UNIVERSIDADE FEDERAL FLUMINENSE

UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THIAGO GOUVEIA DA SILVA

The Minimum Labeling Spanning Tree and Related Problems

NITERÓI

2018

UNIVERSIDADE FEDERAL FLUMINENSE

UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THIAGO GOUVEIA DA SILVA

The Minimum Labeling Spanning Tree and Related Problems

Thesis presented to Universidade Federal Fluminense and Université d'Avignon et des Pays de Vaucluse in partial fulfillment of the requirements for the degree of Doctor of Science/Philosophy.

Topic Area: Algorithms and Optimization.

Advisors:
Luiz Satoru Ochi
Philippe Michelon

NITERÓI

2018

Ficha catalográfica automática - SDC/BEE

```
S586t   Silva, Thiago Gouveia da
          The Minimum Labeling Spanning Tree and Related Problems /
        Thiago Gouveia da Silva ; Luiz Satoru Ochi, orientador.
        Niterói, 2018.
          180 f. : il.

          Tese (doutorado)-Universidade Federal Fluminense, Niterói,
        2018.

        DOI: http://dx.doi.org/10.22409/PGC.2018.d.05398864408

          1. Grafo. 2. Metaheurística híbrida. 3. Heurística. 4.
        Poliedro. 5. Produção intelectual. I. Título II. Ochi,Luiz
        Satoru, orientador. III. Universidade Federal Fluminense.
        Escola de Engenharia.

                                              CDD -
```

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

THIAGO GOUVEIA DA SILVA

The Minimum Labeling Spanning Tree and Related Problems

Thesis presented to Universidade Federal Fluminense and Université d'Avignon et des Pays de Vaucluse in partial fulfillment of the requirements for the degree of Doctor of Science/Philosophy.

Topic Area: Algorithms and Optimization.

Approved on November 1 by:

[illegible]

Prof. D.Sc. Luiz Satoru Ochi - Advisor, IC/UFF

[illegible]

Prof. Ph.D. Philippe Michelon - Advisor, UAPV, France

[illegible]

Prof. Ph.D. Serigne Gueye, UAPV, France

[illegible]

Prof. D.Sc. Lucídio dos Anjos Formiga Cabral, CI/UFPB

[illegible]

Prof. D.Sc. Manoel Bezerra Campêlo Neto, DEMA/UFC

[illegible]

Prof. D.Sc. Fábio Protti, IC/UFF

[illegible]

Prof. D.Sc. Simone de Lima Martins, IC/UFF

Niterói, 2018

To my beloved grandmother,
Hilda Amélia Gouveia.

"I have never been alone."

Acknowledgments

I would like to express my sincere gratitude to every single person that contributed directly or indirectly with this work, they were not few. First, I would like to thank my family, namely my mother Josélia Gouveia, my father Antônio Gabriel, my sister Gabriella Gouveia, and my grandmother Dona Hilda for the support in my whole life and for understanding my absence. I would like to give a special thank to my dear wife, Ahmina Raiara, for being this angel in my life, and to extend this thank to my father in law, William Coelho, to my mother in law, Valéria Solsona, and to the members of my "French family", Marion, Pascaline, and Alain. I would also like to thank my advisors, Prof. Luiz Satoru Ochi and Prof. Philippe Michelon. Without any doubt, this work would not be the same without them. In the same sense, I would like to thank Serigne Gueye for his help, availability, and for the valuable contributions on the polyhedral studies of this thesis. I would also like to thank Prof. Lucídio Cabral for all the scientific and personal support throughout my entire academic journey, someday I hope to be just like him. I would like to thank the members of the jury for the very important suggestions and contributions (I took every single one into account) which helped to improve the quality of this thesis. I would like to thank Teresa Cancela and Helio Augusto, from the IC-UFF staff, as well as Aude Mosca and Nathalie Pederzoli, from the UAPV staff, for the help and patience with all the bureaucratic things. I would like to thank IFPB for allowing me to participate of the doctoral program. I would like to thank the professors Fábio Protti, Loana Tito, Lúcia Drummond, and Eduardo Uchoa for the very inspiring classes. I would like to thank both the Universidade Federal Fluminense and Université d'Avignon et des Pays de Vaucluse for accepting me as an student, and to the Conselho Nacional de Pesquisa (CNPq), for the Science without Borders scholarship grant in the year I was in Avignon, France. Finally, I would like to thank all my friends! It is an extensive list and I apologize if a miss some name. Thank to my childhood friends Nailson (NegaoDB) Cunha, Pedro (PPBB) Paulo, Igor (Curuma) Luis, Alysson (Buda) Lívio, and Álan (Lan Lan Boy) Lívio. Thank to my high school friends Thiago (Curva) Curvelo and Nilton (NJ) Herthel. Thank to my roommates Gustavo (HoN) Rezende, Gilberto (Chimba) Farias, Hugo (El Mulo) Kramer, Eyder (Papai) Rios, Eduardo (Espanhol de Areia) Queiroga, Thaylon (Gigante guerreiro), Bruno (Ixpó), and José (Cubano) Angel. Thank to the fellows from LABIC Edcarlos, Igor, Elio, and Renatha. Thank to my dear friends from Avignon Bruno Rosa, Fernanda Fernandes, Marco Fampa, Sheila, Luis (Luisito) Flores, Vitor (Vitinho) Ponciano, Rosa Figueiredo, Michelli Knetchel, Elvys Linhares, and Rodrigo (tu tomaste mi catchaça) Miyake. Thank to my fellows from DOTA (yes, I need some kind of relax...) Vitor (VNS, Vinissius) Nesselo, Arthur (Tuta) Kramer, Anderson (Danda) Rufino, and Thyn. In the end, I would like to thank (in memoriam) my cousin and best friend Jefferson (Juga Bua) Rufino. You were very special in my life and I know you are cheering for me, wherever you are right now.

Resumo

O Problema da Árvore Geradora com Rotulação Mínima (MLSTP, do inglês *minimum labeling spanning tree problem*) é um problema de otimização combinatória que consiste em encontrar uma árvore de cobertura em um grafo com arestas rotuladas, isto é, um grafo no qual cada aresta possui um rótulo associado, utilizando o menor número de rótulos. Este problema é NP-difícil e tem atraído bastante atenção em pesquisas nos últimos anos. Por sua vez, o Problema Generalizado da Árvore Geradora com Rotulação Mínima (GMLSTP, do inglês *generalized minimum labeling spanning tree problem*) é uma generalização do MLSTP na qual se permite que múltiplos rótulos sejam associados a uma aresta. Ambos os problemas tem aplicações práticas em áreas importantes, como Projeto de Redes de Computadores, Projeto de Redes de Transporte Multimodais e Compactação de Dados. Esta tese aborda vários problemas de conectividade definidos em grafos com arestas rotuladas, em especial o Problema da Árvore Geradora com Rotulação Mínima e sua versão generalizada. As contribuições neste trabalho podem ser classificadas entre teóricas e práticas. Dentre as contribuições teóricas, introduzimos novos conceitos, definições, propriedades e teoremas úteis em relação a grafos com arestas rotuladas, bem como um estudo poliédrico sobre o GMLSTP. Dentre as contribuições práticas, propusemos novas heurísticas — como o algoritmo baseado na metaheurística MSLB e a heurística construtiva pMVCA — e métodos exatos — como novas formulações matemáticas e algoritmos branch-and-cut — para resolver o GMLSTP. Os experimentos computacionais realizados utilizando conjuntos de instâncias bem estabelecidos para o MLSTP são relatados, mostrando que as novas abordagens introduzidas neste trabalho alcançaram os melhores resultados para métodos heurísticos e exatos em comparação com estado da arte da literatura.

Palavras-chave: *Grafo com arestas rotuladas*, *Árvore de cobertura*, *Meta-heurística*, *Combinatória poliédrica*, *Branch-and-bound.*

Abstract

The minimum labeling spanning tree problem (MLSTP) is a combinatorial optimization problem that consists in finding a spanning tree in a simple edge-labeled graph, i.e., a graph in which each edge has one label associated, by using a minimum number of labels. It is an NP-hard problem that has attracted substantial research attention in recent years. In its turn, the generalized minimum labeling spanning tree problem (GMLSTP) is a generalization of the MLSTP that allows the situation in which multiple labels can be assigned to an edge. Both problems have several practical applications in important areas such as computer network design, multimodal transportation network design, and data compression. This thesis addresses several connectivity problems defined over edge-labeled graphs, in special the minimum labeling spanning tree problem and its generalized version. The contributions in this work can be classified between theoretical and practical. On the theoretical side, we have introduced new useful concepts, definitions, properties and theorems regarding edge-labeled graphs, as well as a polyhedral study on the GMLSTP. On the practical side, we have proposed new heuristics — such as the metaheuristic-based algorithm MSLB, and the constructive heuristic pMVCA — and exact methods — such as new mathematical formulations and branch-and-cut algorithms — for solving the GMLSTP. Computational experiments over well established benchmarks for the MLSTP are reported, showing that the new approaches introduced in this work have achieved the best results for both heuristic and exact methods in comparison with the state-of-the-art methods in the literature.

Keywords: *Edge-labeled graph*, *Spanning tree*, *Metaheuristic*, *Polyhedral Combinatorics*, *Branch-and-bound*.

Résumé

Soit L un ensemble fini d'élements appelés étiquettes. On appelle graphe étiqueté simple, un graphe simple dans lequel à chaque arête est associée une étiquette prise dans L. Le problème de l'arbre couvrant de nombre d'étiquettes minimal (en anglais: the *minimum labeling spanning tree problem*, MLSTP) est un problème d'optimisation combinatoire consistant à trouver un arbre couvrant dans un graphe étiqueté simple en utilisant un nombre minimum d'étiquettes. Le problème est NP-dur. Il a fait l'objet d'un nombre important de recherche au cours des dernières années. L'une de ces directions de recherche a par ailleurs conduit à l'étude d'une généralisation dite problème generalisé de l'arbre couvrant de nombre d'étiquettes minimal (en anglais: the *generalized minimum labeling spanning tree problem*, GMLSTP). Le problème GMLSTP modélise les situations dans lesquelles plusieurs étiquettes peuvent être assignées à une arête. Les deux problèmes ont plusieurs applications pratiques dans des domaines importants tels que la conception de réseaux informatiques, la conception de réseaux de transport multimodaux et la compression de données. Nous proposons dans cette thèse plusieurs résultats théoriques contribuant à l'implantation de nouveaux schémas de résolution pratique de ces problèmes. En particulier, sur le plan théorique, nous avons introduit de nouveaux concepts, définitions, propriétés et théorèmes utiles, ainsi qu'une étude polyédrale du domaine des points réalisables d'une nouvelle formulation de GMLSTP. Cette formulation et son analyse ont permi le développement d'algorithmes de branchement et de coupe (branch-and-cut) pour la résolution exacte des problèmes. De nouvelles heuristiques ont été également développées — telles que l'algorithme basé sur la métaheuristique MSLB, et l'heuristique constructive pMVCA. Des résultats d'expériences numériques sur des benchmarks du problème MLSTP sont données. Elles démontrent la qualité des approches proposées dans cette thèse puisque, aussi bien pour les approches exactes qu'approchées, nous obtenons, comparativement à l'état de l'art du domaine, les meilleurs résultats de la littérature.

Mots-clés: *Graphes à arêtes étiquetées*, *Arbre couvrant*, *Métaheuristiques*, *Polyédres*, *Branch-and-bound*.

List of Figures

List of Tables

List of Acronyms

ALD	Arc-labeled digraph
BC	Branch-and-cut
BCP	Branch-and-cut-and-price
BKS	Best-known solution
B-MLSTP	Bounded minimum labeling spanning tree problem
B-MRSTP	Bounded minimum representation spanning tree problem
B-SCP	Decision version of the set covering problem
CCMLSTP	Cost-constrained minimum label spanning tree problem
CEP	Cluster editing problem
CGA	Carousel greedy algorithm
CTSP	Colorful traveling salesman problem
DFS	Depth-first search
ELG	Edge-labeled digraph
GDP	Gross Domestic Product
GMLSTP	Generalized minimum labeling spanning tree problem
IGA	Iterated greedy algorithm
kLSFP	K-labeled spanning forest problem
LB	Local branching
LCMSTP	Label-constrained minimum spanning tree problem
LCP	Label covering problem
LH	Labeled hypercube
LR	Linear Relaxation
MCGFP	Min-color generalized forest problem
MC/MLSTP	Minimal cost/minimal label spanning tree problem
MFMLP	Maximum flow minimal labeling problem
MIP	Mixed-integer programming
MLAP	Minimal labeling arborescence problem
MLEkCGP	Minimum labeling edge-k-connected graph problem

MLGCP	Minimum labeling global cut problem
MLGFP	Minimum labeling given flow problem
MLLkCGP	Minimum labeling label-k-connected graph problem
MLPP	Minimum labeling path problem
MLSCP	Minimal labeling strongly connected problem
MLstCP	Minimum labeling s-t cut problem
MLSteiner	Minimum labeling Steiner problem
MLSTP	Minimum labeling spanning tree problem
MLVBGP	Minimum labeling vertex-biconnected graph problem
MMCC	Maximal monochromatic connected components
MRSTP	Minimum representation spanning tree problem
MSLB	Multi-start local branching
MSTP	Minimum spanning tree problem
MVCA	Maximum vertex covering algorithm
PC-MLT	Prize-collecting minimum labeling tree problem
pMVCA	Parametrized version of MVCA
RCL	Restricted Candidate List
SA	Simulated annealing
SC	Supply chain
SCP	Set covering problem
STP	Steiner tree problem in graphs

Contents

Part I

Object, Scope, and Motivation

The first part of this thesis presents the object of this research, delimits the scope of the work, and justifies the choice of this theme by emphasizing its relevance. Initially, it introduces the minimum labeling spanning tree problem along with some basic concepts and definitions and discuss the motivations to study it. In the sequel, an extensive literature review describes the state-of-the-art exact, heuristic and meta-heuristic methods for the problem.

Chapter 1

Introduction

Trees are one of the most fundamental concepts in Graph Theory. A tree is an undirected connected graph that has no cycles, or, alternatively, it is an undirected graph in which any two vertices are connected by a unique simple path. In its turn, a spanning tree of a graph G is a subgraph which is a tree and spans all the vertices of G. Spanning trees are closely related to connected graphs in the sense that a graph has a spanning tree if and only if it is connected. Also, a spanning tree is a connected graph with the minimum number of edges.

Given a connected undirected weighted graph G, the minimum spanning tree problem (MSTP) is a classical optimization problem that aims to find a spanning tree of G such that the sum of the weights of the selected edges is minimum. On the one hand, there are some algorithms that solve the MSTP in polynomial time, e.g. the Prim's algorithm and the Kruskal's algorithm (Cormen et al., 2009). On the other hand, several problems derived from the MSTP are NP-Hard, such as the Steiner tree problem (Karp, 1972), the generalized minimum spanning tree problem (Myung et al., 1995), the capacited minimum spanning tree problem (Jothi and Raghavachari, 2005), and the degree constrained minimum spanning tree problem (Bui and Zrncic, 2006), among others.

The main subject of this Thesis is the minimum labeling spanning tree problem (MLSTP), another variant of the MSTP. The MLSTP is an NP-hard combinatorial optimization problem, introduced by Chang and Leu (1997), that consists in finding a spanning tree in an edge-labeled graph (ELG), a graph in which each edge has one label associated, by using a minimum number of labels. The MLSTP has applications in areas such as computer networks (Consoli et al., 2009), multimodal transportation networks (Van-Nes, 2002), and data compression (Chwatal et al., 2009), as discussed later in Section 1.2.

ELGs are commonly used to model situations where it is desirable to represent the same characteristic among different regions of the graph. One can say ELGs can represent qualitative properties instead of quantitative ones. Several problems defined over ELGs have been addressed in the last few years, such as the MLSTP (Chang and Leu, 1997), the minimum labeling s-t-path problem (Carr et al., 2000), the minimum labeling s-t cut problem (Coudert et al., 2007), the minimum labeling Steiner problem (Cerulli et al., 2006a), the labeled maximum matching problem (Carrabs et al., 2009), and the maximum flow with the minimum number of labels (Granata et al., 2013), among others. All these variants illustrate the importance of the problem in the scientific literature.

