the performance of the Branch and Cut. We can see that the algorithm reaches the 70% of instances optimally solved in about 64 seconds, and almost the 80% in 7 minutes.

Chapter 5

The 2-Edge-Connected Minimum Branch Vertices Problem

5.1 Introduction

Given an undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, the *2-Edge-Connected Minimum Branch Vertices (2ECMBV) problem*, consists of determining a spanning subgraph $H = (V, E')$ of G, which satisfies the following properties:

1. H is 2-edge-connected, namely there exist at least 2 edge-disjoint paths between every pair of vertices;

2. the number of branch vertices in H is minimum, where a vertex v is *branch* in H if its degree is greater than 2.

Let us consider the graph G depicted in Figure 5.1(a). In Figure 5.1(b) there is a 2-edge-connected spanning subgraph of G with one branch vertex having degree four. Finally, in Figure 5.1(c) is shown the optimal solution to the 2ECMBV problem on G that is a 2-edge-connected spanning subgraph without branch vertices. Let us note that, the optimal solution to the 2ECMBV problem on a graph G is zero, if and only if there exists in G a *Hamiltonian cycle*, that is a cycle passing once through all its vertices, and in such a case, G is said to be *Hamiltonian*. For instance, the graph in Figure 5.1(a) is Hamiltonian. The existence of a Hamiltonian cycle in a given graph is a well studied problem, both from the algorithmic and the graph-theoretic point of

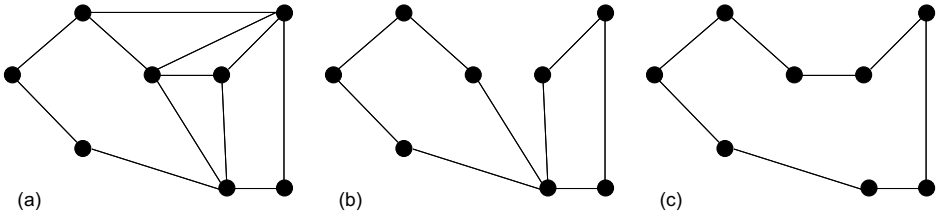

Figure 5.1: (a) An undirected graph $G = (V, E)$. (b) A 2-edge-connected spanning subgraph of G with one branch vertex. (c) An optimal solution to the 2ECMBV problem on G with zero branch vertices.

view. Trivially, if G is a complete graph, it is Hamiltonian. Moreover, if the graph satisfies any of a number of density conditions a Hamiltonian cycle is guaranteed to exist.

Applications to the 2ECMBV problem arises in the context of survivable optical networks. In the design of optical networks we need to take into account the costs necessary for the construction and the maintenance of the network, and also its survivability. A great impact on the total cost of the network is given by the the use of electronic devices with the task of splitting the optical signal. More in detail, whenever a light signal enters a node having degree greater than 2, it has to be split by a switch before being sent to the next nodes. Thus, to contain the costs, the number of switches has to be limited, and since a switch must be located in each branch vertex, it is necessary to design networks with the minimum number of branch vertices. For this motivation, Gargano et al. [25] introduced the Minimum Branch Vertices (MBV) problem, which consists of finding a spanning tree of a given graph with the minimum number of branch vertices. Carrabs et al. [3] introduced four IP formulations for the MBV problem, while Silvestri et al. [55] derived some valid inequalities and proposed a hybrid formulation with both undirected and directed variables, which was solved through a Branch and Cut algorithm. Landete et al. [38] investigated decomposition methods for degree dependent spanning tree problems. Finally, Merabet et al. [40] proposed a generalization of the MBV problem, where it has been introduced the notion of k-branch vertex, as a vertex with degree greater than $k+2$.

Furthermore, while dealing with network design problems, we also have to satisfy some survivability constraints, which regard the capacity of the network of restoring the service, in the event of node or link failure. Obviously, from this point of view a

tree like structure is the worst: whenever an edge fails, the service is over. The need to introduce *k-connectivity* requirements has led to the introduction of the 2ECMBV problem. Let us recall that a graph G if *k-edge-connected* if there exist at least k edge-disjoint paths between every pair of vertices. As the parameter k increases the grade of survivability increases too, but also the total cost of the network. In order to balance economic aspects and to provide a good level of protection against link's failure, we decide to consider the 2-edge-connectivity requirement. To the best of our knowledge, the 2ECMBV problem has never been introduced before. It is NP-hard, indeed finding an optimal solution to the 2ECMBV problem on a graph G in polynomial time is equivalent to establishing in polynomial time whether G is Hamiltonian. Grötschel, Monma and Stoer studied in detail network design problems with connectivity constraints in [29] and [30]. They proposed a model mixing edge and node survivability requirements, examined the dimension of the associated polytope and proved facet results. Monma et al. [41] introduced *Low-Connectivity Constrained Network Design Problems*, where each node v is characterized by an integer $r_v \in \{0,1,2\}$, and $min\{r_v, r_u\}$ node/edge-disjoint paths between every pair of nodes u, v are required. Further details about Low-Connectivity Constrained Network Design Problems can be found in Chapter 2.

In this chapter, we introduce an integer linear programming formulation for the 2ECMBV problem, some polyhedral results, and a polyhedral-based exact Branch and Cut algorithm. In Section 5.2 we introduce a mathematical formulation for the 2ECMBV problem, and we show some properties of the 2-edge-connected subgraphs. Section 5.3 is devoted to the polyhedral analysis: we determine the dimension of polyhedron and prove facet results for several inequalities. The separation problems for all the inequalities are showed in Section 5.4 and are embedded in a Branch and Cut algorithm.

5.2 Mathematical Formulation

The 2ECMBV problem can be formulated as an integer linear program (ILP) as follows. Let x_e be a binary variable equal to 1 if $e \in E$ is selected. Moreover, let y_v be a binary variable equal to 1 if $v \in V$ is a branch vertex. Given $E' \subseteq E$ and $V' \subseteq V$, we denote by $x(E') = \sum_{e \in E'} x_e$, and $y(V') = \sum_{v \in V'} y_v$. For $S, T \subseteq V$, we define $E(S:T) = \{\{u,v\} \in E : u \in S, v \in T\}$. We denote by $E(S) = E(S:S)$ the set of edges

having both extremes in S, and by $\delta(S) = E(S : V \setminus S)$ the set of edges incident on vertices belonging to S. We define $N(S) = \{u \in V \setminus S : \exists\{u,v\} \in \delta(S), v \in S\}$.When $S = \{v\}$, $\delta(\{v\})$ becomes simply $\delta(v)$, $N(\{v\})$ becomes $N(v)$, and we denote by $d(v)$ the cardinality of $\delta(v)$. Moreover, given $S \subseteq V$ and $v \in S$, by $\delta_S(v)$ we denote the subset $\delta(v) \cap E(S)$, and with $d_s(v)$ the cardinality of $\delta_S(v)$. Given $W \subseteq V$, we denote by $G[W]$, the subgraph of G induced by W, namely $G[W] = (W, E(W))$. Finally, given $v \in V$, we denote by $G \setminus \{v\}$ the subgraph $(V \setminus \{v\}, E \setminus \delta(v))$, and given $e \in E$ we denote by $G \setminus \{e\}$ the subgraph $(V, E \setminus \{e\})$.
In the following we give an ILP formulation for the 2ECMBV problem:

$$\text{Minimize } y = \sum_{v \in v} y_v \tag{5.1}$$

subject to

$$x(\delta(W)) \geq 2 \qquad W \subset V, W \neq \emptyset \tag{5.2}$$

$$x(\delta(v)) - 2 \leq (d(v) - 2)y_v \qquad v \in V \tag{5.3}$$

$$x_e \in \{0,1\} \qquad e \in E \tag{5.4}$$

$$y_v \in \{0,1\} \qquad v \in V \tag{5.5}$$

The objective function (5.1) minimizes the number of branch vertices. Constraints (5.2) ensure the 2-edge-connectivity. Indeed, according to Menger's theorem [39], the number of edge-disjoint paths between a pair of vertices is at least k, if and only if the smallest cut disconnecting those vertices has size greater than or equal to k. In what follows, we will refer to constraints (5.2) as *Cut Inequalities*. Constraints (5.3) ensure that variable y_v is equal to 1 if at least three edges in $\delta(v)$ are selected.

5.2.1 2-Edge-Connected Subgraph Properties

In this section we introduce some properties regarding 2-edge-connected subgraphs, that will be resumed in Section 5.3 for deriving polyhedral results about the 2ECMBV problem.

Definition 5.2.1. *An edge $e \in E$ is a bridge in G, if $G \setminus \{e\}$ is not connected.*

Let us consider the graph G shown in Figure 5.2(a). The edge $\{u,v\}$ is a bridge

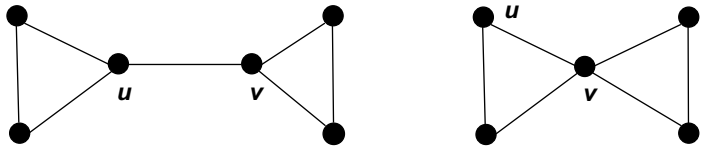

Figure 5.2: (a) A graph G such that the edge $\{u,v\}$ is a bridge in G. (b) A graph G such that the edge $\{u,v\}$ is essential in G and v is a cut vertex in G.

in G, since $G \setminus \{u,v\}$ is disconnected in two components. Let us denote by $B(G)$ the subset of E containing all the bridges of G,

$$B(G) = \{e \in E : e \text{ is a bridge}\}.$$

The following remark holds:

Remark 6. *G is 2-edge-connected if and only if $B(G) = \emptyset$.*

Obviously, if $B(G) \neq \emptyset$ there is no feasible solution to the 2ECMBV problem on G. Therefore, in what follows we assume that $B(G) = \emptyset$. We now introduce the notion of 2-edge-essential edges of a graph G.

Definition 5.2.2. *An edge $e \in E$ is 2-edge-essential in G, if the subgraph $G \setminus \{e\}$ is not 2-edge-connected.*

From now on, instead of 2-edge essential, we call these edges *essential*. Let us consider the graph G shown in Figure 5.2(b). The edge $\{u,v\}$ is essential in G, indeed $G \setminus \{u,v\}$ is not 2-edge-connected. From the definition it follows that any essential edge belongs to every feasible solution to the 2ECMBV problem. We denote by $ES(G)$ the set of all the essential edges in G,

$$ES(G) = \{e \in E : e \text{ is essential in } G\}.$$

It results that:

$$x_e = 1, \qquad \text{for any } e \in ES(G).$$

Finally, let us recall the definition of cut vertices of a graph:

Definition 5.2.3. *A vertex $v \in V$ is a cut vertex in G, if the subgraph $G \setminus \{v\}$ is not connected.*

For example, let us consider the graph G shown in Figure 5.2(b): the vertex v is a cut vertex, since $G \setminus \{v\}$ is disconnected in two components. Clearly, a cut vertex v is branch in any feasible solution to the 2ECMBV problem. Thus, if we denote by $V_B(G)$ the set of the cut vertices in G, it results that any feasible solution satisfies the following:

$$y_v = 1, \qquad \text{for any } v \in V_B(G).$$

Given a vertex $v \in V \setminus V_B$, since $G \setminus \{v\}$ is connected, there are two possibilities: $G \setminus \{v\}$ is also 2-edge-connected, or not. The following lemma holds:

Lemma 5.2.1. *If $G \setminus \{v\}$ is 2-edge-connected, then there exists a 2-edge-connected spanning subgraph G_v of G, such that v is not branch in G_v.*

Proof. Since $G \setminus \{v\}$ is 2-edge-connected, the subgraph $G_v = (V, (E \setminus \delta(v)) \cup \{e, f\})$ is 2-edge-connected, for any $e, f \in \delta(v)$. Moreover, v is not branch in G_v, as it has degree two in it. □

Let us now investigate the other alternative, that is the subgraph $G' = G \setminus \{v\}$ is not 2-edge-connected: this implies that the set of the bridges of G', $B(G') \subseteq E \setminus \delta(v)$, is not empty. Therefore, the subgraph obtained by removing all the edges in $B(G')$ from G' is not connected: let $C_1, ..., C_t$ be the connected components of $G' \setminus B(G')$. Now we distinguish two further cases:

(1) $|\delta(C_i) \cap B(G')| \leq 2$, for any $i \in \{1, ..., t\}$ (see Figure 5.3(a));

(2) there exists $i \in \{1, ..., t\}$, such that $|\delta(C_i) \cap B(G')| \geq 3$ (see Figure 5.3(b)).

Let us consider the graph G/v having as many vertices as the number of connected components in $G' \setminus B(G')$, namely t vertices called $u_1, ..., u_t$. There exists an edge $\{u_i, u_j\}$ in G/v if and only if there exists an edge between the components C_i and C_j in G'. Therefore, each edge in G/v corresponds to a bridge of G', in other words the edges incident on u_i are the edges belonging to $\delta(C_i) \cap B(G')$. It is easy to see that G/v is a tree.

Lemma 5.2.2. *If (1) holds, then there exists a 2-edge-connected spanning subgraph G_v of G, such that v is not branch in G_v.*

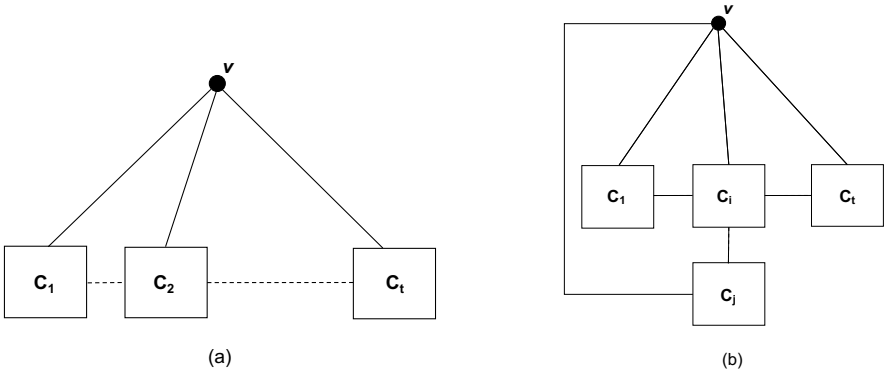

Figure 5.3: (a) A graph G such that $G \setminus \{v\}$ is not 2-edge-connected and $|\delta(C_i) \cap B(G')| \leq 2$, for any $i \in \{1,...,t\}$. (b) A graph G such that $G \setminus \{v\}$ is not 2-edge-connected and there exists $i \in \{1,...,t\}$ such that $|\delta(C_i) \cap B(G')| \geq 3$.