This work presents new useful concepts and definitions regarding ELGs; heuristic, exact and hybrid approaches to solve the MLSTP; a polyhedral study of this problem; and a brief discussion about problems related to the MLSTP. In the remaining of this Chapter we introduce some basic concepts and definitions which are necessary to fully understand our contribution. We also discuss the motivations for studying the MLSTP, present the general and specific objectives of this work, and make a brief overview of the next chapters of this Thesis.

1.1 Basic concepts and definitions

In this Section we introduce some basic concepts and definitions necessary for understanding this work, as well as conventions that will be used throughout this Thesis. First, we state the concept of ELG and give the formal definition of the MLSTP:

Definition 1.1. *An edge-labeled graph is an undirected simple graph $G = (V,E,L)$, in which V is the set of vertices, E is the set of edges, L is the set of labels, and $l(e) \in L$ represents the label associated with the edge e, $\forall e \in E$.*

Definition 1.2. *Given a connected ELG $G = (V,E,L)$, the MLSTP aims to find a spanning tree $T = (V,E',L')$, such that $E' \subseteq E$, $L' \subseteq L$, and $|L'|$ is minimized.*

An equivalent definition for the MLSTP is given by Brüggemann et al. (2003), based on the concept of subgraph induced by a set of labels.

Definition 1.3. *Given an ELG $G = (V,E,L)$, and a subset of labels $L' \subseteq L$, $G[L']$ is the spanning subgraph of G induced by the set of edges $E(L') = \{e \in E \mid l(e) \in L'\}$.*

Definition 1.4. *Let $G = (V,E,L)$ be a connected ELG, the MLSTP aims to find a smallest cardinality subset $L' \subseteq L$ such that $G[L']$ is connected.*

Proving the equivalence between Definitions 1.2 and 1.4 is straightforward. Indeed, as observed by Xiong et al. (2005a), if $G[L']$ is connected, then any spanning tree of $G[L']$ has at most $|L'|$ labels; if L^* is the smallest set of labels such that $G[L^*]$ is connected, then any spanning tree of $G[L^*]$ is a minimum labeling spanning tree of G. Throughout this Thesis we use this Definition 1.4 of the MLSTP.

Let $G = (V,E,L)$ be an ELG as given in Definition 1.1. In order to standardize its representation:

- The elements of the set of vertices V will be represented by positive integer numbers, for instance: $V = \{1,2,3,\ldots\}$;
- The elements of the set of labels L will be represented by uppercase letters, for instance: $L = \{A,B,C,\ldots\}$;
- Each element of the set of edges E will be represented in the form $e = (p,q)$, where $p,q \in V$ are the endpoints of e, for instance: $E = \{e_1 = (1,2), e_2 = (1,3), e_3 = (2,3),\ldots\}$.

In order to provide a better visualization of the ELGs, as observed in the Figure 1.1, the graphs will have a color associated with each label $l \in L$, as indicated by a subtitle near each graph. For this reason, the terms label and color will be used interchangeably throughout this work. Additionally, there is a letter close to each edge (arc) indicating its label.

Figure 1.1 illustrates the MLSTP. Fig. 1.1a shows an ELG $G = (V,E,L)$ in which $V = \{1,2,3,4,5,6,7,8\}$, $L = \{A,B,C,D,E,F\}$, and the label $l(e)$ associated with the edge $e \in E$ is indicated by the letter close to it. Fig. 1.1b presents an optimal solution $G[L^*]$, for $L^* = \{A,D,F\}$, with cost $|L^*| = 3$.

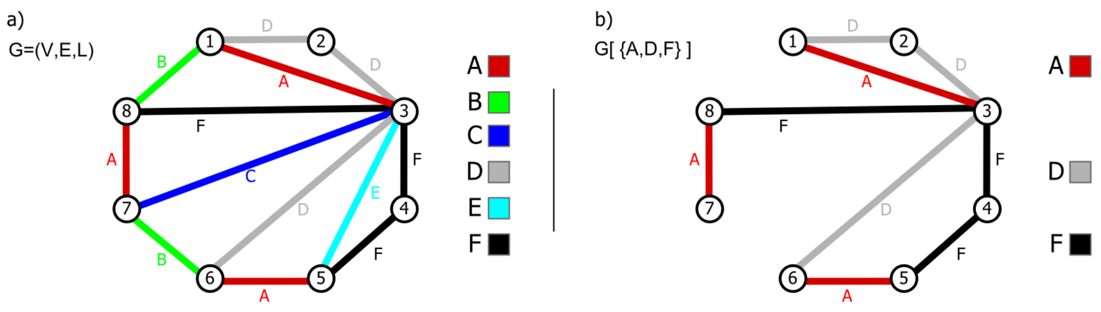


Figure 1.1: Small example for the MLSTP. **(a)** A small ELG $G = (V,E,L)$ with 8 vertices, 6 labels and 13 edges. **(b)** $G[\{A,D,F\}]$, an optimal solution for G

The generalized minimum labeling spanning tree problem (GMLSTP) is an extension of the MLSTP proposed by Chen et al. (2008) that allows *multi-labeling*. In this case, multiple labels can be assigned to an edge, which could be mathematically represented by the function $l^m : E \to 2^L$. Note that, if an edge $e \in E$ has $q = |l^m(e)| \geq 2$, it may be replaced with q parallel edges (leading to a multigraph), where each edge is associated with a different label of $l^m(e)$. Remark that any method proposed for the GMLSTP can be directly used on the MLSTP, since the latter is a special case of the former. Follows a simple definition of the GMLSTP based on the multigraph representation of multi-labeling.

Definition 1.5. *Let $G = (V,E,L)$ be a connected multi ELG, i.e. an ELG which could have parallel edges and loops. The GMLSTP aims to find a smallest cardinality subset $L' \in L$ such that $G[L']$ is connected.*

The example in Fig. 1.2 shows a small instance for the GMLSTP in which $V = \{1,2,3,4,5,6,7,8\}$, $L = \{A,B,C,D,E,F\}$. Figure 1.2a illustrates how multi-labeling is addressed by the function $l^m : E \to 2^L$; Fig. 1.2b shows the multigraph approach; Fig. 1.2c presents the graph $G[L^*]$, which is an optimal solution with the set $L^* = \{B,C,D,F\}$; and Fig. 1.2d shows an optimal labeled spanning tree obtained from $G[L^*]$.

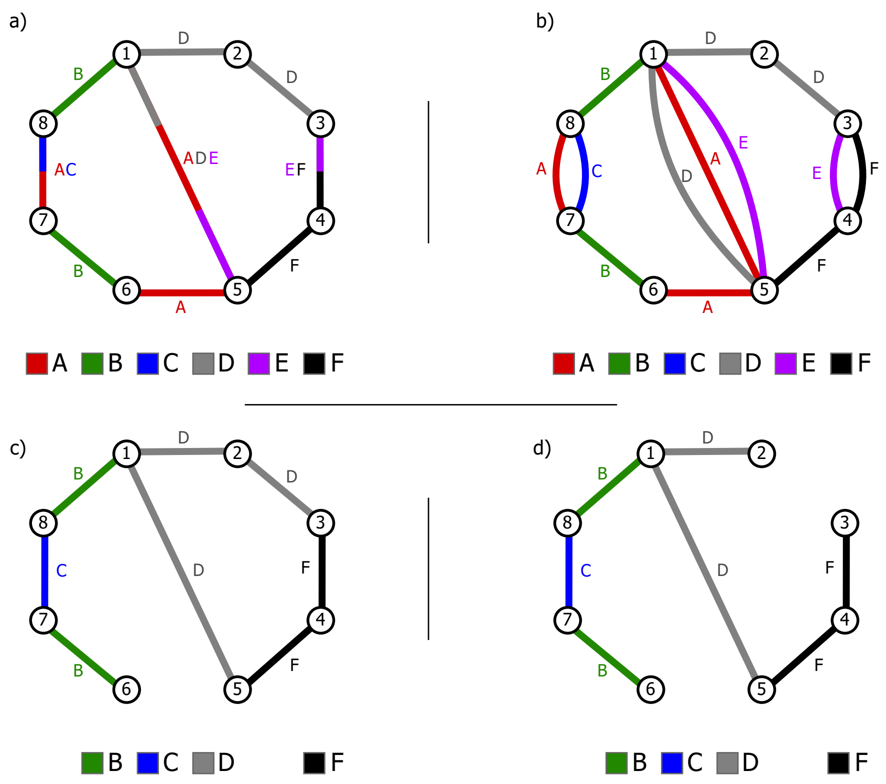


Figure 1.2: A small GMLSTP instance **(a)** with the $l^m : E \to 2^L$ representation; **(b)** with the multigraph representation. **(c)** A minimum cardinality subset $L^* \subseteq L$ such that $G[L^*]$ is connected. **(d)** A labeled spanning tree obtained from $G[L^*]$, thence an optimal solution for this instance

Finally, in some occasions, it is also convenient to define the concept of arc-labeled digraphs (ALD):

Definition 1.6. *An arc-labeled digraph is a directed graph $D = (V, A, L)$, in which V is the set of vertices, A is the set of arcs, L is the set of labels, and the function $l^a : A \rightarrow L$ returns the label associated with the input arc.*

Analogously to the definition of E on ELGs, each arc $a \in A$ is represented as $a = (p, q)$, for $p, q \in V$, such that p is the tail of a and q is its head. For instance: $A = \{a_1 = (1,2), a_2 = (1,3), a_3 = (2,3), \ldots\}$.

1.2 Motivation

Much attention has been spent with problems defined on ELGs in the last few years, specially with the MLSTP, one of the most (if not the most) widely studied problem regarding the ones defined on this kind of graph. The MLSTP has been addressed by many heuristic, exact and approximation methods. The main motivation of this work is to develop new exact, heuristic and hybrid techniques for the MLSTP in order to obtain better results both in terms of solution quality and computational time. Moreover, to highlight the relevance of the theme, it is possible to identify important applications for the MLSTP in several areas, such as:

Design of homogeneous computer networks: As described by Consoli et al. (2009), the MLSTP (GMLSTP) can be applied in the design of homogeneous computer networks to provide connectivity between all the nodes by using the minimum number of physical (or logical) media types. This kind of network presents some desirable characteristics, such as low cost to implement and lower complexity, which reduces the maintenance costs. Examples of different possible communication media types are (mono or multimode) fiber optic cables, twisted pair Ethernet cables, coaxial cables, or different technologies/frequencies of wireless links (Tanenbaum, 2003).

The Figure 1.3 illustrates the GMLSTP applied to the design of a homogeneous computer network. Fig. 1.3(a) shows the logical project of a network with six routers. The lines represent the possible links while the letters represent the media that is compatible with each router. Fig 1.3(b) demonstrates the ELG generated from the logical project: for each router there is a vertex, for each protocol there is a label, and for each link there is a set of edges corresponding to the common protocols between two routers. Fig. 1.3(c) presents the solution of the GMLSTP associated with the logical project.

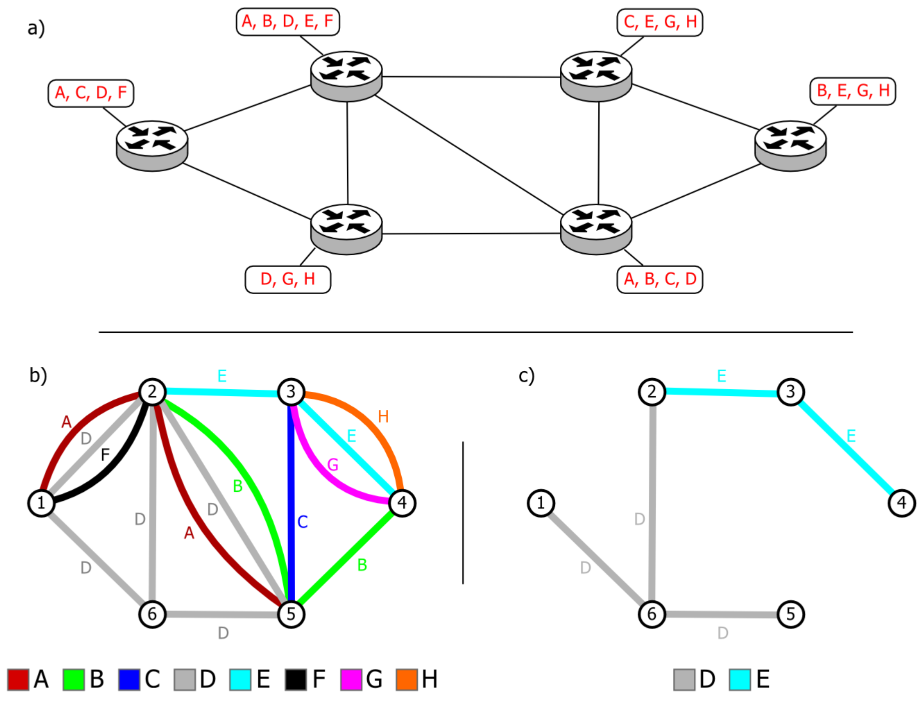


Figure 1.3: Example of a homogeneous computer network design modeled as an instance of the GMLSTP. **(a)** The logical network and **(b)** the ELG associated with it. **(c)** The solution for this instance of the GMLSTP

Design of multimodal transport networks : According to Van-Nes (2002), the multimodal transport happens when two or more transport modes are used for a trip, and a transfer is necessary between them. Real examples of multimodal transportation systems are the *Park & Ride Systems* (Lam et al., 2001), offered to attract car users to public transport; and the *TrainTaxi* system from Netherlands, which uses a dedicated taxi fleet to collect and distribute train passengers.

The fundamental component of the multimodal transport system is the design of the multimodal transport network. It consists in planning the interconnection between the different types of transport modes and their transfer points. Multimodal transport networks are an interesting approach to deal with the current (public) transport problems, such as deterioration of access to city centers, recurrent congestion and environmental impacts.

Figure 1.4 presents the design of the multimodal transport network of a hypothetical metropolis that aims to reduce the use of private cars. Fig. 1.4(a) presents the set of regions considered. Fig. 1.4(b) shows the new transport modes proposed. Fig. 1.4(c) describes the possible interconnections the project foresees.

The Figure 1.5 shows the GMLSTP instance created from the project of Figure 1.4. In this case, minimizing the number of labels means to use alternative transport methods to interconnect all desired regions with minimal cost. Fig 1.5(a) represents the input ELG and Fig. 1.5(b) presents a solution that connects all the proposed regions using only bicycle lanes, trains and ferry boats.