Proof. If (1) holds, the graph G/v is a path and then there are exactly two leaves in it, namely there are two connected components C_i and C_j with $i, j \in \{1,...,t\}$, such that $|\delta(C_i) \cap B(G')| = |\delta(C_j) \cap B(G')| = 1$, while $|\delta(C_k) \cap B(G')| = 2$, for any $k \neq i, j$. Since G is 2-edge-connected, there exists $e \in \delta(v) \cap \delta(C_i)$ and $f \in \delta(v) \cap \delta(C_j)$. The subgraph $G_v = (V, E_v)$, where $E_v = E \setminus \delta(v) \cup \{e, f\}$, is 2-edge-connected and v is not branch in it. □

For example, let us consider the graph G depicted in Figure 5.4(a). $G \setminus \{v_1\}$ is not 2-edge-connected, and $B(G \setminus \{v_1\}) = \{\{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}\}$. The connected components of $G \setminus \{v_1\} \setminus B(G \setminus \{v_1\})$ are $C_1 = \{v_2\}$, $C_2 = \{v_3\}$, $C_3 = \{v_4\}$ and $C_4 = \{v_5\}$, and since $|\delta(C_i) \cap B(G \setminus \{v_1\})| \leq 2$, for any $i = 1, 2, 3, 4$, we are in case (1). Therefore, the subgraph $G_{v_1} = (V, E_{v_1})$, with $E_{v_1} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_1, v_5\}\}$, is a 2-edge-connected spanning subgraph of G, where v_1 is not branch. It remains to examine case (2).

Lemma 5.2.3. *If (2) holds, vertex v is branch in any feasible solution to the 2ECMBV problem.*

Proof. When (2) holds, in the graph G/v there exists at least a vertex with degree three then there are at least three leaves, namely there are three connected components C_i, C_j and C_k such that $|\delta(C_i) \cap B(G')| = |\delta(C_j) \cap B(G')| = |\delta(C_k) \cap B(G')| = 1$. Since G is 2-edge-connected, there exist $e \in \delta(v) \cap \delta(C_i)$, $f \in \delta(v) \cap \delta(C_j)$ and $g \in \delta(v) \cap \delta(C_k)$.

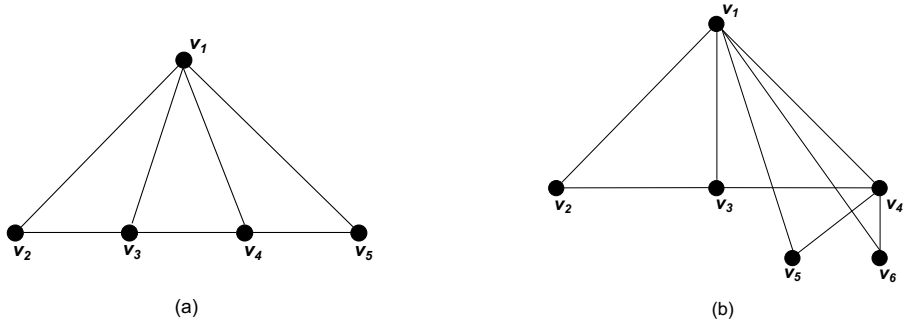

Figure 5.4: (a) A graph $G = (V,E)$ such that $G \setminus \{v_1\}$ is not 2-edge-connected and (1) holds. (b) A graph $G = (V,E)$ such that $G \setminus \{v_1\}$ is not 2-edge-connected and (2) holds.

To ensure 2-edge-connectivity e, f and g must be selected in any feasible solution, thus v is branch in any feasible 2ECMBV solution. □

Let us consider the graph G in Figure 5.4(b). $G \setminus \{v_1\}$ is not 2-edge-connected, and $B(G \setminus \{v_1\}) = \{\{v_2,v_3\},\{v_3,v_4\},\{v_4,v_5\},\{v_4,v_6\}\}$. The connected components of $G \setminus \{v_1\} \setminus B(G \setminus \{v_1\})$ are $C_1 = \{v_2\}$, $C_2 = \{v_3\}$, $C_3 = \{v_4\}$, $C_4 = \{v_5\}$ and $C_5 = \{v_6\}$. We are in case (2), since $|\delta(C_3) \cap B(G \setminus \{v_1\})| = 3$. Moreover, $|\delta(C_1) \cap B(G \setminus \{v_1\})| = |\delta(C_4) \cap B(G \setminus \{v_1\})| = |\delta(C_5) \cap B(G \setminus \{v_1\})| = 1$, thus edges $\{v_1,v_2\},\{v_1,v_5\}$ and $\{v_1,v_6\}$ must be selected in any 2ECMBV solution, and then v_1 is always branch.
It is worth emphasizing that, given a vertex $v \in V$ such that $G \setminus \{v\}$ is not 2-edge-connected, to guarantee 2-edge-connectivity must be selected at least an edge in $\delta(v) \cap \delta(C)$, for any connected component C of $G \setminus \{v\} \setminus B(G \setminus \{v\})$, such that $|\delta(C) \cap B(G \setminus \{v\})| = 1$. Thus, the following inequality holds for any feasible solution to the 2ECMBV problem:

$$x(\delta(v) \cap \delta(C)) \geq 1. \tag{5.6}$$

If $G \setminus \{v\}$ is not 2-edge-connected, we define the subset $\delta_b(v) \subseteq \delta(v)$, as the subset of edges of $\delta(v)$ containing any edge incident on a connected component C of $G \setminus \{v\} \setminus B(G \setminus \{v\})$, such that $|\delta(C) \cap B(G \setminus \{v\})| \geq 2$, namely

$$\delta_b(v) = \{e \in \delta(v) \cap \delta(C) : C \text{ connected component of } G \setminus \{v\} \setminus B(G \setminus \{v\}), \\ |\delta(C) \cap B(G \setminus \{v\})| \geq 2\}$$

For instance, in the graph shown in Figure 5.4(a), $\delta_b(v_1) = \{\{v_1,v_3\},\{v_1,v_4\}\}$. If

$G \setminus \{v\}$ is 2-edge-connected, we set $\delta_b(v) = \emptyset$. The following lemma holds:

Lemma 5.2.4. *A vertex $v \in V$ is branch in any feasible 2ECMBV solution containing an edge in $\delta_b(v)$.*

Proof. If $G \setminus \{v\}$ is not 2-edge-connected, since (5.6) holds, at least one edge in $\delta(v) \cap \delta(C)$ must be selected, for any connected component C of $G \setminus \{v\} \setminus B(G \setminus \{v\})$, such that $|\delta(C) \cap B(G \setminus \{v\})| = 1$. Furthermore, these connected components are at least two, thus selecting an edge in $\delta_b(v)$ causes v to be branch. □

Therefore, the following inequality is satisfied by any feasible 2ECMBV solution, for any $v \in V$ such that $G \setminus \{v\}$ is not 2-edge-connected:

$$y_v \geq x_e \qquad\qquad e \in \delta_b(v)$$

To resume, vertex $v \in V$ is branch in any feasible solution to the 2ECMBV problem, if at least one of the following occurs:

(a) $v \in V_B(G)$;

(b) $G \setminus \{v\}$ is not 2-edge-connected and there exists at least a connected component C in $G \setminus \{v\} \setminus B(G \setminus \{v\})$ such that $|\delta(C) \cap B(G \setminus \{v\})| \geq 3$.

Let us denote by $BR(G)$, the subset of vertices $v \in V$, such that v satisfies (a) or (b):

$$BR(G) = \{v \in V : v \text{ satisfies (a) or (b)}\}$$

Therefore, the following inequalities is satisfied by any feasible 2ECMBV solution:

$$y_v = 1, \qquad\qquad v \in BR(G).$$

Lemma 5.2.5. *Given a vertex $v \in V$, if $v \notin BR(G)$, then there exists a 2-edge-connected subgraph G_v, such that v is not branch in G_v.*

Proof. According to the definition, if $v \notin BR(G)$, then v satisfies neither the (a), nor the (b). Therefore, from Lemma 5.2.1 and Lemma 5.2.2 follows the assert. □

5.3 Polyhedral Analysis

Let us consider the following polytope,

$$P(G) = conv\{(x,y) \in \mathbb{R}^{|E|+|V|} : (x,y) \text{ satisfies } (5.2) - (5.5)\}$$

In this section, we derive some properties of the polytope $P(G)$. Given $V' \subseteq V$, we represent V' by its characteristic vector, $y^{V'} \in \mathbb{B}^n$, with $y_v^{V'} = 1$ if $v \in V'$, and $y_v^{V'} = 0$ otherwise. Analogously, given $E' \subseteq E$, let $x^{E'} \in \mathbb{B}^m$ be its characteristic vector, with $x_e^{E'} = 1$ if $e \in E'$, and $x_e^{E'} = 0$ otherwise. Moreover, we denote by $\mathbb{0}$ and $\mathbb{1}$ the vectors of all zeros and all ones, respectively.

Proposition 5.3.1. *Let $G = (V,E)$ be an undirected graph such that $P(G) \neq \emptyset$, the affine hull of $P(G)$ is the following:*

$$aff(P(G)) = \{(x,y) \in \mathbb{R}^{|V|+|E|} : x_e = 1, \forall e \in ES(G), y_v = 1, \forall v \in BR(G)\}$$

Proof. If $e \in ES(G)$, then $x_e = 1$, for any $(x,y) \in P(G)$. Furthermore, if $v \in BR(G)$, then $y_v = 1$, for any $(x,y) \in P(G)$. Let us consider the equation $a^T x + b^T y = c$, and let us suppose that it is satisfied by all points in $P(G)$. We may assume that $a_e = 0$, for any $e \in ES(G)$, and $b_v = 0$, for any $v \in BR(G)$. If $e \notin ES(G)$, then $(x^{E\setminus\{e\}}, \mathbb{1}) \in P(G)$ and obviously also $(x^E, \mathbb{1}) \in P(G)$, then $a_e = 0$. Therefore, $a_e = 0$, for any $e \in E$. Moreover, for any $v \in V \setminus BR(G)$, thanks to Lemma 5.2.5, there exists a feasible solution $G_v = (V,E_v)$ where v is not branch, thus $(x^{E_v}, \mathbb{1} \setminus \{v\})$, $(x^{E_v}, \mathbb{1})$ belong to $P(G)$ and then satisfy $a^T x + b^T y = c$. This implies that $b_v = 0$, for any $v \in V$. □

Corollary 5.3.2. *The dimension of $P(G)$ is equal to $|V| + |E| - |ES(G)| - |BR(G)|$.*

Corollary 5.3.3. *$P(G)$ is full-dimensional if and only if $ES(G) = \emptyset$ and $BR(G) = \emptyset$.*

Let us note that, if G is 3-edge-connected, then $ES(G) = \emptyset$. Therefore, in what follows, we assume that G is 3-edge-connected and $BR(G) = \emptyset$.

Proposition 5.3.4. *Inequality $x_e \leq 1$ is facet-defining for $P(G)$, for any $e = \{u,v\} \in E$ such that $e \notin \delta_b(u) \cup \delta_b(v)$.*

Proof. Let us consider the proper face $F_e^1 = \{(x,y) \in P(G) : x_e = 1\}$, and let $a^T x + b^T y = c$ be an equation satisfied by all $(x,y) \in F_e^1$. Since $BR(G) = \emptyset$ and Lemma

5.2.5 holds, for any $w \in V$ there exists a feasible solution $G_w = (V, E_w)$ where w is not branch. Moreover, $e \notin \delta_b(u) \cup \delta_b(v)$, then it is always possible to choose $G_w = (V, E_w)$ such that $e \in E_w$ and w is not branch in it. Thus $(x^{E_w}, \mathbb{1} \setminus \{w\})$ and $(x^{E_w}, \mathbb{1})$ belong to F_e^1. This implies that $b_w = 0$, for any $w \in V$. G is 3-edge-connected, then for any $f \in E \setminus \{e\}$, $(x^{E\setminus\{f\}}, \mathbb{1}) \in F_e^1$. Since $(x^E, \mathbb{1})$ belongs to F_e^1 too, it follows that $a_f = 0$, for any $f \in E \setminus \{e\}$. Therefore, equation $a^T x + b^T y = c$ reduces to $a_e x_e = c$. Finally, since $(x^E, \mathbb{1})$ belongs to F_e^1, $a_e = c$, thus $a^T x + b^T y = c$ is a multiple of $x_e = 1$. □

Proposition 5.3.5. *Inequality $x_e \geq 0$ is facet-defining for $P(G)$, for any $e \in E$, such that $G \setminus \{e\}$ is 3-edge-connected and $BR(G \setminus \{e\}) = \emptyset$.*

Proof. From Corollary 5.3.2 follows that $dim(P(G \setminus \{e\})) = |V| + |E \setminus \{e\}| - |ES(G \setminus \{e\})| - |BR(G \setminus \{e\})| = |V| + |E| - 1$. Thus, there exist $|V| + |E|$ affinely independent incidence vectors which satisfy $x_e \geq 0$ with equality. □

Proposition 5.3.6. *Inequality $y_v \leq 1$, is facet-defining for $P(G)$, for any $v \in V$.*

Proof. Let us consider the proper face $F_v^1 = \{(x,y) \in P(G) : y_v = 1\}$, and let $a^T x + b^T y = c$ be an equation satisfied by all $(x,y) \in F_v^1$. For any $u \in V \setminus \{v\}$, let $G_u = (V, E_u)$ be a 2-edge-connected subgraph such that u is not branch. It is easy to see that both $(x^{E_u}, \mathbb{1} \setminus \{u\})$ and $(x^{E_u}, \mathbb{1})$ belong to F_v^1, thus $b_u = 0$, for any $u \in V \setminus \{v\}$. For each $e \in E$, $(x^{E\setminus\{e\}}, \mathbb{1})$ and $(x^E, \mathbb{1})$ belong to F_v^1, and this implies that $a_e = 0$, for any $e \in E$. Therefore equation $a^T x + b^T y = c$ reduces to $b_v y_v = c$. The vector $(x^E, \mathbb{1})$ belongs to F_v^1, then $b_v = c$ and $a^T x + b^T y = c$ is a multiple of $y_v = 1$. □

Proposition 5.3.7. *Inequality $y_v \geq 0$, is not facet-defining for $P(G)$, for any $v \in V$.*

Proof. Inequality $y_v \geq 0$ is dominated by inequality (5.3), since $x(\delta(v)) \geq 2$ holds. □

Proposition 5.3.8. *Inequality (5.2) is facet-defining for $P(G)$, for any subset $W \subset V$, $W \neq \emptyset$, such that:*

1. *$G[W]$ and $G[V \setminus W]$ are 3-edge-connected;*
2. *$BR(G[W]) = \emptyset$ and $BR(G[V \setminus W]) = \emptyset$.*

Proof. Given $W \subset V$, $W \neq \emptyset$, let $F_W = \{(x,y) \in P(G) : x(\delta(W)) = 2\}$ be a proper face, and let $a^T x + b^T y = c$ be an equation satisfied by all $(x,y) \in F_W$. $BR(G[W]) = \emptyset$, then

for any $v \in W$ there exists a 2-edge-connected subgraph in $G[W]$, $G_v^W = (W, E_v^W)$, such that v is not branch in G_v^W. Moreover G is 3-edge-connected and $BR(G) = \emptyset$, then there exist $e, f \in \delta(W)$, and at least one of them does not belong to $\delta(v)$, let $e \notin \delta(v)$. If $f \notin \delta(v)$ too, it is easy to see that $G_v = (V, E_v)$, with $E_v = E_v^W \cup E(V \setminus W) \cup \{e, f\}$ is a 2-edge-connected subgraph such that v is not branch in G_v. On the other hand, if $f \in \delta(v)$, since $BR(G) = \emptyset$, it is always possible to remove an edge $g \in \delta(v) \cap E_v^W$, and $G_v = (V, E_v)$, with $E_v = (E_v^W \setminus \{g\}) \cup E(V \setminus W) \cup \{e, f\}$ is a 2-edge-connected subgraph such that $d_{G_v}(v) = 2$. Thus, $(x^{E_v}, \mathbb{1} \setminus \{v\})$ belongs to F_W. Moreover, $(x^{E_v}, \mathbb{1})$ belongs to F_W too, then $b_v = 0$, for any $v \in W$. Similarly, it can be proven that $b_v = 0$, for any $v \in V \setminus W$. $G[W]$ is 3-edge-connected, then given $e \in E(W)$, $G[W] \setminus \{e\}$ is 2-edge-connected. Then, $G_e = (V, E_e)$, with $E_e = (E(W) \cup E(V \setminus W) \cup \{f, g\}) \setminus \{e\}$ and $f, g \in \delta(W)$, is a 2-edge-connected subgraph. Thus, both $(x^{E_e}, \mathbb{1})$ and $(x^{E_e \cup \{e\}}, \mathbb{1})$ belong to F_W. This implies that $a_e = 0$, for any $e \in E(W)$. In the same way, we can prove that $a_e = 0$, for any $e \in E(V \setminus W)$. Thus, equation $a^T x + b^T y = c$ reduces to $\sum_{e \in \delta(W)} a_e x_e = c$. Let us note that there exist at least $e, f, g \in \delta(W)$, with $e \neq f \neq g$, then $G_{e,f} = (V, E_{e,f})$ and $G_{e,g} = (V, E_{e,g})$, with $E_{e,f} = (E \setminus \delta(W)) \cup \{e, f\}$ and $E_{e,g} = (E \setminus \delta(W)) \cup \{e, g\}$, are two 2-edge-connected subgraphs in G, such that their incidence vectors belong to F_W. This implies that $a_f = a_g$. Since it can be done for any $e, f, g \in \delta(W)$, it follows that $a_e := a$, for any $e \in \delta(W)$. Finally, $(x^{E_{e,f}}, \mathbb{1})$ belongs to F_W, thus $c = 2a$. Therefore, we obtain that equation $a^T x + b^T y = c$ reduces to $ax(\delta(W)) = 2a$, that is a multiple of (5.2). □

Proposition 5.3.9. *Given $v \in V$, inequality (5.3) is facet-defining for $P(G)$ if $G \setminus \{v\}$ is 2-edge-connected and for any $e = \{u, v\} \in \delta(v)$, $e \notin \delta_b(u)$.*

Proof. If $G \setminus \{v\}$ is not 2-edge-connected, it is easy to see that inequality (5.3) is dominated by inequalities involving the edges in $\delta_b(v)$. Given $v \in V$, let us consider the proper face $F_v = \{(x, y) \in P(G) : x(\delta(v)) - (d(v) - 2)y_v = 2\}$, and let $a^T x + b^T y = c$ be an equation satisfied by all $(x, y) \in F_v$. Since $BR(G) = \emptyset$, for any $u \in V \setminus \{v\}$ there exists a 2-edge-connected subgraph $G_u = (V, E_u)$ in G, such that u is not branch in G_u. If $u \in N(v)$, there exists $e = \{u, v\} \in \delta(v)$, but $e \notin \delta_b(u)$, then it is possible to consider a subgraph G_u such that all the edges in $\delta(v)$ belong to E_u and where u is not branch. It is easy to see that both $(x^{E_u}, \mathbb{1} \setminus \{u\})$ and $(x^{E_u}, \mathbb{1})$ belong to F_v. This implies that $b_u = 0$, for any $u \in V \setminus \{v\}$. For each $e \notin \delta(v)$, $(x^{E \setminus \{e\}}, \mathbb{1}) \in F_v$. Moreover $(x^E, \mathbb{1})$ belongs to

F_v too, thus $a_e = 0$, for any $e \in E \setminus \delta(v)$. Equation $a^T x + b^T y = c$ reduces to

$$\sum_{e \in \delta(v)} a_e x_e + b_v y_v = c$$

G is 3-edge-connected, then there exist $e, f, g \in \delta(v)$, such that $e \neq f \neq g$. $G \setminus \{v\}$ is 2-edge-connected, thus $G_{e,f} = (V, E_{e,f})$ and $G_{e,g} = (V, E_{e,g})$, with $E_{e,f} = (E \setminus \delta(v)) \cup \{e, f\}$ and $E_{e,g} = (E \setminus \delta(v)) \cup \{e, g\}$, are two 2-edge-connected subgraphs in G, such that their incidence vectors belong to F_v. This implies that $a_f = a_g$. Since it can be done for any $e, f, g \in \delta(v)$, it follows that $a_e := a$, for any $e \in \delta(v)$. Moreover, there exist $e, f \in \delta(v)$, such that $(x^{E_{e,f}}, \mathbb{1} \setminus \{v\})$ belongs to F_v, thus $c = 2a$. Finally, $(x^E, \mathbb{1}) \in F_v$, hence

$$a\, d(v) + b_v = 2a \Rightarrow b_v = a\, (2 - d(v)).$$

We obtain that equation $a^T x + b^T y = c$ becomes

$$a\, x(\delta(v)) + a\, (2 - d(v)) y_v = 2a.$$

Since it is a multiple of $x(\delta(v)) + (2 - d(v)) y_v = 2$, inequality (5.3) is facet-defining for $P(G)$. □

The following family of valid inequalities has been introduced in [55].

Proposition 5.3.10. *Given $v \in V$ and $S \subseteq \delta(v)$, with $|S| \geq 3$,*

$$x(S) - 2 \leq (|S| - 2) y_v \tag{5.7}$$

is valid for $P(G)$.

Proof. Whenever we take a subset $S \subseteq \delta(v)$, if more than two edges in S are selected, then v must be branch. □

Proposition 5.3.11. *Given $v \in V$ and $S \subseteq \delta(v)$, with $|S| \geq 3$, inequality (5.7) is facet-defining for $P(G)$, if $G \setminus \{v\}$ is 2-edge-connected and for any $e = \{u, v\} \in S$, $e \notin \delta_b(u)$.*

Proof. Since inequality (5.7) is a generalization of (5.3), the proof is almost the same of the one of Proposition 5.3.9. □

Let us introduce a new family of valid inequalities:

Proposition 5.3.12. *Given $S \subset V$, with $|S| \geq 3$, and $T \subseteq E(S)$ such that $d_T(v) \in \{2,3\}$, for any $v \in S$, inequality*

$$y(S) \geq x(T) - |S| + 1 \tag{5.8}$$

is valid for $P(G)$.

Proof. Let S be a proper subset of vertices, with $|S| \geq 3$, and let T be a subset of $E(S)$, such that $d_T(v) \in \{2,3\}$, for any $v \in S$. We distinguish two cases:

- $d_T(v) = 2$, for any $v \in S$: under this hypothesis T is a cycle with no chords, thus $|T| = |S|$. Inequality (5.8) ensures that if all the edges in T are selected, then at least a vertex in S is branch.
- there exist t vertices, $v_1, ..., v_t \in S$, such that $d_T(v_1) = 3 = d_T(v_2) = ... = d_T(v_t)$: in this case $|T| = |S| + t/2$, where t is an even number. Inequality (5.8) ensures that if $x(T) = |T|$, then in S there are at least $t/2 + 1$ branch vertices.

□

Proposition 5.3.13. *Let $S \subset V$, with $|S| \geq 4$, be a subset of vertices satisfying the followings:*

1. *there exist $u, v \in S$ such that $d_S(u) = d_S(v) = 3$;*
2. *$d_S(w) = 2$, for any $w \in S \setminus \{u, v\}$;*
3. *$G[V \setminus S]$ is 3-edge-connected and $BR(G[V \setminus S]) = \emptyset$;*
4. *$|E(w : V \setminus S)| \geq 3$, for any $w \in S \setminus \{u, v\}$;*
5. *$|E(u : V \setminus S)|, |E(v : V \setminus S)| \geq 2$.*

Inequality,

$$y(S) \geq x(E(S)) - |S| + 1 \tag{5.9}$$

is facet-defining for $P(G)$.

Proof. Given $S \subset V$ satisfying hypothesis 1-5, let $F_S = \{(x,y) \in P(G) : y(S) - x(E(S)) = -|S|+1\}$ be a proper face. Let us consider the equation $a^T x + b^T y = c$, satisfied by any point $(x,y) \in F_S$. We will proceed by steps:

- Given a vertex $\bar{w} \in V \setminus S$, since hypothesis 3 holds, there exists a 2-edge-connected subgraph in $G[V \setminus S]$, $\bar{G}_{\bar{w}} = (V \setminus S, \bar{E}_{\bar{w}})$, where $\bar{w}$ is not branch. There exist $f,g \in (\delta(u) \cup \delta(v)) \cap \delta(S)$ such that $f,g \notin \delta(\bar{w})$, then the subgraph $G_{\bar{w}} = (V, E_{\bar{w}})$, with $E_{\bar{w}} = E(S) \cup \bar{E}_{\bar{w}} \cup \{f,g\}$, is a feasible 2ECMBV solution where $\bar{w}$ is not branch. In such a way the incidence vector $(x^{E_{\bar{w}}}, \mathbb{1} \setminus \{\bar{w}, S \setminus \{u,v\}\})$ belongs to F_S. Therefore, $(x^{E_{\bar{w}}}, \mathbb{1} \setminus \{\bar{w}, S \setminus \{u,v\}\})$ and $(x^{E_{\bar{w}}}, \mathbb{1} \setminus (S \setminus \{u,v\}))$ belong to F_S, for any $\bar{w} \in V \setminus S$, and this implies that $b_{\bar{w}} = 0$, for any $\bar{w} \in V \setminus S$.

- Given $e \in E(V \setminus S)$, thanks to hypothesis 3, there exists a 2-edge-connected subgraph in $G[V \setminus S]$ not containing edge e, $\bar{G}_e = (V \setminus S, \bar{E}_e)$. Thus $G_e = (V, E_e)$, with $E_e = E(S) \cup \bar{E}_e \cup \{f,g\}$ and $f,g \in (\delta(u) \cup \delta(v)) \cap \delta(S)$ is a feasible solution not containing e. Since $(x^{E_e}, \mathbb{1} \setminus (S \setminus \{u,v\})) \in F_S$ and $(x^{E_e \cup \{e\}}, \mathbb{1} \setminus (S \setminus \{u,v\})) \in F_S$, it results that $a_e = 0$, for any $e \in E(V \setminus S)$.

- Let e be an edge in $\delta(S)$. Let us consider the subgraph $G' = (V, E')$ of G, with $E' = E(S) \cup E(V \setminus S) \cup \{f,g\}$, with $f,g \in (\delta(u) \cup \delta(v)) \cap \delta(S)$. Since hypothesis 5 holds, it is always possible to choose $f,g \neq e$, thus $(x^{E'}, \mathbb{1} \setminus (S \setminus \{u,v\}) \in F_S$ and $(x^{E' \cup \{e\}}, \mathbb{1} \setminus (S \setminus \{u,v\})) \in F_S$. It follows that $a_e = 0$, for any $e \in \delta(S)$.

- Let us consider $w, w' \in S$. The subgraph $G_w = (V, E_w)$, such that $E_w = E(S) \setminus \{u,v\} \cup E(V \setminus S) \cup \{e_w, e'_w\}$, with $e, e' \in \delta(w) \cap \delta(S)$, is a feasible 2ECMBV solution, and $(x^{E_w}, \mathbb{1} \setminus (S \setminus \{w\}))$ belongs to F_S. Similarly, $G_{w'} = (V, E_{w'})$, such that $E_{w'} = E(S) \setminus \{u,v\} \cup E(V \setminus S) \cup \{e, e'\}$, with $e_{w'}, e'_{w'} \in \delta(w') \cap \delta(S)$, is a feasible solution ans $(x^{E_{w'}}, \mathbb{1} \setminus (S \setminus \{w'\}))$ belongs to F_S. This implies that $b_w = b_{w'}$. Therefore, $b := b_w$, for any $w \in S$. The equation $a^T x + b^T y = c$ has become:

$$\sum_{e \in E(S)} a_e x_e + b\, y(S) = c.$$

- Let us consider $f = \{w, w'\} \in E(S)$. The subgraph $G_S = (V, E_S)$, with $E_S = E(S) \cup E(V \setminus S) \cup \{e, e'\}$, and $e, e' \in (\delta(u) \cup \delta(v)) \cap \delta(S)$, is a feasible solution,

thus $(x^{E_S}, \mathbb{1} \setminus (S \setminus \{u,v\}))$ belongs to F_S. Now we build a feasible solution, $G_f = (V, E_f)$, which does not contain f. Let us distinguish three cases:

- $f = \{u,v\}$: the subgraph $G_f = (V, E_f)$, with $E_f = E(S) \setminus \{u,v\} \cup E(V \setminus S) \cup \{e, e'\}$, $e, e' \in \delta(u) \cap \delta(S)$, is such that $(x^{E_f}, \mathbb{1} \setminus (S \setminus \{u\}))$ belongs to F_S. Hence, $(x^{E_S}, \mathbb{1} \setminus (S \setminus \{u,v\})) \in F_S$ implies $\sum_{e \in E(S)} a_e + 2b = c$, while $(x^{E_f}, \mathbb{1} \setminus (S \setminus \{u\})) \in F_S$ implies $\sum_{e \in E(S)} a_e - a_{\{u,v\}} + b = c$. It results that $a_{\{u,v\}} = -b$.
- $f \in \delta(u)$ ($f \in \delta(v)$): let us suppose that $f = \{u,w\}$, with $w \in S$. The set $E_f = E(S) \setminus \{f\} \cup E(V \setminus S) \cup \{e, e'\}$, with $e \in \delta(w) \cap \delta(S)$ and $e' \in \delta(v) \cap \delta(S)$, is such that that $(x^{E_f}, \mathbb{1} \setminus (S \setminus \{v\})) \in F_S$. Thus, again we have that $a_f = -b$.
- $f \notin \delta(u) \cup \delta(v)$: in this case $f = \{w, w'\}$, with $w, w' \in S \setminus \{u,v\}$. Choosing $E_f = E(S) \setminus \{f, \{u,v\}\} \cup E(V \setminus S) \cup \{e, e'\}$, with $e \in \delta(w) \cap \delta(S)$ and $e' \in \delta(w') \cap \delta(S)$, it results that $(x^{E_f}, \mathbb{1} \setminus S) \in F_S$. Therefore, we have that $\sum_{e \in E(S)} a_e - a_f - a_{\{u,v\}} = c$. Furthermore, $(x^{E_S}, \mathbb{1} \setminus (S \setminus \{u,v\}))$ belongs to F_S, then $\sum_{e \in E(S)} a_e + 2b = c$. This implies that $2b = -a_f - a_{\{u,v\}}$. Since we have already showed that $a_{\{u,v\}} = -b$, we can conclude that $a_f = -b$.