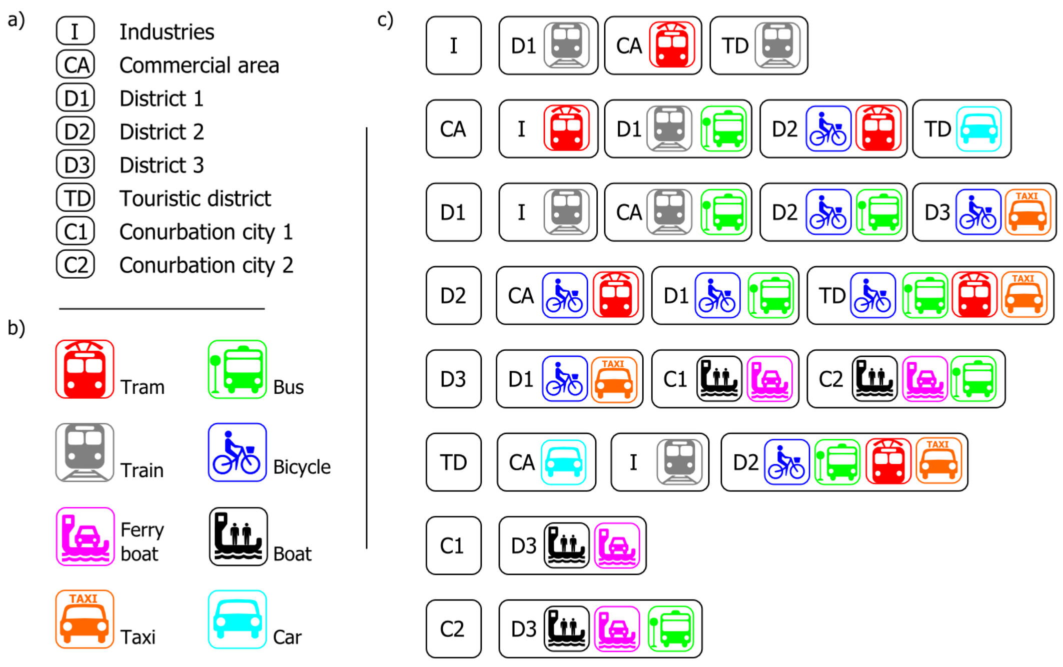


Figure 1.4: Example of a multimodal transport network design. **(a)** The set of locations, **(b)** transport modes, and **(c)** paths planned

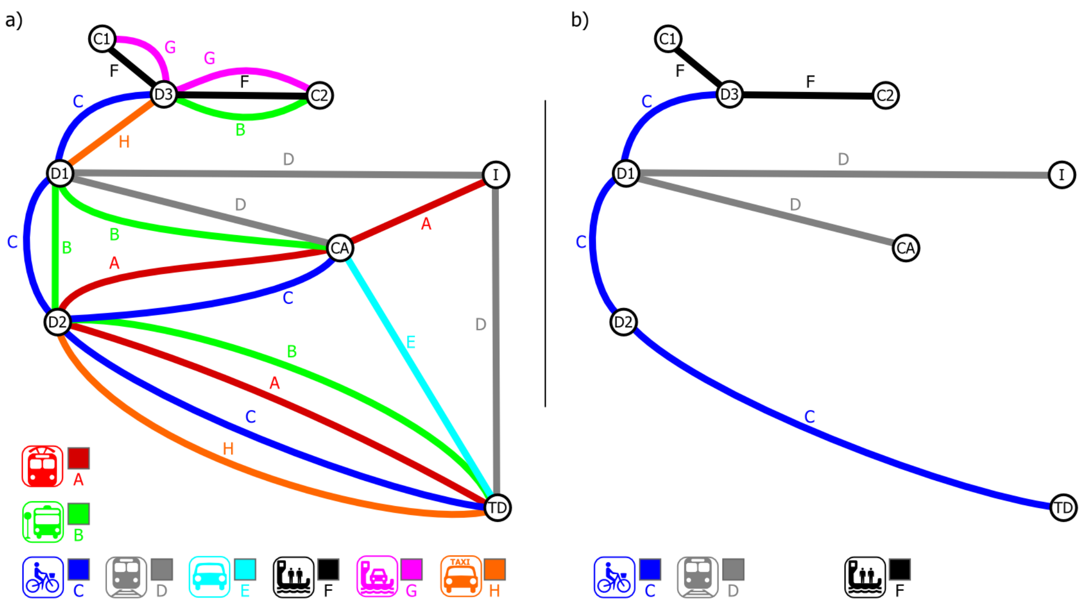


Figure 1.5: The GMLSTP applied to the design of a multimodal transport network. **(a)** The instance generated from the design of Fig. 1.4 and **(b)** the solution for this instance

Design of transportation networks : The Supply Chain (SC) covers all activities related to the flow and transformation of goods from the extraction of the raw material to delivery to the final user. The SC is composed of all entities involved directly or indirectly in this process, such as manufacturers, suppliers, warehouses, carriers and consumers (Ballou, 2006).

The transportation network is the set of transport modes, locations and routes by which a product can be delivered. The mode of transportation is the manner in which a product is moved from one location to another in the supply chain network. Commonly, the companies can choose between air, truck, rail, sea, and package carriers as modes of transport for products. According to Chopra and Meindl (2007), the transportation network is a relevant component of the costs of the SCs, accounting for more than ten percent of the United States Gross Domestic Product (GDP) in 2002.

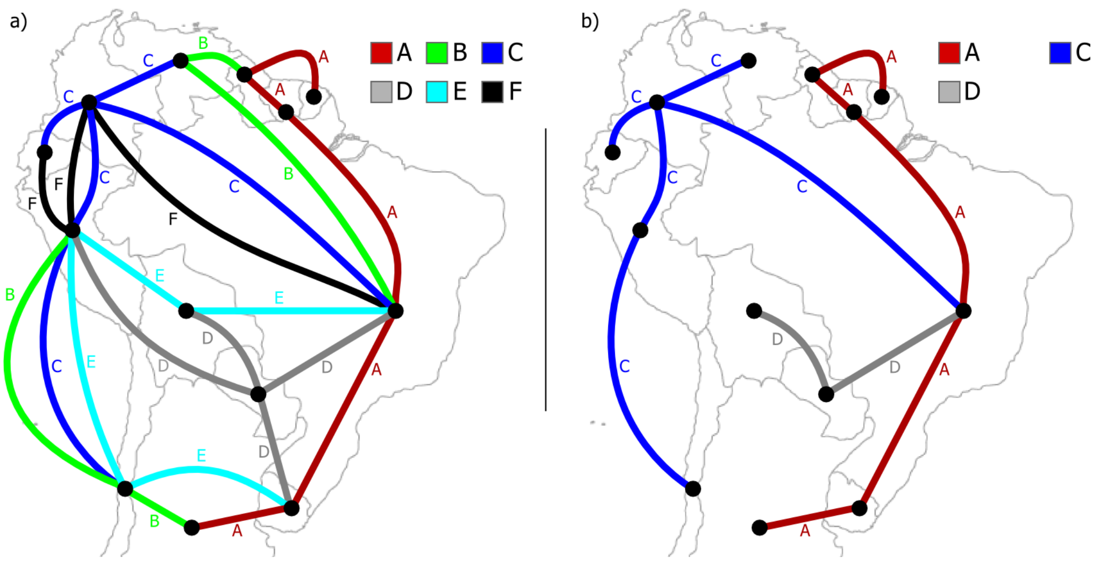


Figure 1.6: Example of a transportation network design. **(a)** The design modeled as an instance of the GMLSTP and **(b)** its solution

Figure 1.6 exemplifies the design of the transportation network of a hypothetical company that wants to deliver goods from Brazil to all the other countries of South America. Fig. 1.6(a) shows the ELG generated: the vertices represent the countries and each label is associated with a carrier that can be hired to move the products between the countries indicated by the edges. Fig. 1.6(b) presents a solution for this instance of the GMLSTP where all countries are served by hiring only the carriers *A*, *C* and *D*.

1.3 Objectives

The main objectives of this work are as follows.

- Review the MLSTP, describing practical applications and some of the most successful solution methods proposed in the literature.

- Perform a theoretical study on the ELGs, leading to new useful concepts and definitions that can be used by the heuristic and exact methods proposed.
- Develop exact approaches based on mixed-integer programming (MIP) for the MLSTP, as well as conduct a polyhedral study for the problem.
- Develop a general hybrid (MIP-heuristic) algorithm capable of solving large instances of the MLSTP in reasonable time.
- Address some variants and problems related to the MLSTP that arose during this study.

1.4 Thesis outline

The remainder of this work is organized as follows.

- Chapter 2 performs a literature review of both MLSTP and GMLSTP. The most successful heuristic, metaheuristic and exact methods for these problems are described.
- Chapter 3 presents new concepts and definitions related to the ELGs as well as theoretical results, such as reduction rules, lower bounds, and ELG transformation techniques.
- Chapter 4 describes a new MIP-based formulation composed only by the edge variables for the GMLSTP, namely CCut, a branch-and-cut algorithm, a new *branch-and-bound* strategy, new sets of valid inequalities and the computational experiments.
- Chapter 5 defines the polytope of CCut, proves that three new families of inequalities and the bounding constraints are facet defining under certain conditions, and compares the polytope defined by the formulations described in this work.
- Chapter 6 describes improvements for the exact methods introduced previously: a new mathematical formulation that extends CCut; two new branching strategies for solving the CCut model; and a new branch-and-cut algorithm. Further, computational experiments are performed to evaluate the proposed methods.
- Chapter 7 presents new constructive and local search heuristics and a hybrid metaheuristic for the GMLSTP, describing how an exact procedure is integrated into the heuristic framework. Computational experiments are reported comparing the proposed methods with the state-of-the-art ones.

- Chapter 8 addresses other connectivity problems defined on ELGs, such as the minimum labeling path problem and the maximum flow minimal labeling problem. Moreover, we propose extensions and/or adaptations of the colorful cuts formulation for solving these problems.

- Chapter 9 addresses the minimum labeling global cut problem, which aims to find the minimum number of labels whose removal disconnects the input ELG. We propose three new mathematical formulations for this problem and branch-and-cut algorithms to solve them.

- Chapter 10 introduces a new connectivity problem defined on ELGs, the minimum representation spanning tree problem. In contrast to the MLSTP, this new problem aims the homogeneity of each vertex in the solution graph. We propose a mathematical formulation for the problem, as well as two new constructive heuristics.

- Chapter 11 discuss the concluding remarks and the possible paths for the continuity of this research.

Chapter 2

Literature Review

This Chapter presents a literature review on the MLSTP, introducing some of the more important results regarding both this problem and the GMLSTP. First, a brief overview on works about the MLSTP and the GMLSTP is presented. In the sequel, we recall an NP-completeness proof for the problem and discuss more in deep the heuristic, metaheuristic, and exact methods with the best results in the literature. The reader should refer to Granata et al. (2013) for more information on problems formulated over ELGs, they have conducted an extensive survey on this kind of problem.

The MLSTP was proposed by Chang and Leu (1997), who have proved it is NP-hard by reduction from the set covering problem. Independently, Broersma and Li (1997) have proved that the MLSTP is NP-hard by a reduction from the minimum dominating set problem. Given f, the maximum number of edges with the same label, Brüggemann et al. (2003) have applied the local search technique *k-switching* and have proved that the MLSTP is polynomial time solvable if $f \leq 2$, but it is NP-hard and APX-complete for $f \geq 3$. Moreover, they have showed that the problem can be approximated with a factor equal to $f/2$ through *k-switching*.

Since the late 1990s, many heuristic and exact methods have been developed for the MLSTP. The most successful greedy heuristic algorithm is the maximum vertex covering algorithm (MVCA) (Chang and Leu, 1997). Given its capacity to achieve relatively good results in a short amount of time, the MVCA is commonly found in construction or rebuilding phases of metaheuristics. Section 2.2 addresses the MVCA in more detail. Recently, Cerrone et al. (2017) have proposed a new general greedy heuristic, namely the carousel greedy algorithm (CGA), and used the MLSTP to validate the concept. They have adapted the iterated greedy algorithm (IGA) (Ruiz and Stützle, 2007) and the pilot method (Duin, Voß, et al., 1999) to the MLSTP and compared them with the CGA, which achieved the best results. The CGA is presented in Section 2.3.

Several metaheuristics-based algorithms have been proposed for the MLSTP. The first one was the genetic algorithm of Xiong et al. (2005a), which has obtained better results in comparison with the MVCA. Cerulli et al. (2005) have analyzed the performance of the metaheuristics reactive tabu search, simulated annealing (SA) and VNS, while Xiong et al. (2006) developed the modified genetic algorithm (MGA), achieving the best results so far. Nummela and Julstrom (2006) have proposed three new genetic algorithms for the MLSTP, but, unfortunately, they have not compared them with the other metaheuristics in the literature.

Consoli et al. (2007) have proposed an hybrid VNS-SA method and Consoli et al. (2009) have implemented methods based on the metaheuristics GRASP and VNS, outperforming both MGA and VNS-SA. Moreover, Chwatal and Raidl (2010) have implemented an ant colony optimization method for the problem. Recently, Consoli et al. (2015) developed two new metaheuristics-based methods, COMPL and INTELL, that use the concepts of complementary space and auto-adjusting parameters over a VNS framework, achieving the best heuristic results for the problem so far. These methods are detailed in Section 2.4. Cerrone et al. (2016) have proposed a multi ethnic genetic algorithm for the MLSTP, but only presented preliminary experiments on small ELGs.

The first exact method for the MLSTP was proposed by Chang and Leu (1997): it consists of an implicit enumeration method based on an A^*-search algorithm. Chen et al. (2008) proposed the first mixed integer linear program (MIP) formulation for the problem based on the Miller−Tucker−Zemlin (MTZ) inequalities for cycle elimination, whereas Captivo et al. (2009) proposed a MIP formulation based on the root-oriented single commodity flow model (SCF).

In their extensive study on mathematical programming techniques for solving the GMLSTP, Chwatal and Raidl (2011) adapted previous formulations—SCF and MTZ—to the GMLSTP and proposed four new MIP models:

- The multicommodity flow formulation (MCF) extends the SCF with the introduction of multiple commodities.
- The cycle-elimination formulation (CE) ensures the feasibility of the integer solutions by enforcing the minimum number of edges and prohibiting cycles.
- The directed cut formulation (DCut) is a directed cut-based formulation with an exponential number of constraints.
- The epsilon connectivity formulation (EC) is a cut-based formulation with an exponential

number of constraints but without arc variables.

Chwatal and Raidl (2011) further proposed cuts to strengthen the models and introduced a polyhedral comparison to evaluate the relative quality of its linear relaxations (LR). They also implemented branch-and-cut (BC) and branch-and-cut-and-price (BCP) algorithms based on the formulations CE, DCut, and EC. The present investigation is focused on the DCut and EC models, which provided the best numerical results along with some strengthening cuts. Sections 2.5, 2.6 and 2.7 detail these formulations.

The next sections recall the NP-completeness proof provided by Chang and Leu (1997) for the MLSTP and discuss more deeply the most successful heuristic, metaheuristic, and exact methods proposed for the MLSTP.

2.1 NP-completeness proof

Chang and Leu (1997) have proved the MLSTP is NP-Complete by transforming the decision version of the set covering problem to the decision version of the MLSTP. Given an ELG $G = (V, E, L)$ and a constant $k \in \mathbb{Z}^+$, the bounded minimum labeling spanning tree problem (B-MLSTP) aims to decide if there is a spanning tree $T = (V, E', L')$ of G such that $|L'| \leq k$. It is easy to see that B-MLSTP is NP since a non-deterministic algorithm needs only to guess a subset of edges $E' \in E$ and check in polynomial time if $T = (V, E', L')$ is connected and if $|L'| \leq k$.

Given U the universe set, $\mathcal{S}$ a set of subsets of U, and a constant $k \in \mathbb{Z}^+$, the question associated with the decision version of the set covering problem, which is NP-complete (Karp, 1972), is if there exists a set $\mathcal{C} \subseteq \mathcal{S}$ such that $|\mathcal{C}| \leq k$ and $\bigcup_{s \in \mathcal{C}}(s) = U$. First, let $G(U, \mathcal{S}) = (V, E, L)$ be an ELG built as follows: let $V' = \{v^u \mid u \in U\}$, $V'' = \{v^s \mid s \in \mathcal{S}\}$, and $V = V' \cup V'' \cup \{v^*\}$, where v^* is a special vertex; let $L' = \{k^s \mid s \in \mathcal{S}\}$ and $L = L' \cup \{k^*\}$, where k^* is a special label; finally, let $E' = \{e = (v^*, v^s) \mid s \in \mathcal{S}\}$, such that $l(e) = k^*, \forall e \in E'$, let $E'' = \{e = (v^s, v^u) \mid s \in \mathcal{S}$ such that $u \in s\}$, such that $l(e) = k^s, \forall e \in E''$, and let $E = E' \cup E''$.

It is straightforward to verify that this construction can be accomplished in polynomial time and that $G(U, \mathcal{S}) = (V, E, L)$ has a spanning tree with k labels if and only if a set covering with $k - 1$ sets does exist. Figure 2.1 illustrates the construction of the graph $G(U, \mathcal{S})$. Fig. 2.1(a) presents the sets U and $\mathcal{S}$ of an example instance for the set covering problem. Fig. 2.1(b) shows the ELG $G(U, \mathcal{S}) = (V, E, L)$. Fig. 2.1(c) presents a solution: a minimum labeling spanning tree for the graph $G(U, \mathcal{S})$.

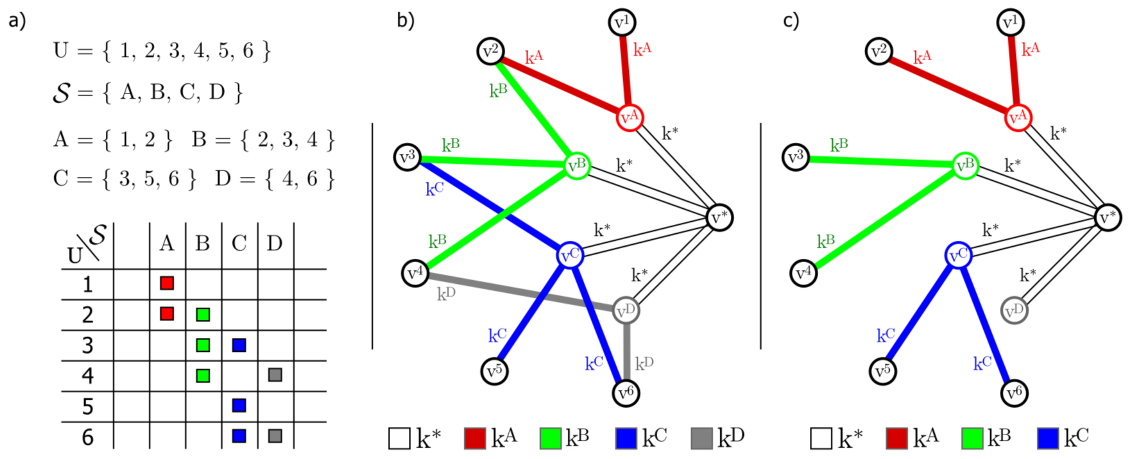


Figure 2.1: Transformation from the set covering problem to the MLSTP. **(a)** An input instance $(U,\mathbb{S})$ for the set covering problem. **(b)** the instance of the MLSTP originated from $(U,\mathbb{S})$. **(c)** A solution of the MLSTP for this input graph

2.2 MVCA

It is possible to state that MVCA is the most important constructive heuristic for the MLSTP. As introduced previously, it has been used to provide initial solutions or to complete partial ones by many metaheuristic-based methods, such as the VNS and GRASP procedures proposed by Consoli et al. (2009) and the COMPL and INTELL proposed by Consoli et al. (2015). Even other constructive heuristics, such as the Pilot Method (Cerulli et al., 2005) and the Carousel Greedy (Cerrone et al., 2017), rely on MVCA to build or to rebuild solutions.

As proposed by Chang and Leu (1997), the MVCA consists in: starting with an empty ELG $G = (V, E' = \emptyset, L' = \emptyset)$, iteratively add to L' the label l that covers as many uncovered vertices as possible, until G is connected. Chang and Leu (1997) have demonstrated that the time complexity of MVCA is $O(|V| \cdot |E| \cdot |L|)$.

Arguing that the MVCA could lead to unconnected solutions, Krumke and Wirth (1998) have proposed a modified version the MVCA in order to guarantee the correctness of the solutions generated. The modified MVCA starts with the solution $L' = \emptyset$ and, while $G[L']$ is not connected, at each iteration, adds to L' the label $l \in L \backslash L'$ that minimizes the number of connected components of $G[L']$. They have showed that the time complexity of the modified MVCA is $O(|E| + |L| \cdot \alpha(|E|,|V|) \cdot min\{|L| \cdot |V|, |E|\})$, where α is the inverse of the Ackerman's function. The Algorithm 2.1 presents the modified version of the MVCA.

In addition, Krumke and Wirth (1998) also have proved that MVCA can yield a solution no greater than $1 + 2ln|V|$ times optimal. Afterwards, Wan et al. (2002) have improved this bound to $1 + ln(|V| - 1)$. Given $E(L')$, $L' \subseteq L$, the set of edges with label $l(e) \in L'$, Xiong et al.

Algorithm 2.1: The Modified MVCA

```
1 procedure MVCA($G = (V, E, L)$)
2     Let $C \leftarrow \emptyset$ be the set of labels of the solution;
3     while $G[C]$ is not connected do
4         let $l \in L \backslash C$ be the label that minimizes the number of components of $G[C \cup \{l\}]$;
5         $C \leftarrow C \cup \{l\}$ ;
6     return $C$;
```

(2005b) have demonstrated that if $|E(\{l\})| \leq f$, $\forall l \in L$, then the MVCA has an approximation factor of H_f, where H_f is the fth harmonic number (equation 2.1). Later, they constructed a worst-case family of graphs such that the MVCA solution is exactly H_f times the optimal solution.

$$H_f = \sum_{i=1}^{f} i^{-1} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{f}. \tag{2.1}$$

Consoli et al. (2006) have studied several variants of the MVCA, such as the Frequency MVCA, the Random MVCA, and the Pilot-First MVCA. Recently, (Cerrone et al., 2017) have conducted an experiment on the quality of the selections performed by the MVCA. They have generated 10.000 ELGs randomly in such a way that the optimal solution is known in advance, and executed the MVCA for each graph. They have shown that the first selections of the MVCA are not good, once the average percentage of the selected labels that are in the optimal solution was less than 55%. On the other hand, for the last selections, the average percentage of the selected labels that are in the optimal solution was between 55 and 77%. Furthermore, in a second experiment, the authors have demonstrated that the MVCA performs very poorly when L has a small cardinality.

2.3 The Carousel Greedy Algorithm

The carousel greedy algorithm (CGA) is a generalized greedy heuristic proposed by (Cerrone et al., 2017) which seeks to overcome some of the known problems of the traditional greedy approaches, such as the poor quality of the first greedy choices. The authors have applied the CGA to the MLSTP and compared it with others constructive heuristics, namely the IGA, the Pilot Method and the MVCA. The CGA achieved the best performance.

The CGA adapted to the MLSTP is presented in the Algorithm 2.2. The method takes two parameters: $\alpha \in \mathbb{N}$ controls the number of iterations in the carousel phase, while $\beta \in (0,1]$ controls the size of the carousel. The first step (line 2) is to build an initial solution using the MVCA. The destruction phase (line 3) removes some labels from the initial solution, leaving a partial one. The carousel phase (lines 4 to 7) consists in removing the label that was first added to the current solution and to add a new one using the greedy criteria of MVCA. The final phase (lines 8 to 10) applies the MVCA until a feasible solution is obtained. Figure 2.2 illustrates an execution of CGA adapted for the MLSTP.

Algorithm 2.2: The Carousel Greedy Algorithm

```
1  procedure CGA($G = (V, E, L), \alpha, \beta$)
2      Let $C \leftarrow$ MVCA($G$) be the initial solution, and let $s = |C|$;
3      Remove from $C$ the $s\beta$ labels that were added last;
4      for $i = 1$ to $s\alpha$ do
5          Remove from $C$ the label that was first added to this set;
6          let $l \in L \backslash C$ be the label that minimizes the number of components of $G[C \cup \{l\}]$;
7          $C \leftarrow C \cup \{l\}$ ;
8      while $G[C]$ is not connected do
9          let $l \in L \backslash C$ be the label that minimizes the number of components of $G[C \cup \{l\}]$;
10         $C \leftarrow C \cup \{l\}$ ;
11     return $C$;
```

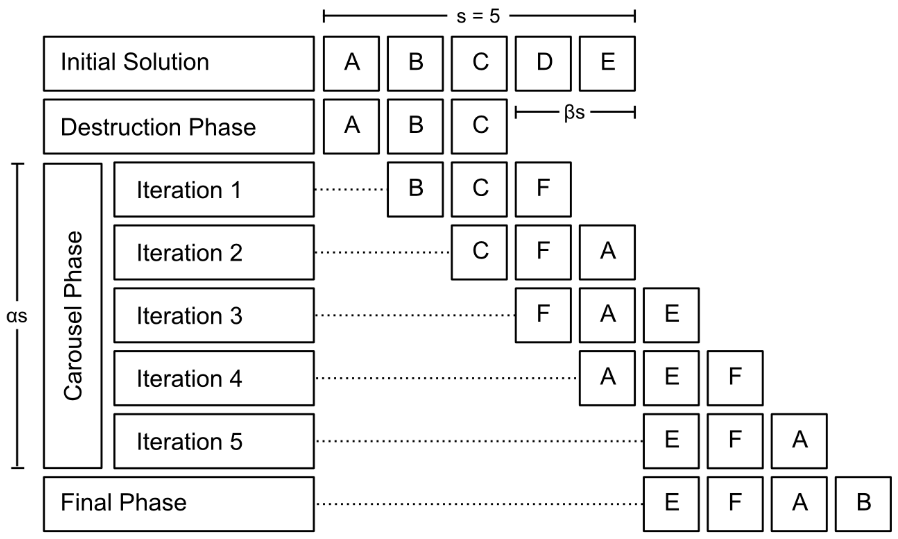


Figure 2.2: Illustration of an execution of the carousel greedy algorithm adapted for the MLSTP. The parameters for this execution are $s = 5$, $\alpha = 1$, and $\beta = 0.4$

2.4 VNS-Based Algorithms

This Section presents three algorithms based on the metaheuristic VNS, proposed by Consoli et al. (2009) and Consoli et al. (2015) for the MLSTP. These methods, specially the INTELL, have achieved the best results in comparison with the other metaheuristic-based methods in the literature.

The VNS-based algorithm proposed by Consoli et al. (2009), subsequently denominated VNS for short, starts from a randomly generated solution and, until a stop condition is reached, iteratively alternates between the *Shaking-Phase* and the *Local-Search* procedures, as detailed later. Algorithm 2.3 describes the VNS procedure. The *randomSolution* procedure (line 2) consists in adding labels randomly to the solution until it is connected. The parameters $k, k_{max} \in \mathbb{N}$, which control the cardinality of the visited neighborhoods, are set on line 4. The value of k_{max} is an important parameter to tune in order to obtain an optimal balance between intensification and diversification. The *Shaking-Phase* procedure (line 6) takes a solution C as input and performs k random modifications on it. Each modification consists in choosing (equiprobably) between adding/removing a random label to/from C. Finally, the *Local-Search* procedure (line 7) first completes infeasible solutions given by the *Shaking-Phase* by using the MVCA procedure and then it tries to delete labels, one by one, whilst maintaining feasibility.

Algorithm 2.3: VNS para o MLSTP.

```
1  procedure VNS($G = (V, E, L)$)
2      $C \leftarrow$ randomSolution();
3      repeat
4          $k \leftarrow 1, k_{max} \leftarrow \frac{4}{3}|C|$;
5          while $k < k_{max}$ do
6              $C' \leftarrow$ Shaking-Phase$(C, k)$;
7              $C' \leftarrow$ Local-Search$(C')$;
8              if $|C'| < |C|$ then
9                  $C \leftarrow C'$;
10                 $k \leftarrow 1, k_{max} \leftarrow \frac{4}{3}|C|$;
11             else
12                 $k \leftarrow k + 1$;
13     until stop condition;
14     return $C$;
```

Consoli et al. (2015) have proposed two enhancements to the previous VNS algorithm: the *complementary space search*, which builds a new solution by only using labels that are not in the current solution; and a probabilistic auto-adjustable local search procedure inspired by the simulated annealing metaheuristic. These two enhancements have improved the efficiency and robustness of the previous version of the VNS.

The first extension applied to VNS, the complementary space search, is an additional local search mechanism that aims to increase the diversification of the method, allowing it to visit different regions of the solution space. Given C, a solution for the MLSTP, the complementary space $\overline{C} = L \backslash C$ is the set of labels of L that are not in C. Once the main loop of the VNS (line 5) is not able to improve the current solution, the MVCA is used to produce a new solution by using only the set of labels $\overline{C}$. The VNS method enhanced with the complementary space search mechanism is denominated COMPL.

Figure 2.3 illustrates an execution of the method COMPL: if begins from the solution C_0, generated randomly, and applies on it the procedures *Shaking-Phase* and *Local-Search* from the previous version of VNS. Once these procedures were not able to enhance C_0, the complementary space search is carried out on $\overline{C_0}$, yielding the solution C_1. Then, the local search finds the solution C_2, that is better than C_1, but cannot enhance it. The complementary space search generates the solution C_3, by using $\overline{C_2}$, and, since the method did not find any better solution, it returns C_2.

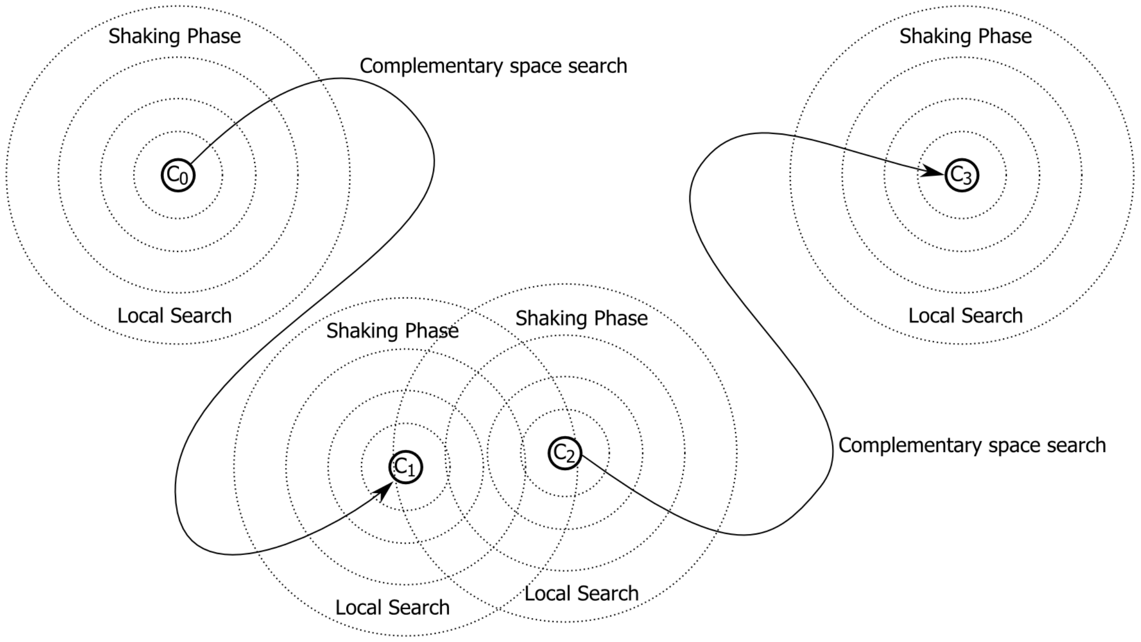


Figure 2.3: Example of execution of the algorithm COMPL. The circles represent the Shaking-Phase and the Local-Search procedures, while the arrow illustrates the complementary space search mechanism

The second improvement proposed by Consoli et al. (2015) is to replace the deterministic procedure *Local-Search* by a probabilistic local search heuristic inspired by the metaheuristic Simulated Annealing (SA). The proposed new local search uses a self-tuning parameter T that allows the method to adapt itself to the instance and react to the behavior of the algorithm. The acceptance probability of a worse label into a partial solution is evaluated according to the usual SA criterion, by the Boltzmann function. The resulting procedure, described in the flowchart of Figure 2.4, represents the INTELL algorithm, which has the best known metaheuristic results for the MLSTP in the literature.

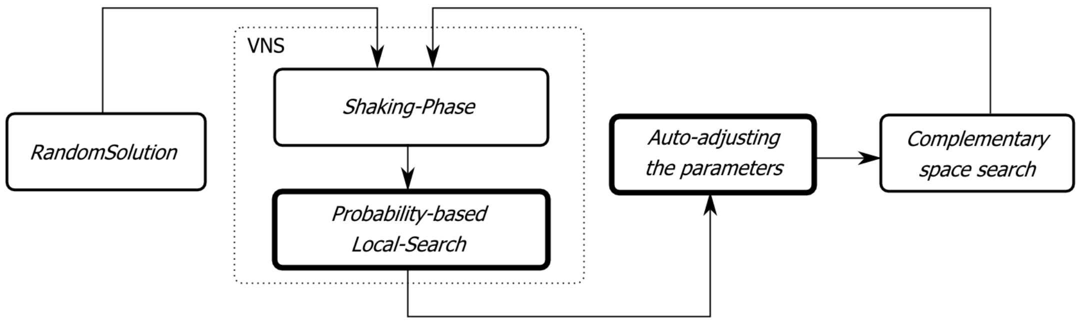

VNS
Shaking-Phase
RandomSolution
Probability-based Local-Search
Auto-adjusting the parameters
Complementary space search

Figure 2.4: Flowchart of the algorithm INTELL for the MLSTP. The highlighted components are the ones that differentiate the method INTELL from VNS and COMPL

2.5 The Directed Cut Formulation

The directed cut (DCut) formulation, proposed by Chwatal and Raidl (2011), is a root-oriented model that ensures the connectivity of the solution by enforcing the existence of a valid directed path from an arbitrary root vertex $r \in V$ to all other vertices of the graph. To this end, it defines the ALD $D = (V, A, L)$, in which every edge $e \in E$ yields two antiparallel arcs a, $\overline{a}$ to the set A. Let $e(a)$ be the originating edge of a, and let the function $l^a : A \to L$ denote the label of the arc a such that $l^a(a) = l(e(a))$.