Therefore, the equation $a^T x + b^T y = c$ has become:

$$-b\, x(E(S)) + b\, y(S) = c.$$

- The subgraph $G_S = (V, E_S)$, such that $E_S = E(S) \cup E(V \setminus S) \cup \{e, e'\}$, with $e, e' \in (\delta(u) \cup \delta(v)) \cap \delta(S)$, is such that its incident vector belongs to F_S. Therefore, we obtain

$$-b\,(|S| + 1) + 2b = c \rightarrow c = -b\,|S| + b.$$

The equation $a^T x + b^T y = c$ has become:

$$-b\, x(E(S)) + b\, y(S) = -b\,|S| + b,$$

that is a multiple of (5.9). □

5.4 Branch and Cut Algorithm

We designed a Branch and Cut algorithm for the 2ECMBV problem based on the ILP formulation introduced in Section 5.2. A preprocessing procedure can be carried out before executing the algorithm by using the properties introduced in Subsection 5.2.1. More in detail, for any $e \in E$, if $e \in ES(G)$ we add to the model the constraint $x_e = 1$. Then, we look for cut vertices in G, and we add to the model the following constraints:

$$\begin{aligned} y_v &= 1, && v \in V_B(G) \\ x(\delta(v)) &\geq 2c_v, && v \in V_B(G) \end{aligned}$$

where c_v is the number of connected components in $G \setminus \{v\}$. The steps of the Branch and Cut algorithm are summarized in Algorithm 3. The initial linear program (LP) model is the following:

$$\text{Minimize } y = \sum_{v \in v} y_v$$

subject to

$$\begin{aligned} x(\delta(v)) &\geq 2 + y_v && v \in V \\ x(\delta(v)) - 2 &\leq (d(v) - 2)y_v && v \in V \\ 0 &\leq x_e \leq 1 && e \in E \\ 0 &\leq y_v \leq 1 && v \in V \end{aligned}$$

This LP model is obtained by considering only the Cut Inequalities corresponding to $W = \{v\}$, for any $v \in V$, and relaxing the integrality constraints on the variables of the original formulation. For any subproblem L', we compute the optimal LP solution $(x^*_{LP}(L'))$ and if it is feasible for the ILP and better than the incumbent solution, then the incumbent is updated (line 14). Otherwise, if the LP solution is not feasible, we search for violated constraints (5.2), (5.7) and (5.8) (lines 16-28). We search for violated constraints (5.8) only if the current LP solution satisfies the Cut Inequalities (lines 20-24). This procedure is repeated until there exist inequalities violated by the current LP solution. When improvements are no longer possible, we branch on the variables using the default parameters of CPLEX (lines 29-31). In the following subsection, we

Algorithm 3: Branch and Cut algorithm for the 2ECMBV problem

Input: integer linear program ILP
Output: optimal solution of ILP

```
1  L = ∅;
2  x' ← null;                    //incumbent
3  z(x') ← ∞;                    //value of the incumbent
4  L_0 ← first subproblem;
5  L ← L_0;
6  while L ≠ ∅ do
7      found ← true;
8      L' ← subproblem from L;
9      while found == true do
10         found ← false;
11         x*_LP(L') ← optimal LP solution of the subproblem L';
12         if z(x*_LP(L')) < z(x') then
13             if x*_LP(L') is feasible then
14                 x' ← x*_LP(L');          //update incumbent
15             else
16                 search for violated constraints (5.2);
17                 if violated constraints (5.2) are identified then
18                     add them to the model;
19                     found ← true;
20                 else if violated constraints (5.2) are not identified then
21                     search for violated constraints (5.8);
22                     if violated constraints (5.8) are identified then
23                         add them to the model;
24                         found ← true;
25                 search for violated constraints (5.7);
26                 if violated constraints (5.7) are identified then
27                     add them to the model;
28                     found ← true;
29                 else
30                     do the branching → subproblems L_1, L_2;
31                     L ← L_1, L_2;
```

describe the separation procedures for each class of valid inequalities.

5.4.1 Separation Procedures

Cut Inequalities are separated by computing the minimum cut in a graph with capacities given by the current LP solution. More in detail, to determine the minimum-cut we use the Gomory-Hu algorithm [27], which requires $n-1$ maximum flow computations. If the capacity of the minimum cut is less than 2, we add the corresponding violated inequalities (5.2).

The separation procedure for (5.7) was introduced by Lucena et al. [1]. Given a feasible solution $(\tilde{x},\tilde{y})$ for the LP relaxation, and $v \in V$ with $d(v) \geq 4$, let us consider the set $\{\tilde{x}_{e_1},...,\tilde{x}_{e_{d(v)}}\}$ containing the variables associated to the edges in $\delta(v)$, ordered in a non-increasing way. Given $k \in \{3,...,d(v)-1\}$, we compute $\sum_{i=1}^{k} \tilde{x}_{e_i} - (k-2)\tilde{y}_v$: if this value is greater than 2, we have identified a subset S with $|S| = k$, for which (5.7) is violated and that provides the largest value for the left-hand side of the inequality. For any $v \in V$ with $d(v) \geq 4$, we at first search for violated inequalities (5.7) with $S \subseteq \delta(v)$ and $|S| = 3$ and we add a subset of the most violated ones. Then, we apply the described procedure for $k \in \{4,...,d(v)-1\}$ and add to the model at most one violated constraint for each k.

We separate inequalities (5.8) heuristically, as described below. Given the current relaxed solution $(\tilde{x},\tilde{y})$, let us consider the following minimization problem:

$$min\{\tilde{y}(S) - \tilde{x}(E(S)) + |S| : \forall S \subset V, S \neq \emptyset\} \tag{5.10}$$

It is easy to see that if there exist violated inequalities (5.8), then the minimum value of (5.10) is less than one. Indeed, if there exist $S \subset V$, $S \neq \emptyset$, and $T \subseteq E(S)$, with $d_T(v) \in \{2,3\}$, for any $v \in S$, such that $\tilde{y}(S) - \tilde{x}(T) + |S| < 1$, it results that $\tilde{y}(S) - \tilde{x}(E(S)) + |S| \leq \tilde{y}(S) - \tilde{x}(T) + |S| < 1$. On the contrary, if the minimum of (5.10) is greater than or equal to 1, then there are no violated constraints (5.8). To solve the problem (5.10) we solve a maximum flow problem on an auxiliary directed graph, $G^* = (V^*, A^*)$ with capacities depending on the current solution, build as follows. The set of nodes V^* is obtained by adding to V a source node s and a sink node t, $V^* = V \cup \{s,t\}$. Every edge

$e = \{u,v\} \in E$ is replaced in G^* by two directed arcs, (u,v) and (v,u), with capacities $c_{uv} = c_{vu} = \frac{1}{2}\tilde{x}_e$. Finally, source node s is connected to any node $v \in V$, by arcs (s,v), with capacities $c_{sv} = \frac{\tilde{x}(\delta(v))}{2} - 1$, while sink node t is connected to each node $v \in V$, by arcs (v,t), with capacity $c_{vt} = \tilde{y}_v$. It is worth noting that $c_{uv} \geq 0$, for any $(u,v) \in E^*$, indeed the current LP solution satisfies Cut Inequalities (5.2).
Given $S \subseteq V$, let us consider the cut set $\{S \cup \{s\} : V \cup \{t\} \setminus S\}$ in G^*. It results that:

$$
\begin{aligned}
c(\{S \cup \{s\} : V \cup \{t\} \setminus S\}) &= \sum_{v \in V \setminus S} \big(\frac{\tilde{x}(\delta(v))}{2} - 1\big) + \tilde{y}(S) + c(S : V \setminus S) = \\
&= \tilde{y}(S) + \sum_{v \in V} \big(\frac{\tilde{x}(\delta(v))}{2} - 1\big) - \sum_{v \in S} \big(\frac{\tilde{x}(\delta(v))}{2} - 1\big) + \frac{1}{2}\tilde{x}(E(S : V \setminus S)) = \\
&= \tilde{y}(S) + |S| - \sum_{v \in S} \frac{\tilde{x}(\delta(v))}{2} + \frac{1}{2}\tilde{x}(E(S : V \setminus S)) + \sum_{v \in V} \big(\frac{\tilde{x}(\delta(v))}{2} - 1\big) = \\
&= \tilde{y}(S) + |S| - \tilde{x}(E(S)) + \sum_{v \in V} \big(\frac{\tilde{x}(\delta(v))}{2} - 1\big)
\end{aligned}
$$

Thus, solving the minimization problem (5.10) is equivalent to find a minimum $\{s,t\}$-cut in G^*. Assuming that $V = \{v_1, ..., v_n\}$, to ensure $S \neq \emptyset$, for any $h = 1, ..., n-1$ we solve a maximum flow problem with capacities as defined before, except for $c_{s,v_h} = +\infty$ and, if $h \geq 2$, we set $c_{v_1,t} = ... = c_{v_{h-1},t} = +\infty$. We record the cut-set that is minimum over the $n-1$ iterations. Once computed the minimum $\{s,t\}$-cut in G^*, we have a subset of vertices S such that $\tilde{y}(S) < \tilde{x}(E(S)) - |S| + 1$. If $d_S(v) \in \{2,3\}$, for any $v \in S$, we have identified a violated inequality (5.8), otherwise we check whether there exists a subset $T \subseteq E(S)$, such that $d_T(v) \in \{2,3\}$, for any $v \in S$, and $\tilde{y}(S) < \tilde{x}(T) - |S| + 1$. For this purpose, starting from the empty set $T = \emptyset$, for any $v \in S$ we add to T at most three edges e in $\delta(v) \cap E(S)$, choosing the ones with the highest value $\tilde{x}_e$ and ensuring that $d_T(v) \in \{2,3\}$. Finally we check if inequality (5.8) corresponding to T is violated, and if so we add it to the model.

5.5 Computational Results

The Branch and Cut algorithm was coded in C++ on an OSX platform, running on an Intel Core i7 3.4 GHz processor with 8 GB of RAM. For the model the Concert library

of IBM ILOG CPLEX 12.8 was used (default parameters and single thread mode).

5.5.1 Instances Generation

In the literature it does not exist benchmark instances for the 2ECMBV problem, hence we need to generate a set of instances to test the Branch and Cut algorithm. Our purpose is to generate a graph G which meets the following features: G is non-Hamiltonian and G is 3-connected. The request that G is non-Hamiltonian ensures that the optimal solution to the 2ECMBV problem is greater than zero, while if it is 3-connected it results that $ES(G) = \emptyset$ and $BR(G) = \emptyset$, thus the propositions proved in Section 5.3 hold. Let us note that in this case the preprocessing procedure is useless as in G there are no essential edges and no cut vertices. A family of 3-connected non-Hamiltonian graphs can be generated following the procedure described in Chapter 3. Let $G' = (V', E')$ be a complete graph such that $|V'| = n' \geq 4$. Let us consider an integer q such that $n' \geq 3q$, and let $W_1, ..., W_q \subseteq V'$ be q disjoint subsets of vertices such that $|W_i| = 3$, for any $i = 1, ..., q$. Given q disjoint sets of vertices $T_1, ..., T_q$, with $|T_i| \geq 3$, for any $i = 1, ..., q$, we build the graph $G(G', W_1, ..., W_q, T_1, ..., T_q) = (V, E)$, where $V = V' \cup T_1 \cup ... \cup T_q$ and $E = E' \cup \{\{u, v\} : u \in T_i, v \in W_i\}_{i=1,...,q}$. In Chapter 3 we showed that $G(G', W_1, ..., W_q, T_1, ..., T_q)$ is 3-connected and non-Hamiltonian.
We generated two sets of instances: *Small instances* with $n' \in \{15, 20, 25, 30\}$ and *Large instances* with $n' \in \{35, 40, 45, 50\}$. We denote by $\bar{n}$ the sum $\sum_{i=1}^{q} |T_i|$, that is the total number of vertices added to the complete graph G'. The integers $\bar{n}$ and q are chosen as follows: $\bar{n} \in \{\lfloor 0.5 \times n' \rfloor, \lfloor 0.8 \times n' \rfloor, n', \lfloor 1.5 \times n' \rfloor, \lfloor 2 \times n' \rfloor, \lfloor 2.5 \times n' \rfloor, \lfloor 3 \times n' \rfloor\}$ and $q \in \{\lfloor \bar{n}/5 \rfloor, \lfloor \bar{n}/3 \rfloor\}$. For each combination of n', $\bar{n}$ and q we have a different scenario and for each scenario we generated five instances, thus the total number of instances is 560. Therefore, each line in the tables represents a scenario composed on 5 instances with the same characteristics and different topologies and the results reported in each line are the average values on these 5 instances.

Tables 5.1 and 5.2 show the computational experiments with the Branch and Cut algorithm on the set of Small instances, while the computational results for the set of Large instances are reported in tables 5.3 and 5.4. The heading of the tables is the following. In the first five columns we reported the informations about the instances: the

number of vertices of the graph G' (n'), the cardinality of $T_1 \cup ... \cup T_q$ ($\bar{n}$), the integer q, the number of vertices of the instance (n) and the number of edges (m). In column Opt is reported the value of the optimal solution computed within the time limit, in column $Nodes$ there is the number of nodes of the Branch and Bound tree. The next three columns contain the informations about the added cuts: column $CutIneq$ reports the number of inequalities (5.2) and columns (5.7) and (5.8) contains the number of the corresponding added inequalities (5.7) and (5.8). Finally column $time$ reports the computational time in seconds. If in a scenario there are a instances that were not optimally solved within the time limit, (a) appears close to the solution value.

All the Small instances are solved to optimality within the time limit, and it requires at most $407,2$ seconds. The Large instances seem to be more difficult, indeed 91 out of 280 of them are not optimally solved within the time limit. Let us point out that for the instances not solved to optimality the percentage gap between the upper and the lower bound in most cases is lower than 20% and the optimal solutions could be reached by setting a time limit of 2 hours. The computational complexity is related to the parameter n', indeed as it increases the computational time increases too. For instance, let us compare the scenario having $n' = 25$, $q = 8$ and $n = 100$ with the scenario having $n' = 40$, $q = 8$ and $n = 100$: the first one is optimally solved in $1,3$ seconds, while in the second scenario two instances are not solved to optimality within the time limit. The largest number of violated cuts generated corresponds to the valid inequalities (5.7).