The DCut formulation uses the set of binary variables $z_l \in \{0,1\}, \forall l \in L$ to indicate that the label l is in the solution, and the variables $y_a, \forall a \in A$, equals to 0, to show that the arc a is not used in the final arborescence. The program (2.2) through (2.7) presents the DCut formulation proposed by Chwatal and Raidl (2011). For sake of unified notation the formulation given here is adapted for the multigraph approach.

$$\text{Minimize} \sum_{l \in L} z_l \tag{2.2}$$

$$\text{s.t.} \quad \sum_{a\in\delta^-(S)} y_a \geq 1, \qquad \forall S \subseteq V\backslash\{r\}, \tag{2.3}$$

$$z_{l^a(a)} \geq y_a, \qquad \forall a \in A, \tag{2.4}$$

$$y_a + y_{\overline{a}} \leq 1, \qquad \forall a \in A, \tag{2.5}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L, \tag{2.6}$$

$$y_a \geq 0, \qquad \forall a \in A. \tag{2.7}$$

The objective function (2.2) aims to minimize the number of labels; the exponential number of directed cut constraints (2.3) ensures the feasibility of the solution, where $\delta^-(S)$ denotes the set of ingoing arcs of the cut set $[S, V\backslash S]$, for $S \subset V$; constraints (2.4) bind the label and arc variables; constraints (2.5) reinforce the formulation by prohibiting single edge circuits, where $\overline{a}$ represents the antiparallel arc of a; and expressions (2.6) and (2.7) define the domain of the variables.

The B&C algorithm proposed by Chwatal and Raidl (2011) for the DCut formulation separates inequalities (2.3) by computing the maximum flow from r to each vertex $v \in V\backslash\{r\}$. If the minimum $(r-v)$-cut found is less than 1, then the corresponding constraint is added to the model.

2.6 The Epsilon Connectivity Formulation

The epsilon connectivity formulation was motivated by the findings of Captivo et al. (2009), who showed (for flow formulations) that the correct optimal objective function value can be obtained even if the edge variables are continuous. Chwatal and Raidl (2011) extended this result for further GMLSTP formulations such as DCut and EC.

Hence, the EC formulation defines the continuous edge variable $x_e, \forall e \in E$, to denote that e is used or not in the final tree. The EC formulation uses the same set of binary variables $z_l \in \{0,1\}, \forall l \in L$, as the DCut model. The program (2.8) through (2.12) presents the EC formulation proposed by Chwatal and Raidl (2011). For sake of unified notation the formulation given here is adapted for the multigraph approach.

$$\text{Minimize} \sum_{l \in L} z_l \tag{2.8}$$

$$\text{s.t.} \quad \sum_{e \in \delta(S)} x_e \geq \varepsilon, \qquad \forall S \subset V, S \neq \emptyset, \tag{2.9}$$

$$z_{l(e)} \geq x_e, \qquad \forall e \in E, \tag{2.10}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L, \tag{2.11}$$

$$x_e \geq 0, \qquad \forall e \in E. \tag{2.12}$$

The objective function (2.8) minimizes the number of labels; with some arbitrary small real number ε, the exponential set of cut-based constraints (2.9) ensures the feasibility of the solution, where $\delta(S)$ denotes the edges of the cut set $[S, V \backslash S]$; constraints (2.10) bind label and edge variables; expressions (2.11) and (2.12) define their domains. The model can be further strengthened by inequalities (2.13).

$$\sum_{e \in \delta(v)} x_e \geq 1, \qquad \forall v \in V. \tag{2.13}$$

In their work, Chwatal and Raidl (2011) proposed a simple separation routine for inequalities (2.9): Given a solution for the linear relaxation of the model, select an arbitrary node and execute a depth-first search (DFS) procedure considering only edges e with $x_e \geq \varepsilon$. If the DFS is unable to reach all the vertices, then the cut is added to the model. The DFS procedure can be executed in $O(|V| + |E|)$ time, which is faster than the maximum flow algorithm used in the DCut formulation.

2.7 Strengthening the formulations

Furthermore, Chwatal and Raidl (2011) proposed additional inequalities to strengthen the formulations. The tree search constraints (2.14) and (2.15) fix the number of edges (arcs) by reinforcing the search for a valid spanning tree (arborescence) in the DCut and EC models, respectively.

$$\sum_{a \in A} y_a = |V| - 1. \tag{2.14}$$

$$\sum_{e \in E} x_e = |V| - 1. \tag{2.15}$$

The strong linkage constraints, namely (2.16) and (2.17), provide a more direct link between the label and edge (arcs) variables by replacing their respective inequalities (2.4) and (2.10). Note that the tree search and strong linkage constraints cannot coexist because this simultaneity could discard several valid solutions, including the optimal one. Moreover, the strong linkage cannot be used in the multi-labeling scenario when it is addressed by the function $l^m : E \to 2^L$. However, this restriction does not apply to the multigraph approach.

$$z_{l^a(a)} = y_a, \qquad \forall a \in A. \tag{2.16}$$

$$z_{l(e)} = x_e, \qquad \forall e \in E. \tag{2.17}$$

Let $l^s(E')$ be the set of labels represented by the set of edges $E' \subseteq E$. The node label inequalities (2.18) strengthen the DCut and EC models with respect to its linear relaxations (LR) by requiring at least one active label for every cut $[S, V \backslash S]$ with $|S| = 1$. Chwatal and Raidl (2011) showed that a vertex v is occasionally sufficiently connected in the LR, according to inequalities (2.3) and (2.13), with $S = \{v\}$, but the LR is infeasible with respect to the associated node label inequality.

$$\sum_{l \in l^s(\delta(v))} z_l \geq 1, \qquad \forall v \in V. \tag{2.18}$$

Part II

Theoretical, Exact, and Heuristic Approaches

The second part of this thesis presents the main contributions of this work with respect to both the MLSTP and the GMLSTP. We present new interesting theoretical results about edge-labeled graphs, such as label contraction operations and polyhedral studies. Further, we introduce a new mathematical formulation and a new MIP-based metaheuristic for these problems. Finally, we carry out computational experiments in order to evaluate the performance of the proposed methods in comparison with the best ones in the literature.

Chapter 3

Theoretical Aspects

In this Chapter we present new useful concepts, definitions and theoretical results regarding ELGs, the MLSTP, and the GMLSTP. First, we recall the definition of ELG and introduce some new notation. Then, we formalize the concepts of edge and label contraction on ELGs, as well as discuss the relation between label contraction and induction by labels. In the sequel, we address procedures that transform ELGs while keeping the optimality with respect to the GMLSTP. Finally, we propose bounds on the number of edges, vertices, and on the objective function for the GMLSTP.

Recall, from Definition 1.1, that an ELG $G = (V, E, L)$ is an undirected graph in which V is the set of vertices, E is the set of edges, and L is the set of labels. Further, let the function $l : E \to L$ represents the label associated with the edge e and the function $E(L') = \{e \in E \mid l(e) \in L'\}$, $L' \subseteq L$, be the set of edges that have the label in L'. Also, recall, from Definition 1.3, that $G[L'] = (V, E(L'), L')$ is the spanning subgraph of G induced by the set of edges $E(L')$.

3.1 Label Contraction

In this Section, we introduce the *label contraction* operation on ELGs and discuss some important characteristics of the contracted graphs. For the next definitions, let $G = (V, E, L)$ be an edge-labeled graph such that $k \in E$, and $i, j \in V$. Furthermore, let $\pi(k, i, j)$ be a function that changes each endpoint of the edge k to j if it is equal to i or keep it unchanged otherwise.

Definition 3.1. *The contraction of an edge $e = (v, w) \in E$, $v \neq w$, results in a new graph $G/e = (V', E', L)$, where $V' = V \backslash \{v\}$, $E' = \{\pi(k, v, w) \mid k \in E \backslash \{e\}\}$, and $l(\pi(k, v, w)) = l(k)$, $\forall k \in E'$. If $v = w$, the contraction of e results simply on its removal from E.*

Definition 3.2. *By extension, the label contraction, denoted by $G /\!\!/ L'$, for $L' \subseteq L$, is the operation that performs the contraction of every edge in $E(L')$ on the graph G.*

Figure 3.1 illustrates the concepts of edge and label contraction. Fig. 3.1(a) presents an small ELG G. Fig. 3.1(b) shows the graph G/e, the resulting graph after the contraction of the edge $e = (7,1)$. Fig. 3.1(c) presents the graph $G /\!\!/ \{D\}$, that is the graph resulting from G after the contraction of the label D.

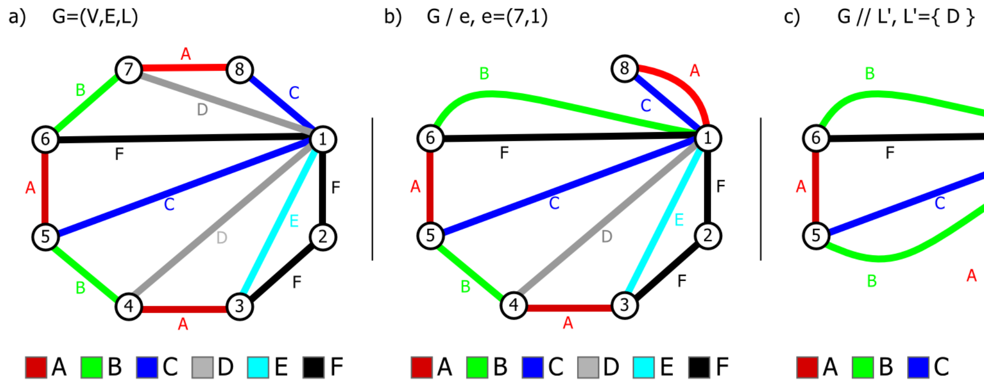


Figure 3.1: Illustration of the concepts of edge and label contraction. **(a)** the ELG $G = (V,E,L)$. **(b)** the graph G after the contraction of the edge $e = (7,1)$, and **(c)** the graph G after the contraction of the label D

Once given the concept of label contraction, we draw our attention to the relation between $G = (V,E,L)$, $G[L']$ and $G /\!\!/ L'$, for any $L' \subseteq L$. First let us suppose that $G[L']$ has w maximal connected components (for short components), each one spanning a set of vertices $S \subseteq V$. Let $W(G[L']) = \{S_1, S_2, \cdots, S_w\}$ represent the components of $G[L']$ such that S_i is the set of vertices spanned by the component i. It follows that:

- $H = G /\!\!/ L'$ has exactly w vertices, one for each component of $G[L']$. Let $v(S_i)$ be the vertex of H that was originated from the component S_i of $G[L']$. We can represent the set of vertices of H in relation to $G[L']$ as $\{v(S_1), v(S_2), \cdots, v(S_w)\}$.
- E' has one edge for each edge of E that has endpoints in different connected components of $G[L']$. Formally: Given $S_a, S_b \in W(G[L'])$, if $e = (v1, v2) \in E$ such that $v1 \in S_a$, $v2 \in S_b$, and $S_a \neq S_b$ then $e' = (v(S_a), v(S_b)) \in E'$.
- E' has one loop edge for each edge of E that has endpoints in the same connected component of $G[L']$. Formally: Given $S_a \in W(G[L'])$, if $e = (v1, v2) \in E$ such that $l(e) \notin L'$ and $v1, v2 \in S_a$, then the loop edge $e' = (v(S_a), v(S_a)) \in E'$.

Figure 3.2 illustrates the relation between G, $G[L']$ and $G \,/\!\!/\, L'$, for $L' = \{B, C\}$. Fig. 3.2a, 3.2b and 3.2d present, respectively, the example graph G, $G[L']$, and $G \,/\!\!/\, L'$. Fig. 3.2c highlights the components of $G[L']$ on G, evidencing that each component of $G[L']$ is related to a vertex on $G \,/\!\!/\, L'$, and each edge on $G \,/\!\!/\, L'$ is related to an edge on G.

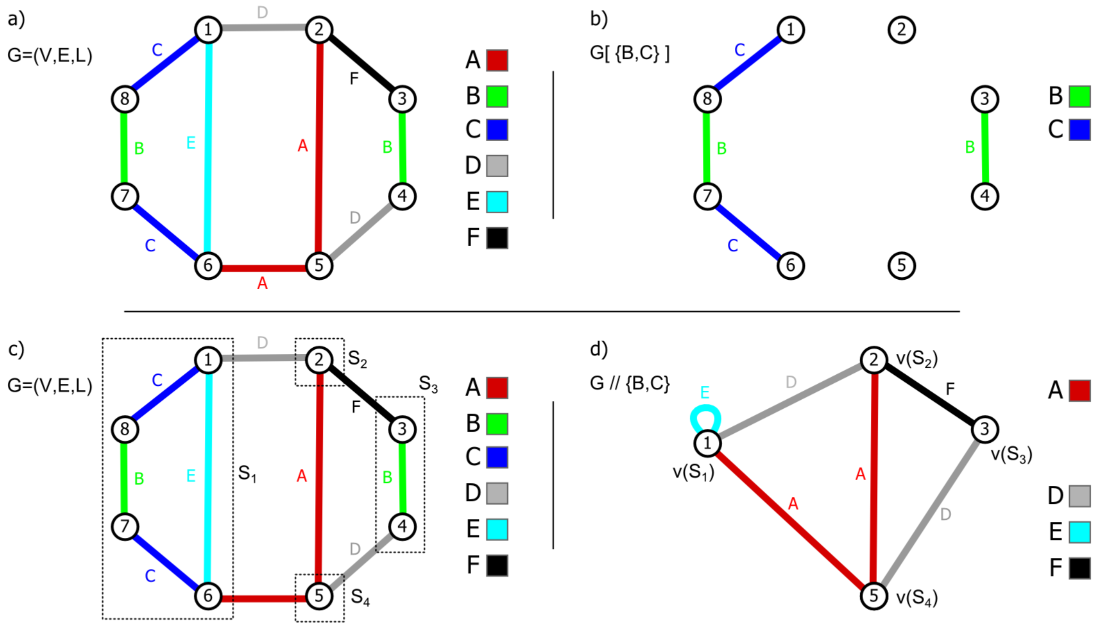


Figure 3.2: Relation between G, $G[L']$, and $G \,/\!\!/\, L'$, for $L' = \{B, C\}$. **(a)** The ELG G. **(b)** $G[L']$. **(c)** The set of components of $G[L']$ highlighted on G. **(d)** The graph $G \,/\!\!/\, L'$ with one vertex for each component of $G[L']$, one edge for each edge of G with endpoints in different components of $G[L']$, and one loop edge for each edge e of G with $l(e) \notin L'$ and endpoints in the same component of $G[L']$

In the sequel, we define a small variant of the MLSTP, namely the partial minimum labeling spanning tree problem (pMLSTP), and use it to prove an important relation between the MLSTP and the label contraction operation.

Definition 3.3. *Let $L' \subset L$ be a subset of the labels of G such that $G[L']$ is not connected, namely a partial solution for the MLSTP on G. The partial minimum labeling spanning tree problem consists in finding a set $L'' \subseteq L$ such that $G[L' \cup L'']$ is connected and $|L''|$ is minimized.*

Theorem 3.1. *Solving the pMLSTP for an ELG $G = (V, E, L)$ and a partial solution $L' \subset L$ is equivalent to solving the GMLSTP on $H = G \,/\!\!/\, L' = (V', E', L)$.*

Proof. *Let $C \subseteq L \backslash L'$. To prove this Theorem, we show that $G[L' \cup C]$ is connected if and only if $H[C]$ is connected.*

($\Rightarrow$) Suppose $G[L' \cup C]$ is connected and $H[C]$ is disconnected. In such case, $H[C]$ has an empty cut-set $[N, V' \backslash N]$, $N \subset V', N \neq \emptyset$. Further, let $N = \{v(S_1), v(S_2), \cdots, v(S_n)\}$ be the representation of this set in terms of $W(G[L'])$. Further, let $W' = \{S \mid v(S) \in N\}$, $W' \subset W(G[L'])$,

and $W'' = W(G[L'])\backslash W'$. Since $G[L' \cup C]$ is connected, there is an edge $e = (v_1, v_2) \in E$ such that $v_1 \in S_a$, $v_2 \in S_b$, $S_a \in W'$ and $S_b \in W''$. Hence $v(S_a) \in N$, $v(S_b) \in V'\backslash N$, and the edge $e' = (v(S_a), v(S_b)) \in [N, V'\backslash N]$.

($\Leftarrow$) Suppose $H[C]$ is connected and $G[L' \cup C]$ is disconnected. In this case, $G[L' \cup C]$ has an empty cut-set $[N, \overline{N}]$, $N \subset V$, $N \neq \emptyset$, $\overline{N} = V\backslash N$. Since $G[L' \cup C]$ is $G[L']$ plus the edges of $E(C)$, and given that $W(G[L']) = \{S_1, S_2, \cdots, S_w\}$, we have, without loss of generality, that $N = S_1 \cup S_2 \cup \cdots \cup S_n$ and $\overline{N} = S_{n+1} \cup S_{n+2} \cup \cdots \cup S_w$, $n < w$. However, given that $H[C]$ is connected, the cut-set $[\{v(S_1), v(S_2), \cdots, v(S_n)\}, \{v(S_{n+1}), v(S_{n+2}), \cdots, v(S_w)\}]$ is not empty and the edge $e' = (v(S_a), v(S_b))$, $1 \leq a \leq n$, $n < b \leq w$ belongs to it. In this case, $G[L' \cup C]$ has an edge $e = (v_a, v_b)$, $v_a \in N$, $v_b \in \overline{N}$, and $[N, \overline{N}]$ is not empty. □

Lastly, the Proposition 3.1 and the Definitions 3.4 and 3.5 address the relation between graphs obtained by the contraction of the same ELG but using distinct sets of labels. Let $G = (V, E, L)$ be an ELG, $M \subset E$, and $L', L'' \subseteq L$. Further, let $\Pi(e, M)$ be the resulting edge after applying $e \leftarrow \pi(e, v, w)$, for each $(v, w) \in M$.

Definition 3.4. *Let $G \mathbin{/\!/} L' = (V', E', L)$. The edge $e \in E$ is an originating edge of $e' \in E'$ if $\Pi(e, E(L')) = e'$ and $l(e) = l(\Pi(e, E(L')))$.*

Definition 3.5. *Let $G \mathbin{/\!/} L' = (V', E', L)$, and $G \mathbin{/\!/} L'' = (V'', E'', L)$. The edge $e'' \in E''$ is the projection of the edge $e' \in E'$ in the set of edges E'', denoted by $\xi(e', E'')$, if $e \in E$ is an originating edge of both e' and e''.*

Proposition 3.1. *Let $G \mathbin{/\!/} L' = (V', E', L)$, $G \mathbin{/\!/} L'' = (V'', E'', L)$, and $e' \in E'$. If $l(e') \notin L''$, the edge e' has a projection in the set E''.*

Demonstration. *Let $e \in E$ be an originating edge of $e' \in E'$. If $l(e') \in L''$, by Definitions 3.1 and 3.2, no edge with label $l(e')$ belongs to $G \mathbin{/\!/} L''$. On the other hand, if $l(e') \notin L''$, $e'' = \Pi(e, E(L'')) \in E''$, and, by definition, e is an originating edge of e''.* □

Figure 3.3 illustrates the Definitions 3.4 and 3.5. Fig. 3.3a, 3.3b and 3.3c present, respectively, the ELG $G = (V, E, L)$, $G' = (V', E', L') = G \mathbin{/\!/} \{D, F\}$, and $G'' = (V'', E'', L'') = G \mathbin{/\!/} \{B, C\}$. Consider the edges $e = (5, 6) \in E$, $e' = (1, 4) \in E'$, and $e'' = (1, 5) \in E''$. We have that e is the originating edge of both e' and e'', then $\xi(e', E'') = e''$ as well as $\xi(e'', E') = e'$.

3.2 Transformations on Edge-Labeled Graphs

In this Section we address some characteristics of the ELGs that allow us to transform it without losing the optimality with respect to the GMLSTP. These transformations lead to graphs with a

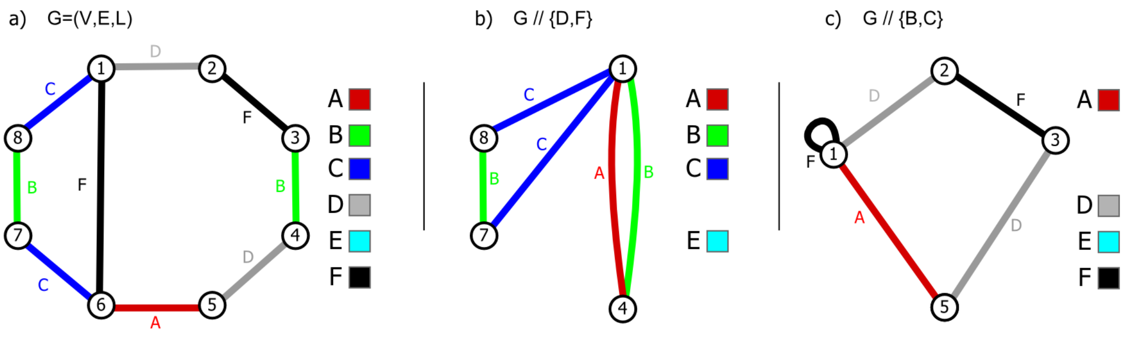


Figure 3.3: The concepts of originating edge and edge projection. **(a)** G, an input ELG. **(b)** $G' = G /\!\!/ \{D,F\}$. **(c)** $G'' = G /\!\!/ \{B,C\}$. The edge $e = (5,6)$ on G is the originating edge of the edges $e' = (1,4)$ on G' and $e'' = (1,5)$ on G''. Thence, e'' is a projection of e' and vice versa

smaller set of edges, vertices and/or labels, what can improve the efficiency of both heuristic and exact methods for the problem. Remark that Krumke and Wirth (1998) has briefly mentioned the results presented in Proposition 3.2 and Corollary 3.1, but here we give a more formal treatment for these results, as well as extend them. First we give the formal definitions of monochromatic cycles and cuts, as well as discuss how to deal with it.

Definition 3.6. *Given an ELG, a monochromatic cycle is a cycle formed solely by edges with the same label.*

Proposition 3.2. *(Krumke and Wirth, 1998): Without loss of optimality, any monochromatic cycle can be broken by arbitrarily choosing an edge and removing it from the graph.*

Proof. *Let $G(V,E,L)$ be an edge-labeled graph, and $P = \{e_1 = (v_1,v_2),\ e_2 = (v_2,v_3),\ \cdots,\ e_n = (v_n,v_1)\}$, such that $l(i) = k$, $\forall\, i \in P$, a monochromatic cycle of G. Without loss of generality, let $e = e_1 = (v_1,v_2)$ be an arbitrary edge of P, let $G' = (V, E\backslash\{e\}, L)$ be the resulting graph after the removal of e from G, and let C be any subset of L. We prove this proposition by demonstrating that $G[C]$ is connected if and only if $G'[C]$ is connected.*

If $k \notin C$ then this statement holds. Indeed, $G[C]$ is exactly $G'[C]$.

($\Rightarrow$) For any $C \subseteq L$, $k \in C$, suppose $G[C]$ is connected and $G'[C]$ is disconnected. In this case $G'[C]$ has exactly two disjoint connected components, each one spanning respectively the set of vertices $S, \overline{S} \subset V$, such that $v_1 \in S$ and $v_2 \in \overline{S}$. However, the path $P\backslash\{e\}$ connects v_2 to v_1, and hence S to $\overline{S}$. Then $G'[C]$ is connected.

($\Leftarrow$) For any $C \subseteq L$, $k \in C$, suppose $G'[C]$ is connected and $G[C]$ is disconnected. $G[C]$ is exactly the graph $G'[C]$ plus the edge e. The addition of an edge cannot disconnect the graph $G[C]$. $\square$

Based on Proposition 3.2, without loss of optimality for the GMLSTP, we can preprocess the input graph by breaking all of its monochromatic cycles. Therefore, with regards to the GMLSTP, we can deal just with *monochromatic-cycle-free graphs* (i.e. a graph G such that $G[\{l\}]$ does not have cycles, for any $l \in L$). Let $k = |L|$, $n = |V|$, and $m = |E|$. Krumke and Wirth (1998) have proposed a monochromatic cycles removal method that can be executed in $O(k \cdot n + m)$. As an alternative, we propose a new monochromatic cycles removal procedure (MCR) that can be carried out in $O(\alpha(m,n) \cdot m)$, where α stands for the inverse of the Ackerman's function.

Algorithm 3.1 describes the procedure MCR, which uses disjoint sets data structures (Cormen et al., 2009). The initializatoins of the data structures are performed in lines 2, 3, 5, and 6. The main loop of line 4 iterates over each label of the input ELG, while the loop of the lines 7 to 10 iterates on the set of edges with the label l, removing the cycle ones. Lastly, a monochromatic-cycle-free graph is returned in line 13. The initialization of the structures takes $O(n)$. The internal loop performs at most 3 *Union-Find* operations for each edge of the graph, while keeping the history of the changes on the *Union-Find* structure S allows to restore it with the same number of operations that have changed it. Thence, the complexity of the MCR is $O(n + \alpha(m,n) \cdot m)$, and since the input ELG is connected, $m \geq n - 1$, and the complexity of the MCR is $O(\alpha(m,n) \cdot m)$.

Algorithm 3.1: The monochromatic cycles removal procedure (MCR).

```
1  procedure MCR($G = (V,E,L)$)
2      Let $E'' \leftarrow \emptyset$ be the resulting set of edges;
3      Let $S \leftarrow Initialize(S)$ represent the Union-Find data structures;
4      foreach $l \in L$ do
5          Let $E^l \leftarrow \emptyset$ be an auxiliary set of edges;
6          Let $H \leftarrow \emptyset$ carry the history of changes on $S$;
7          foreach $e = (v_1, v_2) \in E(\{l\})$ do
8              if $Find(v_1) \neq Find(v_2)$ then
9                  $E^l \leftarrow E^l \cup \{e\}$;
10                 $H \leftarrow Union(v_1, v_2)$;
11         $S \leftarrow Restore\text{-}Union\text{-}Find\text{-}DS(S,H)$;
12         $E'' \leftarrow E'' \cup E^l$;
13     return $G(V,E'',L)$;
```

Figure 3.4 illustrates the concept of monochromatic cycles. Fig 3.4(a) presents an example ELG G, which have two monochromatic cycles: the cycle $1 \cdot 6 \cdot 8 \cdot 1$ with label F and the cycle $2 \cdot 3 \cdot 4 \cdot 5 \cdot 2$ with label C. Fig. 3.4(b) shows the graph G after breaking the cycle with label F, while Fig. 3.4(c) shows G after breaking the cycle with label C. Note that the graph of Fig. 3.4(c) is a monochromatic-cycles-free graph.

Definition 3.7. *Given an ELG $G = (V,E,L)$, and a set of vertices $S \subset V$, the cut-set $[S, V \backslash S]$ is a monochromatic cut if $l(e) = k$, $\forall e \in [S, V \backslash S]$.*

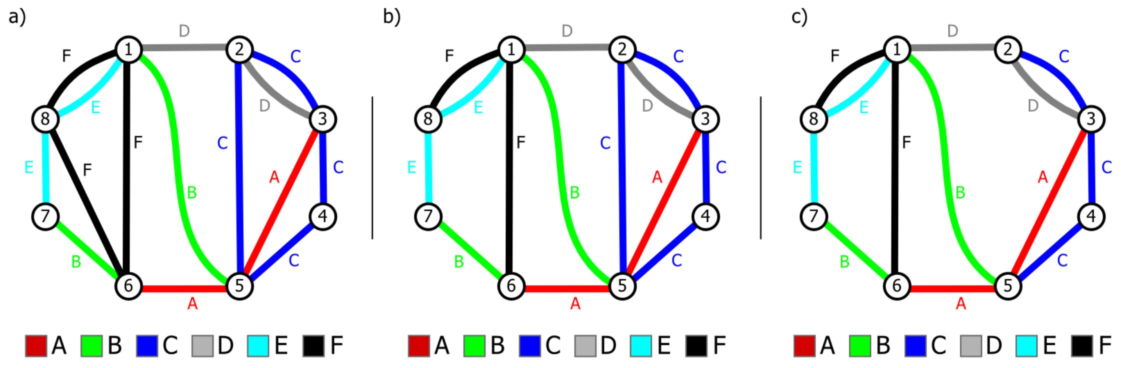


Figure 3.4: Illustration of the concept of monochromatic cycles. **(a)** en ELG G with two monochromatic cycles. **(b)** the graph G after the removal of the cycle with the label F. (c) a monochromatic-cycles-free graph

Proposition 3.3. *For solving the GMLSTP it is possible to deal only with monochromatic-cuts-free graphs.*

Demonstration. *From Definition 3.7, if G has a monochromatic cut with the label k, this label belongs to every feasible solution for the problem. In this case, from Theorem 3.1, we have that solving the GMLSTP for G* $/\!\!/ \{k\}$ *is equivalent to solving the problem for G and we can deal only with monochromatic-cuts-free graphs.* □

Figure 3.5 illustrates the concept of monochromatic cuts. Fig 3.5(a) presents an ELG G with a monochromatic cut $[S, V\backslash S]$, for $S = \{3,4,5\}$, that has only edges with the label A. Fig 3.5(b) highlights the fact that the removal of all the edges with the label A from G leads to a disconnected graph. Fig 3.5(c) shows the graph G after the contraction of the label A.

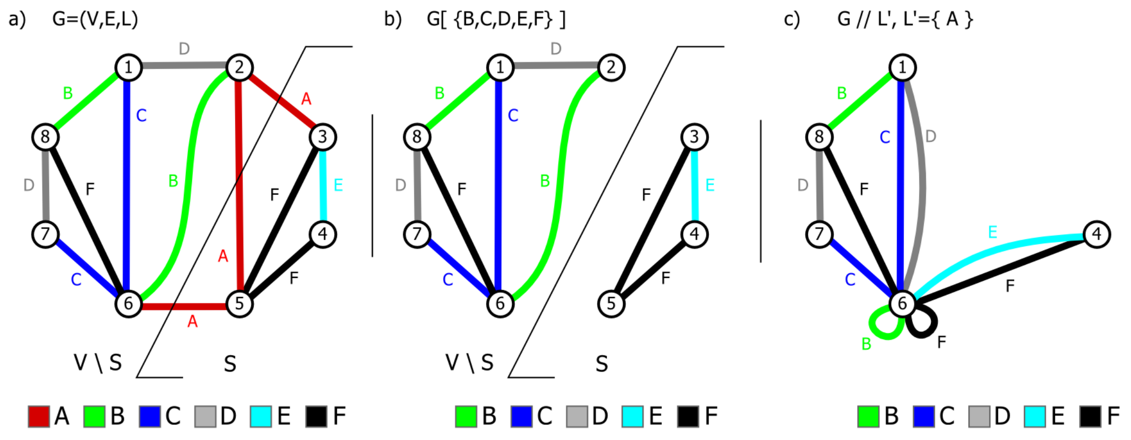


Figure 3.5: Illustration of the concept of monochromatic cuts. **(a)** The ELG G with the monochromatic cut $[S, V\backslash S]$, for $S = \{3,4,5\}$. **(b)** the disconnected graph $G[\{B,C,D,E,F\}]$. **(c)** the graph $G /\!\!/ \{A\}$

In the sequel we give two new useful definitions on ELGs: the *transitive closure* and the *chromatic closure*. Then, these concepts are used to extend the Proposition 3.2, giving a new way to represent ELGs. And finally, we prove a dominance rule on instances for the GMLSTP.

Definition 3.8. *The transitive closure of a label $l \in L$, denoted by $\mathcal{F}^c(l)$, stands for the expansion of the set of edges $E(\{l\})$ such that if there is a path between the vertices u and v in the graph $G[\{l\}]$, then the edge $e = (u,v) \in \mathcal{F}^c(l)$.*

Definition 3.9. *By extension, the chromatic transitive closure, for short chromatic closure, of an ELG $G = (V,E,L)$, denoted by $\mathcal{F}(G)$, is the graph $H = (V,E',L)$, which has the same sets of vertices and labels of G, and the set of edges defined by the union of the transitive closures of each label of G. Formally, $E' = \bigcup_{l \in L} \mathcal{F}^c(l)$.*

Figure 3.6 illustrates the concepts of transitive and chromatic closures. Figure 3.6(a) presents a small ELG $G = (V,E,L)$. Fig. 3.6(b) highlights $\mathcal{F}^c(F)$, the transitive closure of the label $F \in L$, and the Fig. 3.6(c) shows $\mathcal{F}(G)$, the chromatic closure of the graph G.