Finally, to evaluate the effect of the valid inequalities (5.8) used in our Branch and Cut approach, we performed an experiment consisting of comparing the complete algorithm described in Section 5.4 (*Complete B&C*) with another version (*Basic B&C*) where the violation checking test for the valid inequalities (5.8) is not performed. Figure 5.5 displays the percentage of optimally solved instances within the Cpu time. In more detail, the horizontal axis represents the Cpu time in seconds, while the vertical axis represents the percentage of optimally solved instances within a fixed Cpu time. Thus, it is easy to see that as faster is the curve growth, as better is the performance of the Branch and Cut. The blue curve is associated with the Complete B&C, while the green one with the Basic B&C. The separation of the valid inequalities (5.8) reduces the computational time. Indeed, by adding those cuts the Branch and Cut algorithm is able to solve to optimality 32 additional instances and takes substantially less time to

n'	n̄	q	n	m	Opt	Nodes	CutIneq	(5.7)	(5.8)	time
15	7	1	22	126	2	4,8	2,6	12	0	0,0
		2	22	126	2	4,2	4,4	12	0	0,0
	12	2	27	141	3,8	31,4	10	118,2	2,8	0,1
		4	27	141	3,8	40,4	12	156,8	4,2	0,1
	15	3	30	150	4,6	60,4	14,8	239,4	4	0,1
		5	30	150	4,8	75,8	15,4	288,4	6	0,1
	22	3	37	171	5,2	47	11,2	196,6	2,4	0,1
		5	37	171	5,8	45,6	10,8	272,8	3,8	0,1
	30	3	45	195	7,2	72,8	9,8	409,8	2,6	0,2
		5	45	195	5,8	25,2	9,4	166	2,2	0,1
	37	3	52	216	7,6	72,4	10,6	385,6	2	0,2
		5	52	216	7,2	178,8	10,6	507	1,6	0,6
	45	3	60	240	7,4	110	11,4	494,6	2	0,6
		5	60	240	7	45	7	294,6	6,2	0,2
20	10	2	30	220	3,4	20,4	10,2	59,4	0	0,0
		3	30	220	3	22,8	13	67,4	0	0,0
	16	3	36	238	4,6	133,2	19,8	481,6	9	0,7
		5	36	238	4,8	99,4	19,6	377,6	6,8	0,5
	20	4	40	250	6,6	450,2	20,2	763,2	10,2	4,8
		6	40	250	6,4	309,4	16,6	559,4	7,8	3,3
	30	4	50	280	7,8	199,2	18,2	657,6	1	1,1
		6	50	280	8	238,8	22,4	866,6	10	1,5
	40	4	60	310	7	77,4	17,8	375,2	1,8	0,5
		6	60	310	8	136,6	23,8	681,8	9,2	0,8
	50	4	70	340	9,4	184,4	24,8	931,6	1,8	1,9
		6	70	340	8,8	220	25,4	880	3,4	2,2
	60	4	80	370	7,6	101,8	19,8	531	1,8	1,4
		6	80	370	10,2	220,8	26,6	1076	3,4	1,7

Table 5.1: Computational results for instances with $n' = 15$ and $n' = 20$.

n'	n̄	q	n	m	Opt	Nodes	CutIneq	(5.7)	(5.8)	time
25	12	2	37	336	3,6	60	22,2	170,2	4,6	0,2
		4	37	336	3,8	71,8	29,2	231,8	6,8	0,2
	20	4	45	360	6,6	593,4	59,6	1211,8	5	11,6
		6	45	360	6,8	221	34,4	510,8	2,4	3,1
	25	5	50	375	7,8	1610,2	37,8	1913,4	10,8	71,9
		8	50	375	8	2343	36	2516,6	11,4	107,2
	37	5	62	411	8,8	304,6	37,8	1543,6	7,6	4,8
		8	62	411	8,6	734,4	39,8	1363,8	1,8	21,4
	50	5	75	450	11,2	507,6	53,4	1972,4	6	8,1
		8	75	450	8,8	145,4	39,6	779,4	5,8	1,2
	62	5	87	486	12,6	965,2	61,6	2389	2,2	16,3
		8	87	486	9,8	488,2	46,6	1560,2	4,2	15,8
	75	5	100	525	13,6	1389,4	76,8	2891,8	4	28,7
		8	100	525	10,2	115	31,2	807,2	1,6	1,3
30	15	3	45	480	4,8	80,2	28,4	332,6	5,4	0,4
		5	45	480	4,6	104	33,4	425,6	7,8	0,5
	24	4	54	507	7	162,4	48	791,4	17,6	2,2
		8	54	507	7,6	237,8	61,2	1075,2	26,4	3,3
	30	6	60	525	10	790,6	68,4	3148,2	50,2	52,9
		10	60	525	8,4	413	66,8	1567,2	15,6	16,0
	45	6	75	570	10,4	400,6	65,6	1839,8	4	19,1
		10	75	570	11	2940,8	65,8	4541,8	17,8	305,6
	60	6	90	615	13,4	3391,8	119,2	5000	5,8	191,8
		10	90	615	12,4	1592,8	108,6	3070,4	2,4	33,0
	75	6	105	660	12,6	764	109,4	2955	1,8	367,2
		10	105	660	14,4	1525,6	95,8	3907	11,4	65,5
	90	6	120	705	18,2	3229,2	202	6538,8	4,4	407,2
		10	120	705	11,2	170	61,4	1240,2	1	3,7

Table 5.2: Computational results for instances with $n' = 25$ and $n' = 30$.

n'	n̄	q	n	m	Opt	Nodes	CutIneq	(5.7)	(5.8)	time
35	17	3	52	646	5,4	190,4	69,2	714	1,2	3,4
		5	52	646	5,2	146,4	50,8	503,8	0,8	2,4
	28	5	63	679	8	5013,2	193,6	2448	8,2	668,0
		9	63	679	8,8	7548,6	269	3462,6	12	996,8
	35	7	70	700	$10,6^{(2)}$	3659,8	108,8	4514	13,4	1464,1
		11	70	700	$10,8^{(3)}$	4809,8	88	5688	14,4	2181,24
	52	7	87	751	12,6	4382	141	4473,2	7,2	758,5
		11	87	751	$12,2^{(2)}$	7394,8	160	6232	15,6	1568,5
	70	7	105	805	14	4244	219,8	4982	4,2	406,4
		11	105	805	$14,6^{(1)}$	10461,4	309,6	7733	13,8	1383,9
	87	7	122	856	16,2	8704,4	286	7238,8	5,4	794,3
		11	122	856	14,6	2327,8	166,6	3993	4,6	62,8
	105	7	140	910	16,2	14006,2	311	5425,6	0,6	699,7
		11	140	910	17,2	20252,6	352,8	7127,8	1,4	1209,8
40	20	4	60	840	6,8	539,6	136,6	1341,2	4,2	26,9
		6	60	840	7,2	412,2	97,2	952,6	3,2	18,0
	32	6	72	876	$8,8^{(2)}$	3869	277,8	3287,2	9	1446,9
		10	72	876	$9,8^{(3)}$	5706,8	389	4524,2	8,2	2167,7
	40	8	80	900	$12^{(2)}$	1281,8	66	4614,4	21,4	1447,8
		13	80	900	$12,4^{(3)}$	3149,8	227,4	7739	22,2	2326,7
	60	8	100	960	$13,8^{(2)}$	2913,8	181,8	7469,2	14,8	1485,1
		13	100	960	$13,8^{(2)}$	4387,8	217,8	10312,6	42,6	1934,6
	80	8	120	1020	$14,6^{(1)}$	2368,6	224,4	6259,4	1,6	841,3
		13	120	1020	$19^{(1)}$	14961,4	428,6	11592,6	2,4	1643,9
	100	8	140	1080	$20,4^{(3)}$	28239,6	280,8	13372,2	2,2	2775,1
		13	140	1080	$15,8^{(1)}$	14144,2	431,8	7027,6	7,6	852,7
	120	8	160	1140	15,8	6773,8	282,8	5434,2	3	602,6
		13	160	1140	$21^{(1)}$	28405,8	571	11412,6	2,8	1711,4

Table 5.3: Computational results for instances with $n' = 35$ and $n' = 40$.

n'	n̄	q	n	m	Opt	Nodes	CutIneq	(5.7)	(5.8)	time
45	22	4	67	1056	6,4	487,6	112,6	1298	9,4	22,6
		7	67	1056	6,4	671,8	132	1757,8	14	33,4
	36	7	81	1098	11	921,8	177,2	3864	50	113,5
		12	81	1098	10,4	949,4	174,2	3080,2	36	109,7
	45	9	90	1125	$14,2^{(3)}$	3580,8	211,8	8129,8	9,4	2577,8
		15	90	1125	$13,6^{(2)}$	2227,6	214,6	7074	11,2	1562,6
	67	9	112	1191	$15,4^{(2)}$	2804	231,2	9427,8	10,6	1663,2
		15	112	1191	$15,8^{(2)}$	4810,2	369	10965,2	6,4	1841,2
	90	9	135	1260	$19,2^{(2)}$	4572,2	283	9537,2	11,2	1470,8
		15	135	1260	$20,6^{(3)}$	6031,4	338	13823,4	9,8	2350,5
	112	9	157	1326	$19,2^{(2)}$	12685	300,8	9478,6	2,6	1530,6
		15	157	1326	$20,6^{(2)}$	9736,2	384,6	12299,4	18,4	2014,6
	135	9	180	1395	$22,6^{(3)}$	19345,2	519,8	12977,8	3,6	2351,5
		15	180	1395	$20^{(2)}$	8398,8	317,2	9071,8	2,2	1473,8
50	25	5	75	1300	7,2	803,6	175,8	2548,8	27,6	127,0
		8	75	1300	6,8	308	93,6	1281,8	17,2	12,4
	40	8	90	1345	$10,6^{(2)}$	4105,4	444,2	7227,8	57,6	1964,5
		13	90	1345	$11,4^{(2)}$	2401,8	221	6832,6	62,6	2103,3
	50	10	100	1375	$12,2^{(3)}$	3420	406,8	6942	70,4	2222,6
		16	100	1375	$15,8^{(5)}$	2653	190,6	11444,8	110,2	3610,6
	75	10	125	1450	$12,2^{(3)}$	2570,2	369,8	6083,8	43	2196,6
		16	125	1450	$15,8^{(4)}$	2524,8	197,2	11312,8	99,2	3125,4
	100	10	150	1525	$13,4^{(4)}$	2645,8	327	8974	74	2968,0
		16	150	1525	$14,6^{(3)}$	2863,2	240,6	9213,8	68	3024,1
	125	10	175	1600	$14,6^{(4)}$	2277,8	187,6	8581,6	63,4	2896,6
		16	175	1600	$13,4^{(3)}$	1734	221	7647,8	55,6	2207,5
	150	10	200	1675	$14,6^{(3)}$	1953	177,8	9201,8	76,8	2440,6
		16	200	1675	$13,4^{(3)}$	2207	253,8	7912,2	78,2	2250,7

Table 5.4: Computational results for instances with $n' = 45$ and $n' = 50$.

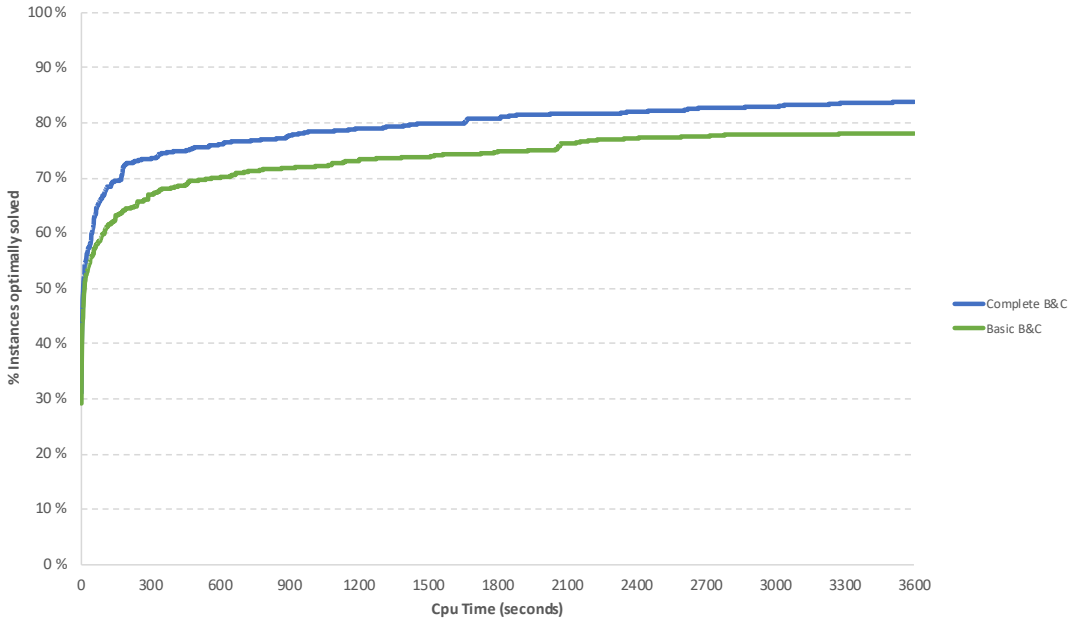

Figure 5.5: Percentage of optimally solved instances within the Cpu time for the Complete B&C (blue) and the Basic B&C (green).

solve the most of the other instances, causing a reduction in computational time of the $21,4\%$.

Conclusions

In this thesis we introduced two network design problems, the *Generalized Minimum Branch Vertices problem*, and the *2-Edge-Connected Minimum Branch Vertices problem*. These problems have the same objective function, that is the minimization of the number of branch vertices, which are vertices with degree greater than two in the network. The Generalized Minimum Branch Vertices problem is defined over a clustered undirected graph and a feasible solution is a tree spanning exactly one vertex for each cluster. While the 2-Edge-Connected Minimum Branch Vertices problem concerns the search for a spanning subgraph which provides protection against the failure of a single edge. The conclusions are presented with respect to each chapter.

Generation of 3-Connected non-Hamiltonian Graphs (Chapter 3)

In this chapter, we provided an overview of some conditions according to which a graph is Hamiltonian or not. Even if deciding whether an undirected graph G is Hamiltonian is a NP-complete problem, there are several results stating sufficient and necessary conditions for a graph to be Hamiltonian. Furthermore, we introduced a procedure for the generation of a class of 3-connected non-Hamiltonian graphs, which were used in the computational experiments of Chapter 5.