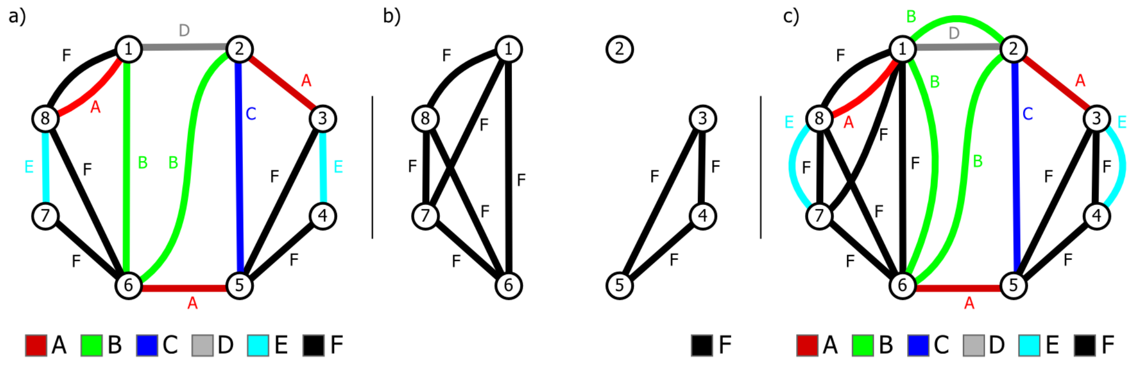


Figure 3.6: Example of transitive and chromatic closures. **(a)** The edge-labeled graph $G = (V,E,L)$. **(b)** The transitive closure of the label F. (c) the chromatic closure of the graph G.

Proposition 3.4. *Solving the GMLSTP for an ELG G is equivalent to solving the problem for the graph $H = \mathcal{F}(G)$.*

Proof. *Let C be any subset of L. We prove this proposition by demonstrating that $H[C] = (V,E',L)$ is connected if and only if $G[C] = (V,E,L)$ is connected.*

($\Rightarrow$) Suppose $H[C]$ is connected and $G[C]$ is disconnected. In this case $G[C]$ has two disjoint set of vertices $S, \overline{S} \subset V$, such that $u \in S$, $v \in \overline{S}$, $e' = (u,v) \in E'$, but there is no path between u and v in $G[C]$. However, from Definition 3.9, if $e' = (u,v) \in E'$, then there is a path between u and v in $G[C]$, a contradiction

($\Leftarrow$) Suppose $G[C]$ is connected and $H[C]$ is disconnected. Since the transitive closure is an extension of the set of edges of G, $H[C]$ is the graph $G[C]$ with additional edges. The addition of edges cannot disconnect the graph $G[C]$. $\square$

Based on Proposition 3.4 and on the concept of transitive closure, we can state a dominance rule on the labels of an ELG with respect to the GMLSTP.

Definition 3.10. *Let $G = (V,E,L)$ be an ELG such that $l,k \in L$. The label k is dominated by the label l if $\mathcal{F}^c(k) \subseteq \mathcal{F}^c(l)$.*

Proposition 3.5. *Let G be an ELG such that the label k is dominated by the label l. The label k can be removed from the set G without loss of optimality for the GMLSTP.*

Demonstration. *This demonstration uses the Proposition 3.4 and the graph $H = \mathcal{F}(G) = (V,E,L)$. Let L^* be an optimal solution for the GMLSTP on H. If $k \notin L^*$, then the proposition holds. If $k \in L^*$ and $l \notin L^*$, then the solution $L' = (L^* \backslash \{k\}) \cup \{l\}$ is also optimal. Indeed, $|L^*| = |L'|$ and $H[L']$ is connected since $E(\{k\}) \subseteq E(\{l\})$. Finally, if $k,l \in L^*$, then L^* is not optimal because the graph $H[L^* \backslash \{k\}]$ is connected. In fact, for every edge removed from $H[L^*]$, there is an parallel edge in $E(\{l\})$.* □

Figure 3.7 illustrates the concept of dominance of labels. Fig. 3.7(a) presents the ELG $G = (V,E,L)$. Fig. 3.7(b) shows the graph $\mathcal{F}(G)$, highlighting the dominance of the labels F and B over the labels E and D, respectively; Fig. 3.7(c) shows the graph G after the removal of the dominated labels;

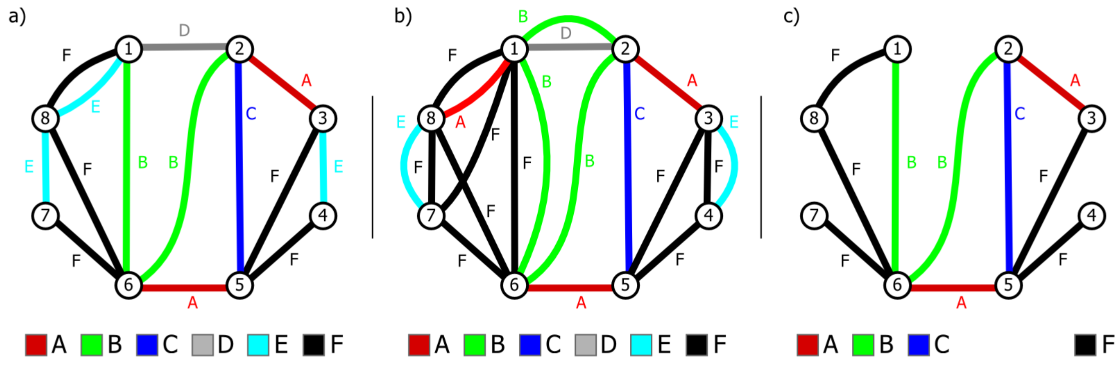


Figure 3.7: The concept of dominance of a label. **(a)** An input ELG $G = (V,E,L)$. **(b)** The graph $\mathcal{F}(G)$. **(c)** The graph G after the removal of the dominated labels, D and E

Lastly, combining Propositions 3.4 and 3.2 we can modify further the input graph. Let G be the input ELG. From Proposition 3.4, we have that we can solve the GMLSTP on $H = \mathcal{F}(G)$ without loss of optimality. Moreover, from Proposition 3.2, we can arbitrarily break the cycles of H. In such case, since $G[\{l\}]$ keeps the same connected components, we are allowed to make any changes on the set of edges of $E[\{l\}]$, $\forall l \in L$. This transformation allows us to represent the input ELG in the way that best suits the resolution of the problem. Another possibility is to store just information on the connected components of $G[\{l\}]$, $\forall l \in L$, instead of edges, in order to save storage/memory space.

Figure 3.8 Illustrates two convenient modifications on the edges of an input ELG. Figure 3.8(a) shows the input ELG $G = (V,E,L)$. In the Fig. 3.8(b), all the connected components of $G[\{l\}]$, $\forall l \in L$, are paths, while in Fig. 3.8(c), all the connected components of $G[\{l\}]$, $\forall l \in L$, are stars.

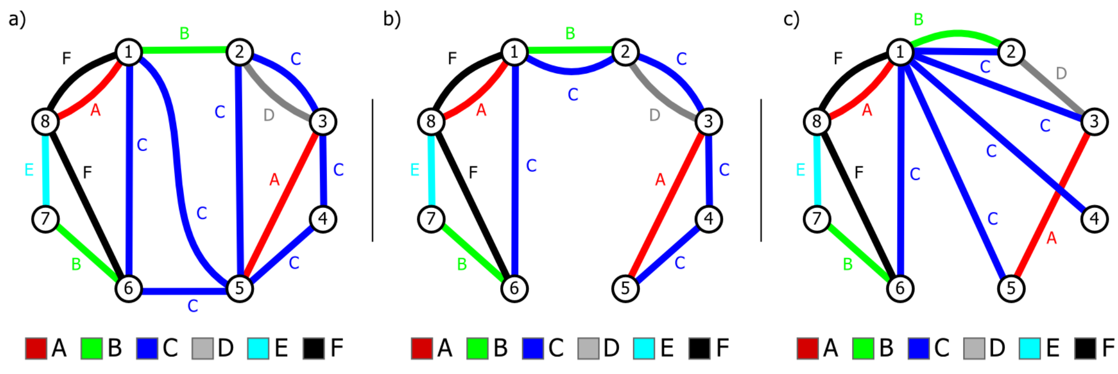


Figure 3.8: Illustration of transformations on the set of edges of ELGs. **(a)** The ELG G. **(b)** Each monochromatic connected component of G represented as a path, and **(c)** as a star

3.3 Bounds for the GMLSTP

In this Section we introduce some useful bounds for the GMLSTP. First, we present two upper bounds on the number of edges that are necessary to consider. In the sequel, we discuss the minimum number of vertices for an instance to be non-trivial, and, finally, we discuss some lower bounds on the objective function of the GMLSTP. From Proposition 3.2 we can derive directly the following corollary:

Corollary 3.1. *(Krumke and Wirth, 1998): With respect to the GMLSTP, the maximum number of edges with the same label that is necessary to consider is* $|V|-1$. *Formally,* $|E(\{l\})| \leq |V|-1$, $\forall l \in L$.

Furthermore, based on Corollary 3.1 and on the definition of graph induced by a set of labels, we can state the following results:

Remark. *Let* $G=(V,E,L)$ *be a monochromatic cycles free graph. The number of connected components of* $G[\{l\}]$ *is* $|V|-|E(\{l\})|$. *Indeed, since* G *is monochromatic cycles free, each edge of* $E(\{l\})$ *reduces the number of connected components of* $G[\{l\}]$ *by* 1. $\square$

In the sequel, we discuss that the bound given by Corollary 3.1 can be further improved if considering only non-trivial instances of the GMLSTP, as defined bellow.

Proposition 3.6. *Let $l \in L$ be the label with the greater $|E(\{l\})|$ in a monochromatic cycles free graph. $|E(\{l\})| \leq |V| - 3$ for non-trivial instances.*

Demonstration. *From Corollary 3.1, $|E(\{l\})| \leq |V| - 1$. Follow two trivial cases:*

$|\mathbf{E}(\{\mathbf{l}\})| = |\mathbf{V}| - \mathbf{1}$*: the optimal solution is $\{l\}$ and it is found trivially by the MCR (Corollary 3.2).*

$|\mathbf{E}(\{\mathbf{l}\})| = |\mathbf{V}| - \mathbf{2}$*: $G[\{l\}]$ has two connected components spanning the set of vertices S and $\overline{S} = V \backslash S$, respectively. Since the input graph is connected, the cut-set $[S, \overline{S}]$ is not empty and the solution is $\{l, l(e)\}$, $e \in [S, \overline{S}]$.* □

Next, we show that is easy to solve the GMLSTP if the number of vertices of the input graph is small.

Proposition 3.7. *Let $G = (V, E, L)$ be a connected monochromatic-cycles-free ELG. The GMLSTP is trivially solvable if $|V| < 5$.*

Demonstration. *The cases are the following:*

$|\mathbf{V}| = \mathbf{1}$*: the solution is $\emptyset$.*

$|\mathbf{V}| = \mathbf{2}$*: the solution is $\{l\}$, for any $l \in L$.*

$|\mathbf{V}| = \mathbf{3}$*: if some label l has two or more edges, the solution is $\{l\}$. Otherwise, take arbitrarily a spanning tree of G and the solution is its set of labels, which have cardinality* 2.

$|\mathbf{V}| = \mathbf{4}$*: if some label l has three or more edges, the solution is $\{l\}$. If some label l has two or more edges, add l to the solution and complete the spanning tree of G arbitrarily with the edge e, the solution is $\{l, l(e)\}$. Otherwise, take arbitrarily a spanning tree of G and the solution is its set of labels, which have cardinality* 3. □

Note that both Propositions 3.6 and 3.7 can be used to terminate earlier any heuristic method, such as the MVCA. In the sequel of this section, we discuss some lower bounds on the objective function of the GMLSTP.

Proposition 3.8. *Let $G = (V, E, L)$ be a monochromatic cycles free graph, and let $L = \{l_1, l_2, \cdots, l_{|L|}\}$ be the set of labels of G such that $|E(\{l_i\})| \geq |E(\{l_{i+1}\})|$, $\forall\ 1 \leq i \leq |L| - 1$. Let b^* be the minimum $b \in \mathbb{N}$ such that $\sum_{i=1}^{b} |E(\{l_b\})| \geq |V| - 1$. b^* is a lower bound for the MLSTP on G.*

Demonstration. *Suppose L' is a solution for the MLSTP and $|L'| < b^*$. In such case, $G[L'] = (V, E', L')$ has $|E| < |V| - 1$ edges. But the minimum number of edges for a graph to be connected is $|V| - 1$.* □

In other words, the minimum number of labels necessary to connect a monochromatic cycles free graph $G = (V, E, L)$ is at least the cardinality of the minimum subset of $L' \subseteq L$ such that $G[L']$ has $|V| - 1$ or more edges. The following corollaries are useful for speeding up the computational experiments performed both in Chapters 4 and 7.

Corollary 3.2. *If $b^* = 1$, it is the optimal solution for the problem, and this solution is found trivially by the MCR procedure (Algorithm 3.1).*

Corollary 3.3. *After the MCR procedure, if any method finds a solution with value 2, this solution is optimal.*

Another lower bound for the GMLSTP can be obtained if we relax the connectivity constraint of the problem. The resulting problem, namely the label covering problem (LCP), can be defined as follows:

Definition 3.11. *Let $G = (V, E, L)$ be an ELG, and let $d(v)$ denote the degree of the vertex $v \in V$. The label covering problem aims to find a set $L' \subseteq L$ such that $d(v) \geq 1$, for all vertices of $G[L']$.*

In this sense, it follows that the optimal solution for the MLSTP could never be small than the optimal solution for the LCP.

Finally, let $G = (V, E, L)$ be an ELG such that $u, v \in V$. If the path from u to v with the minimum number of labels uses $|L'|$, $L' \subseteq L$ labels, the solution for the GMLSTP on G is not lesser than $|L'|$. This lower bound can be extended by testing all possible pairs u, v. The problem of finding the minimum number of labels to connect two vertices on a ELG is better discussed in the Chapter 8.

Chapter 4

MIP-Based Exact Methods

In this chapter, we describe a new mathematical formulation for the GMLSTP, namely CCut. The proposed model is based on the concept of *colorful cuts*; The absence of edges, arcs, and flow variables is the main difference between CCut and previous mathematical formulations for the problem. In remaining of this chapter we introduce the CCut formulation and describe branch-and-cut strategies for solving the model. Finally, we compare the proposed methods with the best ones in the literature.

4.1 The colorful cuts formulation

This Section presents the colorful cuts formulation (CCut), a new cut-based mathematical model for solving both the MLSTP and the GMLSTP. One can say that the CCut formulation is compact because it defines only $|L|$ binary variables. Notwithstanding, it has an exponential number of constraints. Before introducing the formulation, it is necessary to formalize the concepts of *disconnecting set of labels* and *colorful cuts*, as well as to state an important property of these cuts.

Definition 4.1. *The set $K \subseteq L$ is a disconnecting set of labels if the number of connected components of the graph $G[L \backslash K]$ is greater than the number of connected components of G.*

Property 4.1. *Let $\delta(S)$ be the set of edges of the cut set $[S, V \backslash S]$, and let $l^s(\delta(S)) = \{l(e) \mid e \in \delta(S)\}$ be the set of labels represented by the edges of $\delta(S)$. It is apparent that $l^s(\delta(S))$ is a disconnecting set of labels for every $S \subset V$, $S \neq \emptyset$.*

Definition 4.2. *The disconnecting set $K(S) \subseteq L$ is a colorful cut if it is derived from a cut set $\delta(S)$ such that $K(S) = l^s(\delta(S))$ for some $S \subset V$, $S \neq \emptyset$.*

Figure 4.1 illustrates the concept of colorful cuts. Fig. 4.1a presents a small edge-labeled graph G. In Fig. 4.1b the subset of vertices $S = \{1,7,8\}$ is highlighted. Derived from S we have the cut set $\delta(S) = \{(1,2),(1,5),(6,7)\}$, and the colorful cut $K(S) = \{C,E\}$, that is the set of labels represented in $\delta(S)$. Fig. 4.1c reinforces that removing $K(S)$ from G leads to a disconnected graph.