The Generalized Minimum Branch Vertices Problem (Chapter 4)

In this chapter, we provided a mathematical formulation for the Generalized Minimum Branch Vertices problem and we introduced several properties characterizing any feasible solution to the problem. Some properties were used to develop a preprocessing procedure with the aim of detecting useless vertices, namely vertices which do not belong to any feasible solution. We studied the facial structures of the polytope corresponding to the given model: we derived its dimension, several facet-defining inequalities and a new family of valid inequalities. To solve the Generalized Minimum Branch Vertices problem we designed a Branch and Cut algorithm. The computational results showed the effectiveness of the preprocessing procedure, as it allowed us to optimally solve the $6,4\%$ of additional instances and to reduce the overall computational time. Furthermore, the Branch and Cut approach was able to solve almost the 80% of the instances in 7 minutes. As future works we could focus on finding new valid inequalities to integrate into the Branch and Cut algorithm, as well as on the development of heuristic approaches for the problem.

The 2-Edge-Connected Minimum Branch Vertices Problem (Chapter 5)

In this chapter, we studied the 2-Edge-Connected Minimum Branch Vertices problem. We proposed an integer linear programming formulation for the problem and investigated some properties useful to characterize a feasible solution. We studied the polyhedron associated with the proposed formulation obtaining its dimension and some facet results. Furthermore, we introduced a new class of valid inequalities and studied the relative separation problem. A Branch and Cut approach has been developed to solve the 2-Edge-Connected Minimum Branch Vertices problem. To test the algorithm we used a set of 3-connected non-Hamiltonian instances, generated as described in Chapter 3. The computational results showed that the introduction of the new valid

inequalities allowed to optimally solve the $5,7\%$ of additional instances with a reduction of the $21,4\%$ in computational time. Future research can be conducted to improve the Branch and Cut algorithm by introducing new effective classes of valid inequalities. However, given the hardness of this problem, an exact approach is not suitable for large instances, therefore a further research direction could be the study of heuristic algorithms to solve it.

Appendix A

Tables A.1-A.5 report the detailed computational results of the Branch and Cut algorithm for the GMBV problem, described in Chapter 4. The heading of the tables is the following. In the first three columns we reported the informations about the instances: the number of clusters (k), the number of vertices (n), and the number of edges (m). In column *%RV* is reported the percentage removed vertices, in column *Opt* is reported the value of the optimal solution computed within the time limit, in column *Nodes* there is the average of number of nodes of the Branch and Bound tree. In the next three columns there are the informations about the cut added: column $GSEC(4.10)$ reports the average of the number of inequalities (4.10), column $GSEC(4.4)$ reports the average of the number of inequalities (4.4), and column (4.13) contains the average of the number of constraints (4.13) added. Finally column *time* reports the average of the computational time in seconds. When a " $-$ " is reported, no feasible solution has been found.

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
12	36	38	13,9	1	2	31	3	8	0,1
			36,1	2	2	9	0	12	0,1
			8,3	1	5	21	21	12	0,1
			25,0	0	0	8	0	0	0,0
			13,9	1	2	14	5	14	0,1
		44	5,6	0	1	40	0	0	0,1
			5,6	1	2	35	59	35	0,1
			13,9	1	5	30	16	27	0,1
			13,9	1	4	18	26	17	0,1
			8,3	0	0	18	0	0	0,1
		50	13,9	1	3	19	7	13	0,1
			0,0	0	0	55	0	0	0,1
			0,0	0	0	21	7	0	0,1
			0,0	0	0	19	7	0	0,1
			2,8	0	0	28	9	0	0,1
12	48	50	10,4	0	0	14	0	0	0,1
			29,2	1	3	14	8	18	0,1
			6,3	1	1	42	23	28	0,1
			22,9	1	6	29	9	16	0,1
			22,9	1	4	21	22	31	0,1
		57	16,7	1	3	33	42	29	0,1
			12,5	0	0	26	0	0	0,1
			8,3	0	0	41	0	0	0,1
			8,3	1	7	48	46	46	0,1
			10,4	0	0	32	7	0	0,1
		64	6,3	0	0	57	0	0	0,1
			6,3	0	0	63	11	0	0,2
			8,3	1	3	47	65	11	0,2
			8,3	0	0	36	4	0	0,1
			25,0	2	9	38	49	29	0,1
12	72	75	41,7	1	2	32	8	16	0,3
			29,2	0	0	13	0	0	0,2
			8,3	1	7	55	32	66	0,2
			27,8	1	3	41	23	8	0,2
			19,4	1	4	48	24	6	0,2
		83	4,2	0	0	63	7	0	0,2
			12,5	0	0	82	32	0	0,3
			27,8	0	0	52	0	0	0,2
			9,7	0	0	66	10	0	0,2
			22,2	0	3	71	42	0	0,4
		92	2,8	0	0	87	38	0	0,2
			4,2	0	0	49	3	0	0,3
			6,9	0	0	77	3	0	0,3
			2,8	0	0	81	6	0	0,2
			8,3	0	0	58	0	0	0,3
12	96	99	11,5	0	0	64	0	0	0,2
			14,6	0	0	57	0	0	0,2
			19,8	1	6	68	63	58	0,3
			25,0	0	0	42	0	0	0,2
			21,9	1	3	71	77	27	0,3
		109	22,9	0	0	70	9	0	0,3
			24,0	0	0	80	0	0	0,4
			20,8	0	0	94	22	0	0,5
			12,5	0	0	99	14	0	0,5
			9,4	0	0	112	25	0	0,4
		119	9,4	0	0	84	5	0	0,6
			9,4	0	0	91	0	0	0,5
			4,2	0	0	114	0	0	0,4
			5,2	0	0	133	70	0	0,5
			3,1	0	0	35	0	0	0,4
12	120	124	15,8	1	5	94	28	29	0,4
			16,7	0	1	84	7	0	0,3
			29,2	1	5	84	37	22	0,5
			17,5	0	0	52	0	0	0,4
			30,0	1	5	65	77	31	0,4
		135	12,5	1	1	122	156	36	0,8
			13,3	0	0	113	4	0	0,5
			17,5	0	0	92	4	0	0,6
			18,3	0	0	86	0	0	0,6
			10,8	0	0	101	0	0	0,5
		146	5,0	0	0	151	11	0	0,8
			10,0	0	0	131	19	0	0,9
			8,3	0	0	123	0	0	0,9
			8,3	0	0	106	0	0	0,5
			11,7	0	0	169	16	0	1,1

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
16	48	50	31,3	2	2	13	23	15	0,1
			29,2	1	2	7	0	6	0,1
			4,2	2	3	28	13	36	0,1
			16,7	2	2	14	0	21	0,0
			16,7	2	3	27	24	16	0,1
		57	0,0	1	9	46	65	33	0,1
			8,3	1	6	39	35	16	0,1
			14,6	1	3	29	23	56	0,1
			2,1	1	6	51	110	15	0,2
			6,3	0	0	13	6	0	0,1
		64	2,1	0	0	38	0	0	0,1
			0,0	0	0	35	0	0	0,1
			2,1	1	3	56	43	28	0,2
			0,0	0	0	30	7	0	0,1
			2,1	0	0	59	6	0	0,1
16	64	67	7,8	0	0	18	0	0	0,1
			26,6	2	3	33	0	37	0,1
			20,3	1	2	41	33	21	0,1
			20,3	1	4	39	42	46	0,1
			23,4	2	3	28	9	21	0,1
		75	1,6	0	0	46	0	0	0,1
			0,0	1	3	51	27	62	0,2
			1,6	1	7	70	57	46	0,2
			9,4	1	6	68	145	43	0,3
			3,1	1	15	83	80	11	0,3
		83	15,6	1	4	63	55	43	0,3
			4,7	0	0	99	0	0	0,2
			3,1	1	6	75	157	41	0,2
			0,0	1	6	65	101	62	0,3
			15,6	1	4	45	97	43	0,3
16	96	99	22,9	1	4	56	39	27	0,3
			17,7	1	9	77	49	55	0,3
			27,1	1	4	57	20	32	0,3
			26,0	2	10	47	8	32	0,3
			13,5	1	6	72	67	62	0,3
		109	4,2	1	10	101	198	81	0,6
			11,5	1	4	109	132	36	0,7
			10,4	1	10	88	205	40	0,7
			4,2	0	0	76	0	0	0,3
			2,1	0	0	80	11	0	0,3
		119	4,2	0	0	90	8	0	0,3
			6,3	0	0	116	3	0	0,5
			0,0	0	0	137	36	0	0,7
			8,3	0	0	108	0	0	0,8
			0,0	0	0	88	0	0	0,5
16	128	132	34,4	1	13	64	10	71	0,4
			17,2	1	13	92	139	74	0,8
			15,6	1	4	94	55	79	0,6
			18,8	1	16	94	142	51	0,9
			20,3	1	3	78	23	14	0,3
		143	10,2	1	10	124	97	48	1,3
			5,5	0	0	100	22	0	0,5
			10,9	1	7	115	181	32	1,4
			13,3	1	5	115	79	32	0,8
			4,7	1	23	165	249	31	1,6
		155	3,1	0	0	138	0	0	0,8
			0,8	0	0	176	15	0	1,2
			3,9	0	0	78	0	0	0,7
			7,8	0	0	165	72	0	1,2
			0,8	0	0	72	14	0	0,8
16	160	165	24,4	1	10	112	104	95	1,1
			8,1	1	9	128	58	132	0,9
			26,3	1	9	104	100	72	1,0
			34,4	1	5	82	53	28	0,8
			28,8	1	4	96	36	49	0,8
		177	7,5	0	0	95	0	0	1,3
			10,6	0	0	136	13	0	1,0
			5,0	0	0	99	0	0	0,8
			12,5	0	0	163	3	0	1,2
			7,5	0	0	151	30	0	1,1
		190	7,5	0	0	169	3	0	1,5
			8,1	0	0	188	0	0	1,5
			8,8	0	0	201	27	0	1,6
			5,0	0	5	220	123	0	1,7
			5,0	0	0	224	115	0	2,0

Table A.1: Computational results for instances with $k = 12$ and $k = 16$.

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
20	60	62	13,3	2	6	32	56	48	0,1
			30,0	3	4	21	37	33	0,1
			16,7	3	5	30	34	40	0,1
			10,0	2	3	34	53	32	0,1
			8,3	2	5	35	31	60	0,1
		70	8,3	1	3	48	23	39	0,1
			10,0	1	3	31	62	19	0,1
			31,7	3	3	41	42	24	0,1
			13,3	2	3	32	88	27	0,1
			6,7	2	4	49	70	35	0,2
		78	0,0	1	26	80	237	93	0,4
			0,0	1	40	101	229	30	0,4
			3,3	1	6	61	127	67	0,3
			0,0	1	12	73	194	48	0,4
			13,3	1	3	52	70	28	0,2
20	80	83	13,8	1	9	61	71	38	0,2
			26,3	2	5	29	28	54	0,2
			21,3	3	3	42	40	46	0,1
			11,3	3	7	55	24	50	0,2
			26,3	2	4	36	38	43	0,2
		92	3,8	1	7	76	92	84	0,4
			2,5	0	0	55	0	0	0,3
			7,5	1	3	68	134	34	0,3
			3,8	1	7	65	71	53	0,4
			2,5	1	8	83	71	39	0,5
		101	0,0	1	11	100	248	59	0,6
			11,3	1	4	99	188	70	0,5
			0,0	1	8	95	241	38	0,6
			16,3	1	3	62	170	24	0,7
			11,3	1	3	77	58	30	0,3
20	120	124	15,8	2	18	91	129	82	0,6
			19,2	1	3	77	47	38	0,5
			12,5	1	7	87	142	71	0,6
			26,7	1	15	82	89	65	0,6
			5,8	1	5	91	36	46	0,5
		135	3,3	1	6	127	105	50	0,9
			4,2	1	33	148	315	73	1,5
			7,5	1	6	125	231	114	1,1
			11,7	1	10	120	225	60	1,6
			0,8	0	0	123	0	0	0,6
		146	6,7	1	22	156	275	65	1,7
			0,0	0	0	163	5	0	0,8
			3,3	1	27	162	484	64	2,5
			5,8	0	0	103	0	0	0,7
			3,3	0	0	155	11	0	0,7
20	160	165	30,0	2	4	88	59	50	1,0
			13,8	2	11	114	57	72	0,9
			18,1	1	6	113	86	60	0,9
			19,4	1	3	95	46	66	0,8
			19,4	1	4	122	76	52	0,9
		177	5,0	1	26	171	196	76	2,2
			10,6	1	9	145	225	62	2,1
			9,4	1	14	177	509	87	2,9
			13,8	0	0	117	0	0	1,5
			14,4	1	40	151	339	88	2,7
		190	3,1	0	0	180	20	0	1,9
			1,9	0	2	191	50	0	2,2
			5,6	0	0	193	27	0	1,5
			3,8	1	19	226	618	33	6,7
			6,9	1	53	225	735	96	5,2
20	200	206	7,5	1	33	175	266	154	2,8
			27,5	1	44	152	310	70	2,6
			12,0	1	3	139	165	39	1,8
			17,0	1	12	133	139	92	1,8
			17,5	1	23	156	292	184	2,2
		220	7,0	1	19	206	272	80	3,8
			16,0	1	17	176	432	153	3,6
			10,0	1	34	212	269	77	4,3
			13,5	1	17	212	339	63	3,7
			8,5	1	38	226	550	85	6,5
		234	3,5	0	52	302	899	50	11,1
			1,5	1	88	276	830	114	13,1
			2,0	1	60	293	902	136	25,6
			6,5	1	11	305	119	131	20,9
			1,0	1	89	290	954	144	10,7

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
30	90	93	36,7	5	3	20	0	22	0,1
			14,4	5	4	40	79	51	0,2
			15,6	4	17	55	163	52	0,3
			20,0	3	12	36	141	72	0,2
			16,7	4	3	54	58	38	0,2
		103	18,9	3	10	54	161	99	0,3
			0,0	2	28	106	499	103	0,8
			10,0	2	3	60	62	36	0,3
			11,1	3	16	64	383	67	0,4
			14,4	2	28	76	293	55	0,6
		112	15,6	2	7	64	284	96	0,5
			8,9	2	6	69	204	71	0,4
			0,0	2	10	83	383	47	0,7
			1,1	1	3	83	90	24	0,4
			11,1	1	66	93	854	169	0,9
30	120	124	23,3	3	3	81	59	47	0,4
			15,8	4	12	60	137	99	0,5
			20,0	4	7	59	170	77	0,5
			15,0	4	7	88	47	86	0,5
			19,2	2	19	63	175	88	0,6
		135	12,5	3	44	116	365	145	1,3
			12,5	2	3	86	150	72	0,7
			25,0	2	3	64	221	34	0,9
			18,3	4	19	107	247	103	1,3
			11,7	3	5	99	104	70	0,9
		146	0,0	1	40	174	535	91	1,8
			4,2	1	23	153	817	94	2,1
			0,0	1	106	148	1448	202	2,9
			0,0	1	4	117	320	133	1,3
			0,0	1	106	160	1113	223	3,0
30	180	185	12,2	3	30	141	482	136	2,1
			7,2	2	3	134	104	102	1,3
			6,7	2	8	150	111	112	1,6
			5,0	2	28	138	435	185	1,9
			9,4	2	4	128	171	87	1,7
		199	2,8	2	98	202	1288	199	6,5
			3,9	2	34	171	613	194	3,9
			2,2	2	75	176	1052	171	6,7
			1,1	1	28	184	545	183	3,6
			3,3	2	30	178	888	133	4,3
		212	0,6	1	177	242	1792	239	11,5
			2,2	1	70	230	2156	201	14,1
			0,6	1	124	275	2316	214	14,1
			3,3	2	81	209	1774	285	10,5
			2,2	1	356	251	2553	189	31,9
30	240	246	10,4	2	11	160	426	197	3,7
			10,4	2	10	160	351	95	3,6
			13,3	3	34	185	413	204	4,5
			13,8	2	5	177	124	114	2,9
			14,6	2	32	185	429	165	4,1
		262	4,2	1	24	260	206	109	5,6
			8,3	2	37	254	1212	173	9,8
			10,0	2	119	255	1531	251	17,8
			2,9	1	31	260	1133	166	7,9
			4,2	1	95	258	1241	341	11,7
		277	1,3	1	82	316	1434	183	14,2
			2,5	1	73	295	1607	174	15,6
			5,0	1	251	329	2794	247	32,8
			2,5	2	409	322	4510	418	89,8
			2,5	1	642	359	4284	321	65,6
30	300	307	15,7	2	11	212	499	126	7,0
			20,3	2	24	187	333	163	6,7
			7,7	1	73	254	843	227	9,1
			10,3	2	9	214	447	109	5,5
			15,0	2	11	222	621	157	6,4
		324	6,3	1	40	276	1187	187	15,8
			4,3	1	303	327	2633	407	42,6
			6,3	1	241	311	3058	362	66,2
			1,3	1	72	336	1187	239	15,8
			4,3	1	8	277	576	89	11,4
		342	2,7	2	100	411	7042	343	1396,3
			1,0	0	0	354	229	0	8,2
			2,7	1	245	418	4288	413	128,1
			1,7	1	220	398	3748	404	89,7
			2,7	1	166	394	1660	154	51,9

Table A.2: Computational results for instances with $k = 20$ and $k = 30$.

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
40	120	124	19,2	6	3	49	123	70	0,4
			15,0	6	5	58	152	56	0,4
			23,3	6	4	48	141	81	0,3
			8,3	4	9	77	185	80	0,5
			10,8	6	9	60	121	98	0,5
		135	8,3	5	17	78	327	122	0,6
			2,5	3	7	98	516	85	0,9
			13,3	5	35	90	464	100	1,0
			5,8	4	23	109	1006	137	1,3
			2,5	4	43	123	884	135	1,6
		146	5,0	2	8	112	528	133	1,2
			5,8	3	53	129	1486	133	2,6
			3,3	2	57	118	1459	158	2,4
			0,8	2	30	111	666	140	1,6
			2,5	2	40	147	978	151	1,9
40	160	165	8,8	5	14	121	236	111	1,2
			14,4	5	7	86	140	98	0,9
			16,9	5	6	82	125	99	0,8
			10,6	4	6	83	208	133	0,9
			21,3	5	25	98	175	102	1,0
		177	4,4	3	37	165	1093	186	3,9
			4,4	3	134	183	1647	313	6,0
			6,9	4	28	140	530	136	2,4
			5,0	3	18	132	476	133	2,1
			1,9	2	57	160	1905	183	4,8
		190	1,9	2	51	183	1978	220	5,2
			3,1	2	59	181	1748	194	4,7
			0,0	2	130	219	3229	246	12,5
			0,6	2	106	204	2340	221	7,8
			3,1	2	55	211	2069	218	6,4
40	240	246	7,9	4	42	194	802	215	4,7
			6,7	3	18	172	435	184	3,8
			10,8	4	29	182	359	165	3,5
			11,3	5	34	181	532	189	4,5
			16,7	4	8	147	151	183	3,0
		262	1,7	3	122	276	3196	228	23,3
			5,0	3	203	269	2318	398	19,4
			2,1	3	102	243	2275	243	13,7
			4,6	2	48	244	1869	239	10,9
			3,8	2	45	256	2032	204	14,7
		277	0,0	2	139	296	2591	296	17,3
			1,7	2	103	294	3893	305	32,9
			0,4	2	333	337	6204	388	356,8
			0,0	1	42	253	1324	171	10,7
			0,8	2	199	312	4529	310	61,1
40	320	327	16,3	3	102	236	1312	290	12,1
			18,1	3	14	223	723	198	8,2
			14,7	4	347	259	903	381	13,5
			10,6	2	45	245	1250	234	9,6
			8,8	3	170	279	2143	308	18,1
		345	5,6	2	111	324	2344	307	34,8
			1,3	2	292	344	4480	427	62,0
			1,9	2	229	366	5879	341	226,6
			0,9	2	155	328	2782	415	41,4
			1,3	2	257	343	3746	377	54,9
		363	1,9	2	264	369	6275	344	595,2
			1,6	2	306	424	8159	360	201,1
			1,3	1	288	424	7434	340	138,2
			0,9	1	250	379	5984	391	136,5
			1,9	1	445	413	7916	558	190,7
40	400	409	6,5	2	126	362	1731	428	30,7
			6,8	2	51	322	1242	325	21,1
			8,8	2	46	310	1593	319	19,7
			10,0	3	209	363	2280	543	38,3
			7,8	2	63	330	1396	401	26,7
		429	2,5	2	362	417	5151	573	220,1
			3,8	1	27	386	1565	203	35,1
			7,3	2	384	431	7292	339	212,1
			3,3	2	127	364	4934	356	101,7
			3,3	2	229	432	4997	353	107,1
		449	5,0	1	489	530	8613	509	298,1
			1,3	2	1361	561	16320	801	3610,6
			2,5	1	259	452	6573	344	3103,1
			2,5	2	656	502	11009	517	2726,3
			3,8	1	532	504	10263	607	3610,6

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
50	150	155	7,3	7	49	118	1247	160	1,8
			9,3	6	19	103	710	100	1,1
			17,3	8	13	63	329	97	0,7
			24,7	7	10	42	158	60	0,5
			12,7	7	8	80	324	108	0,8
		167	4,7	5	73	134	2434	218	4,7
			11,3	7	33	128	549	160	1,6
			5,3	4	12	88	647	114	0,9
			6,7	4	31	124	1362	170	2,5
			6,7	6	73	144	1540	145	3,1
		179	1,3	3	47	157	2017	221	6,2
			0,0	3	112	186	3191	224	6,9
			4,7	3	43	137	1209	203	2,7
			7,3	4	31	139	1632	131	2,8
			4,7	3	39	165	1803	105	3,9
50	200	206	7,0	6	67	159	973	258	3,5
			20,0	5	4	104	179	151	1,7
			13,0	5	41	118	758	146	2,2
			8,0	6	46	144	491	173	2,5
			11,5	5	3	118	443	124	1,6
		220	5,5	4	43	178	1697	213	6,1
			14,5	5	5	138	365	116	2,1
			4,0	4	63	200	2143	235	7,2
			3,0	3	20	170	651	129	3,7
			2,5	3	203	224	4571	251	19,9
		234	1,5	4	60	233	2643	195	17,0
			0,5	4	308	282	6146	290	59,8
			0,5	2	63	222	3379	274	14,4
			1,0	3	68	251	3533	229	18,2
			1,0	4	91	230	3449	303	19,1
50	300	307	11,7	4	6	202	589	182	6,9
			9,7	5	95	241	661	259	10,4
			6,7	4	84	238	1863	349	14,7
			7,3	5	89	213	2129	271	12,8
			3,3	5	199	280	2010	423	18,2
		324	2,3	4	285	362	6674	449	120,6
			5,0	2	79	275	2551	303	20,4
			9,7	4	131	280	3326	399	30,0
			2,3	3	65	278	2184	295	22,6
			1,7	3	280	328	6124	499	101,6
		342	3,0	2	408	383	12114	484	233,7
			1,7	3	364	393	11868	381	387,0
			0,3	3	870	388	15449	381	864,9
			0,7	2	291	348	10898	604	237,9
			0,0	3	792	402	21092	621	1011,3
	400	409	5,8	4	118	327	2795	352	35,4
			7,0	4	254	329	2990	516	44,0
			5,3	4	388	340	3367	489	68,0
			9,8	4	104	321	2590	308	36,2
			9,5	5	230	337	2867	634	51,5
		429	4,0	2	91	401	4178	378	69,8
			3,8	3	265	423	7087	394	208,6
			1,0	2	230	427	10335	576	220,0
			3,3	3	587	456	9808	453	468,6
			5,0	3	155	402	6037	447	152,1
		449	0,5	2	171	459	6327	407	165,4
			0,8	3	446	436	14196	426	1010,8
			0,3	3	1016	567	30758	595	3610,6
			1,8	2	310	480	9042	484	932,5
			0,8	2	1367	544	21353	644	3610,6
50	500	510	4,4	4	166	454	2908	505	67,3
			7,8	3	113	410	3445	418	113,7
			6,0	3	125	448	4691	467	90,3
			12,8	3	147	399	3195	478	54,0
			5,6	4	199	450	4546	602	96,5
		532	2,2	2	250	490	10003	677	543,7
			1,2	3	1180	515	16631	803	2328,3
			2,4	2	310	517	8218	749	340,0
			3,0	1	404	510	14165	733	1411,8
			4,0	3	1220	484	15104	648	1327,2
		554	2,2	2	717	626	25949	631	3610,6
			3,2	2	580	617	20775	776	3610,6
			2,4	3	737	621	37630	986	3610,6
			0,6	4	521	660	19049	641	3610,6
			0,8	3	900	612	26666	793	3610,6

Table A.3: Computational results for instances with $k = 40$ and $k = 50$.

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
60	180	185	7,8	7	4	99	384	137	1,1
			20,6	10	11	83	223	104	0,8
			20,0	9	4	79	223	89	0,8
			9,4	8	46	103	576	172	2,0
			17,2	7	0	61	46	117	0,6
		199	7,2	6	38	131	1563	158	3,7
			1,7	4	53	128	3717	218	6,9
			17,8	7	4	77	906	117	1,4
			1,1	4	36	169	2094	187	5,8
			9,4	7	52	120	1368	239	3,5
		212	6,1	4	68	169	4687	214	8,3
			2,2	3	41	178	2698	169	7,8
			2,2	5	74	202	3681	232	14,1
			0,0	3	178	223	7308	286	25,9
			2,8	3	88	188	3128	262	7,4
60	240	246	18,8	7	50	143	905	179	3,0
			4,2	6	25	149	622	163	3,3
			8,8	7	38	185	1068	215	4,8
			8,3	7	47	180	830	194	4,4
			5,8	7	100	193	2034	313	7,3
		262	5,4	6	178	235	3413	326	24,7
			3,8	6	410	253	7071	292	56,6
			4,2	5	164	246	4754	299	25,4
			0,8	5	108	259	5699	333	34,9
			4,6	3	115	218	3239	250	15,3
		277	5,4	3	88	227	4961	288	21,0
			1,3	4	256	284	10206	347	107,7
			3,8	3	241	296	10508	414	175,4
			1,7	4	341	297	11471	343	116,6
			0,4	2	155	277	7914	426	54,4
60	360	368	10,3	6	100	273	1049	392	13,5
			8,1	5	175	310	4236	452	39,4
			8,1	5	36	269	2412	239	23,3
			6,1	5	13	272	1292	276	12,4
			8,9	6	443	295	3044	454	49,3
		387	2,5	4	661	437	19451	641	755,8
			1,9	5	430	403	11311	561	461,1
			1,9	3	361	379	11914	463	398,4
			1,9	3	265	348	10595	442	236,3
			1,1	3	975	392	14557	574	911,7
		406	0,8	4	1017	479	34247	670	3431,0
			0,3	3	522	480	32976	700	1398,9
			4,2	3	317	470	19646	605	639,9
			0,6	3	582	486	28645	714	1237,4
			1,4	3	292	442	11711	403	659,3
60	480	489	4,6	4	71	390	3159	376	53,9
			5,8	4	88	376	3051	571	57,7
			17,3	5	114	353	2987	494	53,4
			3,3	5	115	389	3234	556	66,4
			10,2	6	208	388	3641	541	83,8
		511	2,9	5	1155	547	29134	839	3610,6
			2,5	3	499	533	19571	850	3404,4
			1,3	4	654	511	22051	739	1134,3
			1,9	4	1558	529	30260	937	3610,6
			1,0	3	378	541	25634	759	1254,5
		533	1,0	5	463	541	29731	826	3610,6
			2,5	4	1036	644	42034	1082	3610,6
			0,2	3	630	589	42891	768	3610,6
			0,2	4	705	613	39095	906	3610,6
			0,8	3	578	606	40052	626	3610,6
60	600	611	6,0	4	264	533	8319	634	271,9
			6,3	4	163	539	6521	610	207,9
			6,3	6	1691	570	10755	925	1736,3
			5,8	5	192	517	5587	554	180,0
			7,5	4	322	517	6626	655	266,2
		635	0,7	4	820	651	26194	834	3610,6
			3,0	5	562	652	29810	864	3610,6
			1,8	4	916	662	32083	893	3610,6
			2,5	5	691	679	29484	883	3610,6
			4,2	4	981	656	29787	1069	3610,6
		660	1,8	5	340	701	30877	961	3610,6
			1,7	4	528	701	29013	865	3610,6
			0,2	8	764	740	29240	963	3610,6
			0,3	5	612	768	35164	905	3610,6
			1,2	5	677	740	36884	827	3610,6

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
70	210	216	15,7	9	15	88	576	157	1,9
			13,3	9	17	115	698	148	1,8
			10,0	8	50	113	1359	192	2,6
			16,2	10	20	106	526	149	1,7
			17,6	10	31	86	1240	146	1,8
		230	7,1	9	32	167	1876	210	6,3
			4,8	6	27	164	3283	225	7,6
			19,0	10	8	107	846	141	2,1
			8,6	8	41	140	1671	155	4,3
			8,6	7	67	176	1637	221	5,4
		245	5,7	6	64	186	4844	313	15,5
			6,2	4	103	191	8710	257	24,1
			9,5	8	124	193	6576	235	20,8
			3,3	4	67	214	4406	325	13,0
			6,2	7	136	214	5655	266	24,2
70	280	287	12,5	8	266	171	1626	307	11,3
			12,9	7	18	157	1013	224	4,2
			13,2	9	46	200	1579	269	6,9
			6,4	8	34	208	999	284	6,6
			18,9	9	4	141	472	147	3,1
		304	5,7	7	45	259	4670	336	23,5
			5,7	7	211	278	7915	327	56,6
			10,7	7	102	223	3735	313	17,2
			5,4	5	279	282	12281	427	130,1
			7,5	6	253	264	10799	417	103,0
		320	3,9	5	277	318	16685	406	238,4
			1,1	5	125	332	10291	357	82,9
			1,8	5	504	321	16356	400	282,6
			1,8	4	1271	367	41822	798	1993,0
			2,5	4	395	355	21568	409	349,9
70	420	429	2,9	5	393	355	6176	774	137,6
			4,5	6	220	334	3269	383	53,7
			5,2	6	44	339	2417	367	31,7
			6,9	7	770	348	6337	587	189,6
			11,4	7	797	356	2641	514	82,9
		449	1,4	6	1057	469	40694	821	3610,6
			4,5	5	600	446	17877	708	851,0
			0,5	5	361	442	18567	754	684,8
			1,9	5	1528	471	30734	838	2416,8
			3,1	5	275	397	13938	530	418,1
		470	0,2	7	908	573	53119	665	3610,6
			0,5	5	912	535	45673	724	3610,6
			1,4	3	315	487	23914	573	3610,6
			0,7	3	546	503	34593	798	2049,8
			0,7	5	654	562	51290	850	3610,6
70	560	570	5,9	5	881	511	17623	787	1437,0
			8,8	5	273	473	6244	539	203,2
			4,1	5	303	495	8863	806	301,8
			4,8	5	342	495	6015	662	186,1
			3,4	6	271	505	7518	696	284,1
		594	2,1	7	703	659	38444	1114	3610,6
			1,3	5	704	554	36598	722	3610,6
			0,9	5	700	584	35608	771	3610,6
			1,8	4	956	600	38917	1174	3610,6
			1,4	5	688	629	35562	1111	3610,6
		618	0,4	6	353	669	40980	768	3610,6
			0,4	6	600	730	40373	1131	3610,6
			0,7	4	450	706	38045	948	3610,6
			0,4	6	810	717	35625	804	3610,6
			0,7	5	527	683	41067	963	3610,6
70	700	712	6,7	4	3020	554	12027	946	3205,6
			5,9	4	260	626	7298	835	312,7
			2,1	5	674	642	13633	749	833,0
			7,6	5	229	574	6254	576	292,5
			3,7	5	622	625	12161	738	913,3
		738	2,1	5	588	679	38454	1124	3610,6
			1,6	4	487	703	27806	1164	3610,6
			1,0	6	762	766	31543	958	3610,6
			1,6	3	654	784	38278	1099	3610,6
			0,4	8	569	804	35782	1163	3610,6
		765	1,3	4	225	763	33933	824	3610,6
			0,6	10	157	749	31815	567	3610,6
			1,4	9	142	745	28829	675	3610,6
			0,7	7	237	820	27736	643	3610,6
			1,1	6	128	717	30658	559	3610,6

Table A.4: Computational results for instances with $k = 60$ and $k = 70$.

k	n	m	%RV	Opt	Nodes	GSEC(4.10)	GSEC(4.4)	(4.13)	time
80	240	246	14,6	11	9	105	359	133	1,5
			15,8	11	5	101	447	109	1,7
			25,8	14	6	76	506	108	1,3
			16,7	12	12	103	951	171	1,9
			11,7	10	10	132	1056	184	2,8
		262	7,1	8	96	206	5430	256	21,3
			5,8	8	75	195	4230	252	15,0
			3,8	7	358	217	10573	420	60,0
			2,5	7	52	211	3710	318	14,1
			5,8	9	114	220	7373	350	31,6
		277	5,4	7	431	287	16261	387	152,8
			2,5	8	207	264	14048	393	134,5
			5,0	5	166	242	12668	351	62,5
			4,6	6	154	246	14211	369	74,2
			12,9	9	188	200	13247	318	51,7
80	320	327	19,7	9	68	160	2021	272	6,8
			9,1	9	15	209	540	247	5,9
			18,1	10	17	169	999	170	6,5
			5,3	8	154	211	4690	399	23,0
			6,9	10	113	219	1432	326	12,9
		345	3,4	8	245	338	11587	383	130,0
			8,4	8	72	276	6060	306	41,9
			8,4	8	97	271	6094	397	40,6
			10,3	8	335	316	10438	379	152,6
			5,0	7	224	308	11488	510	146,2
		363	0,6	6	114	371	11902	375	127,6
			1,6	7	835	440	68133	668	3610,6
			2,2	6	180	352	17764	348	223,0
			2,5	5	201	364	15469	487	207,0
			4,1	4	146	309	13953	310	139,4
	480	489	3,5	7	379	392	9438	585	201,3
			21,9	8	172	309	3765	460	43,0
			5,6	8	641	416	11572	648	413,6
			12,7	8	483	375	9993	657	290,2
			7,3	7	870	404	10524	711	452,5
		511	2,5	5	556	472	25564	610	1570,1
			1,7	6	1032	543	43908	809	3610,6
			0,2	7	744	514	42920	773	3610,6
			3,3	7	842	529	46750	791	3610,6
			1,0	6	950	523	38284	932	3610,6
		533	1,0	5	601	538	41747	773	3610,6
			0,8	5	458	562	52189	776	3610,6
			1,9	5	827	564	51096	968	3610,6
			2,7	5	407	543	51828	747	3610,6
			0,2	6	480	616	53181	653	3610,6
80	640	651	1,9	6	1347	596	27308	799	3610,6
			5,8	7	481	556	15477	910	977,8
			3,4	7	803	585	10982	733	1154,5
			4,2	6	429	568	14852	838	725,2
			6,6	5	267	556	8953	735	396,3
		676	2,3	7	634	716	39936	1013	3610,6
			1,1	5	508	706	41310	962	3610,6
			1,7	7	335	669	40064	833	3610,6
			3,9	6	517	683	35386	907	3610,6
			1,9	6	448	679	39304	1068	3610,6
		702	0,6	7	258	753	39520	1120	3610,6
			0,8	7	328	779	40107	852	3610,6
			0,8	7	228	701	42868	758	3610,6
			1,4	7	486	784	42085	954	3610,6
			0,6	5	252	749	38189	740	3610,6
80	800	813	3,8	7	952	750	25410	1240	3610,6
			4,1	6	1534	700	20023	1058	3610,6
			4,8	5	530	694	16605	801	1443,1
			6,1	5	464	704	13710	1063	744,9
			2,6	5	791	756	33013	1065	3610,6
		841	2,0	6	455	782	37557	919	3610,6
			0,9	6	364	871	34850	941	3610,6
			1,5	8	399	841	35964	1286	3610,6
			0,3	8	373	818	32165	1037	3610,6
			3,5	6	283	736	31350	710	3610,6
		869	1,5	13	121	841	31719	678	3610,6
			0,6	8	158	920	28065	780	3610,6
			1,1	11	122	820	26944	950	3610,6
			0,5	9	195	885	30754	837	3610,6
			1,6			-			3610,6

Table A.5: Computational results for instances with $k = 80$.

References

[1] L. Simonetti A. Lucena, N. Maculan. Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Science*, 7(3):289–311, 2010.

[2] J.A. Bondy and V. Chvàtal. A method in graph theory. *Discrete Mathematics*, 15 (2):111 – 135, 1976.

[3] F. Carrabs, R. Cerulli, M. Gaudioso, and M. Gentili. Lower and upper bounds for the spanning tree with minimum branch vertices. *Computational Optimization and Applications*, 56(2):405–438, 2013.

[4] V Chvàtal. On hamilton's ideals. *Journal of Combinatorial Theory, Series B*, 12 (2):163 – 168, 1972.

[5] V. Chvàtal. Tough graphs and hamiltonian circuits. *Discrete Math.*, 5(3):215–228, 1973.

[6] V. Chvàtal and P. Erdös. A note on hamiltonian circuits. *Discrete Mathematics*, 2(2):111 – 113, 1972.

[7] Carlos Contreras-Bolton, Gustavo Gatica, Carlos Rey Barra, and Víctor Parada. A multi-operator genetic algorithm for the generalized minimum spanning tree problem. *Expert Systems with Applications*, 50:1 – 8, 2016.

[8] Alexandre Salles da Cunha, Luidi Simonetti, Abilio Lucena, and Bernard Gendron. Formulations and exact solution approaches for the degree preserving spanning tree problem. *Networks*, 65:329–343, 2015.

[9] G. A. Dirac. Some theorems on abstract graphs. *Proceedings of the London Mathematical Society*, s3-2(1):69–81, 1952.

[10] M. Dror, M. Haouari, and J. Chaouachi. Generalized spanning trees. *European Journal of Operational Research*, 120(3):583 – 592, 2000.

[11] J. Edmonds. Submodular functions, matroids and certain polyhedra. *Combinatorial structures and their applications*, pages 69–87, 1970.

[12] Kapali P. Eswaran and R. Endre. Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976.

[13] C. Feremans. *Generalized spanning trees and extensions*. PhD thesis, Université Libre de Bruxelles, 2001.

[14] C. Feremans, M. Labbé, and G. Laporte. A comparative analysis of several formulations for the generalized minimum spanning tree problem. *Networks*, 39(1): 29–34, 2002.

[15] C. Feremans, M. Labbé, and G. Laporte. The generalized minimum spanning tree problem: Polyhedral analysis and branch-and-cut algorithm. *Networks*, 43 (2):71–86, 2004.

[16] Corinne Feremans, Martine Labbé, and Gilbert Laporte. Generalized network design problems. *European Journal of Operational Research*, 148:1–13, 2003.

[17] Corinne Feremans, Martine Labbé, Adam N. Letchford, and Juan-José Salazar-González. Generalized network design polyhedra. *Networks*, 58(2):125–136, 2011.

[18] Cristiane Maria Ferreira, Luiz Ochi, Victor Parada, and Eduardo Uchoa. A grasp-based approach to the generalized minimum spanning tree problem. *Expert Syst. Appl.*, 39:3526–3536, 2012.

[19] M. Fischetti, J. J. Salazar González, and P. Toth. The symmetric generalized traveling salesman polytope. *Networks*, 26(2):113–123, 1995.

[20] M. Fischetti, J. J. Salazar González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45 (3):378–394, 1997.

[21] Bernard Fortz and Martine Labbé. *Polyhedral Approaches to the Design of Survivable Networks*, pages 367–389. Springer US, Boston, MA, 2006.

[22] Tetsuya Fujie. The maximum-leaf spanning tree problem: Formulations and facets. *Networks*, 43:212–223, 2004.

[23] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[24] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, 1977.

[25] Luisa Gargano, Pavol Hell, Ladislav Stacho, and Ugo Vaccaro. *Spanning Trees with Bounded Number of Branch Vertices*, pages 355–365. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[26] Bruce Golden, Saahitya Raghavan, and Daliborka Stanojevic. Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17:290–304, 08 2005.

[27] 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.

[28] Martin Grötschel. On the monotone symmetric travelling salesman problem: Hypohamiltonian/hypotraceable graphs and facets. *Mathematics of Operations Research*, 5(2):285–292, 1980.

[29] Martin Grötschel and Clyde L. Monma. Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM J. Discrete Math.*, 3:502–523, 1990.

[30] Martin Grötschel, C. Monma, and M. Stoer. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, 43, 1995.

[31] Stoer M. Grotschel M., Monma C. Design of survivable networks. *Handbooks in OR/MS*, 7 on Network models:chap 10, pp 617–672, 1995.

[32] Gregory Gutin and Daniel Karapetyan. A memetic algorithm for the generalized traveling salesman problem. *Natural Computing*, 9(1):47–60, Mar 2010.

[33] Henry-Labordere. The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem. *RAIRO Operations Research*, B2:43–49, 1969.

[34] F. K. Hwang and Dana S. Richards. Steiner tree problems. *Networks*, 22(1): 55–89, 1992.

[35] E. Ihler, G. Reich, and P. Widmayer. Class steiner trees and vlsi-design. *Discrete Applied Mathematics*, 90(1–3):173 – 194, 1999.

[36] Hervé Kerivin and A. Ridha Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[37] Joseph 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.

[38] Mercedes Landete, Alfredo Marín, and José Luis Sainz-Pardo. Decomposition methods based on articulation vertices for degree-dependent spanning tree problems. *Computational Optimization and Applications*, 68(3):749–773, Dec 2017.

[39] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10 (1):96–115, 1927.

[40] Massinissa Merabet and Miklós Molnár. Generalization of the Minimum Branch Vertices Spanning Tree Problem. Research report, Nanyang Technological University, Singapore, November 2016.

[41] Clyde L. Monma and David F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4):531–541, 1989.

[42] Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241, 1995.

[43] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial optimization*. Wiley, New York, NY, USA, 2014.

[44] Temel Öncan, Jean-François Cordeau, and Gilbert Laporte. A tabu search heuristic for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 191(2):306 – 319, 2008.

[45] Manfred W. Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[46] P. C. Pop, O. Matei, and C. Sabo. A new approach for solving the generalized traveling salesman problem. In María J. Blesa, Christian Blum, Günther Raidl, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, pages 62–72, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[47] Petrică C. Pop. The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances. *European Journal of Operational Research*, 2019.

[48] Petrica C. Pop, W. Kern, and G. Still. A new relaxation method for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 170(3):900 – 908, 2006.

[49] Petrică C. Pop, Oliviu Matei, Cosmin Sabo, and Adrian Petrovan. A two-level solution approach for solving the generalized minimum spanning tree problem. *European Journal of Operational Research*, 265(2):478 – 487, 2018.

[50] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[51] Jacques Renaud and Fayez F. Boctor. An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal of Operational Research*, 108(3):571 – 584, 1998.

[52] J. P. Saskena. Mathematical model of scheduling clients through welfare agencies. *Journal of the Canadian Operational Research Society*, 8:185–200, 1970.

[53] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, New York, NY, USA, 1998.

[54] John Silberholz and Bruce Golden. *The Generalized Traveling Salesman Problem: A New Genetic Algorithm Approach*, pages 165–181. Springer US, Boston, MA, 2007.

[55] Selene Silvestri, Gilbert Laporte, and Raffaele Cerulli. A branch-and-cut algorithm for the minimum branch vertices spanning tree problem. *Computers and Operations Research*, 81:322 – 332, 2017.

[56] R. Sousselier. Probìeme no. 29: Le cercal des irascible. *Rev. Franc. Rech. Operat.*, 7:405–406, 1963.

[57] R.C. Garg P. Sen S.S. Srivastava, S. Kumar. Generalized traveling salesman problem through n sets of nodes. *CORS Journal*, 7:97–101, 1969.

[58] M. Stoer. Design of survivable networks. *Lecture Notes in Mathematics*, vol. 1531, 1992.

[59] L.A. Wolsey. *Integer Programming*. Wiley, Hoboken, New Jersey, 1998.