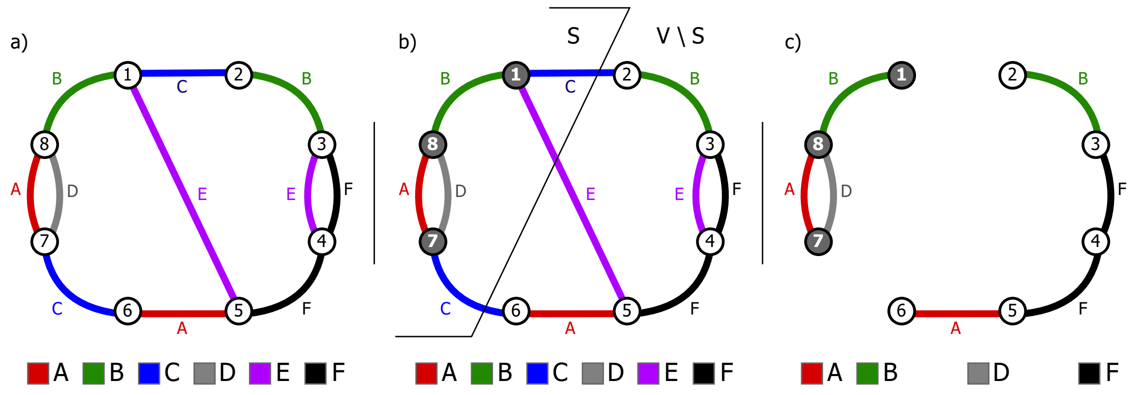


Figure 4.1: Example of colorful cut. **(a)** An edge-labeled graph G. **(b)** The set $S = \{1,7,8\}$, the cut set $\delta(S) = \{(1,2),(1,5),(6,7)\}$, and the colorful cut $K(S) = \{C,E\}$. **(c)** The graph $G[L\backslash K(S)]$, that is disconnected

Proposition 4.1. *A graph $G = (V,E,L)$ is connected if and only if any colorful cut $K(S)$ of G is not empty.*

Proof. *($\Rightarrow$) By way of contradiction, suppose that G is connected and there exists a colorful cut $K(S) = \emptyset$. In this case, $\delta(S) = \emptyset$, and there is no path between the vertices of S and $V\backslash S$.*

($\Leftarrow$) Suppose not. Suppose that any colorful cut $K(S)$ of G is not empty and G is not connected. If G is not connected, it has one maximal connected component $S \subset V$, $S \neq \emptyset$. However, if $K(S) \neq \emptyset$, then $\delta(S) \neq \emptyset$, and there is an edge from S to a vertex $v \in V\backslash S$. Hence, $S \cup \{v\}$ is a connected component, and S is not maximal. □

The CCut formulation is derived directly from Proposition 4.1; it is presented in the program (4.1) through (4.3). The model defines only the group of binary variables $z_l \in \{0,1\}$, for which $z_l = 1$ means that every edge with the label l is in the solution.

$$\text{Minimize} \sum_{l \in L} z_l \tag{4.1}$$

$$\text{s.t.} \quad \sum_{l \in K(S)} z_l \geq 1, \qquad \forall S \subset V, S \neq \emptyset, \tag{4.2}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{4.3}$$

The objective function (4.1) minimizes the number of labels; the exponential set of constraints (4.2) ensures the connectivity of the solution graph by requiring at least one active label for every colorful cut of the graph. Notice that such constraints can be separated heuristicly either by the maximum flow algorithm proposed for the DCut formulation, or by the DFS procedure proposed for the EC formulation. Finally, the set of constraints (4.3) defines the domain of the variables.

The CCut formulation can be further strengthened by inequality (4.4), which is derived from Proposition 3.8. It mimics the tree search constraints (2.14) and (2.15), described respectively for the DCut and EC formulations. Recall that $E(\{l\}) = \{e \in E \mid l(e) = l\}$; the constraint requires a minimum number of edges (corresponding to a tree) to connect the graph.

$$\sum_{l \in L} |E(\{l\})| \cdot z_l \geq |V| - 1. \tag{4.4}$$

Remark that solving the CCut model for medium to large size input graphs is not practical due to the exponential set of inequalities (4.2). For this reason, the constraints (4.2) are replaced initially by the set (4.5), which grants every vertex of the graph has an incident edge. Then, the remaining constraints can be added on demand to the model, as discussed better in Section 4.2.

$$\sum_{l \in K(\{v\})} z_l \geq 1, \qquad \forall v \in V. \tag{4.5}$$

In the following sections we propose branch-and-cut algorithms for solving the CCut formulation and present computational experiments comparing these methods with the best ones in the literature.

4.2 Branch-and-cut algorithms

As introduced previously, solving the CCut formulation for medium to large size input ELGs is not practical due to the size of the exponential set of inequalities (4.2). In such case, it is interesting to use only a subset of these constraints at beginning, namely the constraints (4.5), and add the remaining ones to the model on demand. In this sense, we present in this section three branch-and-cut algorithms we propose for solving the CCut formulation. Refer to Wolsey (1998) or Wolsey and Nemhauser (2014) for more information on branch-and-cut algorithms.

First, let z_l^* be the value of the variable z_l in a given solution for the linear relaxation of the CCut model. We separate the colorful cuts inequalities (4.2) by using a simple DFS procedure, as proposed by Chwatal and Raidl (2011) for the EC formulation (refer to Section 2.6): starting from an arbitrary root node, execute a depth-first search considering only the edges e with $z_{l(e)}^* > 0$; let S be the set of vertices reached by the DFS procedure; if $S \neq V$, then the colorful cut inequality derived from S is added to the model. Observe that this is a heuristic procedure, if carried out on fractional solutions, but it is exact in case of integer solutions. In this case, running the DFS separation for every integer solution ensures the corretude of the model.

Alternatively, it is also possible to separate the colorful cuts inequalities (4.2) by using the maximum flow algorithm, as proposed by Chwatal and Raidl (2011) for the DCut formulation (refer to Section 2.5). However, recall the DFS procedure can be executed in $O(|V|+|E|)$ time, which is faster than the maximum flow algorithm. Besides, observe that both separation routines discussed are only heuristics for fractional solutions. To separate exactly the colorful cuts inequalities, it is necessary to solve a weighted version of the minimum labeling global cut problem, which is discussed in Chapter 9.

In the sequel, we describe the branch-and-cut algorithms BC^A, BC^I, and BC^R. The only difference between the proposed algorithms is the moment they call the separation procedure. In BC^A, the separation procedure is called whenever a solution is found. In BC^I, the separation procedure is called whenever an integer solution is found. Further, in BC^R, the separation routine is carried out on the root node of the branch-and-bound tree until it is not possible to add more cuts, and for the rest of the branch-and-bound tree, the separation procedure only is called when an integer solution is found. Lastly, let $BC^X(F,P)$ denote when the algorithm BC^X is being used to solve the model F using the separation procedure P, and, analogously, let $LR(F,P)$ denote the linear relaxation of the model F using the separation procedure P.

4.3 Computational experiments

This section reports the computational experiments performed in order to evaluate the quality of the CCut formulation along with the proposed branch-and-cut algorithms. First, we study briefly the linear relaxation of CCut and evaluate the algorithms $BC^A(CCut,DFS)$, $BC^R(CCut,DFS)$, and $BC^I(CCut,DFS)$. Then, we compare the one with the best results with the best methods in the literature, namely EC and DCut.

All experiments reported in this chapter were implemented in C++ language and compiled by using g++ 4.6.3 with the optimization flag -O3. The CCut formulation and all of its derived procedures were implemented using the Concert library and Cplex 12.51 as the solver. The experiments were performed on a computer with Intel(R) Core(TM) i7-4790K CPU, 3.4GHz, 16 GB of RAM, and Ubuntu 14.04 as the operating system. Although the processor of this device has more than one core, the algorithms were executed using a single core and a single thread. Further, we turned off all presolve features and all automatic cutting-plane generation procedures while all other parameters of the Cplex were set to their respective default values. We treated the colorful cuts inequalities (4.2) with dynamically generated cutting-planes added as *User Cut Callbacks*, and implemented a *Lazy Constraint Callback* to reject disconnected graphs eventually found as feasible integer solutions of the incomplete model.

We have considered the group 2 of ELGs generated by Cerulli et al. (2005), a benchmark already consolidated in the literature. The group of input graphs used in these experiments has instances with number of vertices $n = |V| \in \{100, 200\}$, number of labels $l = |L| \in \{n/4,\ n/2,\ n,\ 5n/4\}$, and edge densities $d \in \{ld = 0.2, md = 0.5, hd = 0.8\}$. Also, each dataset consists in 10 different graphs for each *n-d-l* configuration, totalizing 240 ELGs.

The first experiment performed aimed to evaluate the impact, on the linear relaxation of CCut, of changing the colorful cuts inequalities (4.2) by the set of inequalities (4.5). To do so, we have executed the procedures $LR(CCut,\emptyset)$ and $LR(CCut,PART_2)$ for the 240 ELGs of the benchmark, where $PART_2$ represents the formulation with the same name (presented in Section 9.1) being used to solve the weighted version of the minimum labeling global cut problem and, thus, separate exactly the colorful cuts inequalities (4.2).

The results of the first experiment are presented in Table 4.1. Each line of the table reports the results for one input ELG identified in the form *n-d-l-i*, where $n = |V|$, $l = |L|$, d is the density of the graph, and $i \in \{0 \cdots 9\}$ is the number of the instance on its dataset. The first column identifies the input instance. The next two columns report the results obtained by each procedure, while the column Δ reports the difference between them. The column % presents the relative difference between the methods. The column *Cuts* shows the number of colorful cuts added to the model. Lastly, the column *t(s)* reports the time taken to solve $LR(CCut,PART_2)$. Observe that the separation procedure did not find cuts for 231 out of 240 input instances, and, for this reason, these results are omitted from the table.

From the results reported in Table 4.1, we can see that changing the colorful cuts inequalities (4.2) by the set of inequalities (4.5) does not cause much impact on the linear relaxation of CCut. Indeed, the exact separation procedure did not find cuts for the majority of the input graphs. Besides, even when the separation is able to add cuts into the model, its impact is not much relevant. In this sense, we have opted to use the DFS routine described previously as separation procedure for the colorful cuts inequalities due to its simplicity and fast running times.

Table 4.1: Effectiveness of separing exactly the colorful cuts inequalities

Instance	$LR(CCut,\emptyset)$	$LR(CCut,PART_2)$	**Cuts**	Δ	**%**	**t(s)**
100-ld-100-4	8.45870	8.47738	1	0.01868	0.22%	0.613
100-ld-100-9	5.53541	5.53549	1	0.00008	0.00%	3.432
100-ld-125-5	6.45614	6.45620	1	0.00006	0.00%	0.891
100-ld-125-6	7.72405	7.73515	4	0.01110	0.14%	1.533
100-md-25-0	1.30000	1.30508	1	0.00508	0.39%	0.234
200-hd-50-1	1.07843	1.07849	1	0.00006	0.01%	3.253
200-hd-50-2	1.07059	1.07066	1	0.00007	0.01%	6.151
200-hd-50-4	1.07110	1.07234	1	0.00124	0.12%	3.153
200-hd-50-6	1.08063	1.08071	1	0.00008	0.01%	5.292

The second experiment aims to evaluate the branch-and-cut algorithms $BC^A = BC^A(CCut,DFS)$, $BC^R = BC^R(CCut,DFS)$, and $BC^I = BC^I(CCut,DFS)$. To do so, we have carried out each procedure for the first instance of each dataset with $n = 200$. The results are reported in Table 4.2. Each line of the table reports the execution of one algorithm for one input ELG. The first column identifies the method while the second one identifies the input instance, using the same format from Table 4.1. The next three columns, *UB*, *LR*, and *LB*, report, respectively, the values of the upper bound, linear relaxation, and lower bound found by the methods. The columns *t(s)* and *lrt(s)* report, respectively, the cpu-time[1] spent by each method to solve the instance and its linear relaxation. The column *Nodes* presents the number of nodes solved in the branch-and-bound tree, and the column *Cuts* shows the number of colorful cuts added to the model. Lastly, the column *Gapr* reports the relative difference between the *LR* and the *UB*: $Gapr = (UB - LR)/UB$. Further, bold results evidence when one method outperformed all competitors.

From the results reported in Table 4.2 it is possible to observe that all methods were able to solve all input instances to optimality in less than one hour. Despite of that, the method BC^A performed too slow in relation to the other two. Comparing the methods BC^R and BC^I we have that the latter outperformed the first in 8 input instances, against 2 in the opposite direction. As

[1]All reported times are in seconds.

expected, the number of cuts added to the model is very small, and there is no impact on the linear relaxations. Surprisingly, the number of nodes visited by BC^I is smaller in comparison with BC^A. It happened because BC^I is able to find good upper bounds faster than BC^A. We have also performed similar experiments comparing the algorithms BC^A, BC^R, and BC^I for the formulations EC, DCut, and its variations (adding the strengthening inequalities described in Section 2.7). BC^I achieved the best performance in all experiments.

According to the experiments performed by Chwatal and Raidl (2011), the best exact methods for the MLSTP in the literature are the branch-and-cut procedures based on the formulations EC^{sn} (the program 2.8 to 2.12 with the strengthening inequalities 2.17 and 2.18) and DCut^{sn} (the program 2.2 to 2.7 with the strengthening inequalities 2.16 and 2.18). The third experiment we performed compares these methods with the branch-and-cut algorithms proposed for solving the formulations CCut (the program 4.1 to 4.3) and CCut^t (the program 4.1 to 4.3 with the strengthening inequality 4.4). The results are reported in Tables 4.3 and 4.4. Each line of these tables reports the execution of one algorithm for one dataset, i.e. a set of 10 ELGs with the same *n-d-l* dimension. The first three columns identify the algorithm[2] and the dataset. The next columns have the same meaning as in Table 4.2, except for the columns *Opt*, indicating the number of instances the method was able to solve to optimality, and *Gap*, that reports the relative difference between the *LB* and the *UB*: $Gap = (UB - LB)/UB$. The *Gap* is zero if the optimality is proved. The columns *UB*, *LR*, *LB*, *Gap*, and *Gapr* report average values considering the ten instances in the dataset, while the columns *t(s)*, *lrt(s)*, *Nodes*[3], and *Cuts* report the total sum of these values. Bold results evidence when one method outperformed all competitors.

From Table 4.3 it is possible to observe that the four methods were able to solve all instances with $n = 100$ to optimality within a time limit of two hours. Notwithstanding, the running times of the methods $BC^I(EC^{sn}, DFS)$ and $BC^I(DCut^{sn}, MaxFlow)$ were worse than the ones of the methods $BC^I(CCut, DFS)$ and $BC^I(CCut^t, DFS)$. The best overall performance was obtained by the algorithm $BC^I(CCut^t, DFS)$. Indeed, it has achieved the best execution time for 10 out of 12 datasets and the best linear relaxations.

From the results presented in Table 4.4 it is possible to observe that the methods were not able to solve all instances with $n = 200$ to optimality within a time limit of two hours for each input graph. In this aspect, the best performing method was the algorithm $BC^I(CCut, DFS)$, that was able to solve 113 instances out of 120, followed by the algorithm $BC^I(CCut^t, DFS)$, which solved 112 instances. In special, these methods have solved all 10 instances in the dataset 200-

[2]CCut stands for $BC^I(CCut, DFS)$, CCut^t stands for $BC^I(CCut^t, DFS)$, EC^{sn} stands for $BC^I(EC^{sn}, DFS)$, and DCut^{sn} stands for $BC^I(DCut^{sn}, MaxFlow)$.

[3]For space reasons, k stands for 10^3 times and m stands for 10^6 times.

md-250, while the other two methods have solved only 1 and 2 instances, respectively. Again, the execution times of the CCut based methods are much smaller, as well as the algorithm $BC^I(CCut^t, DFS)$ has achieved the best linear relaxations. For this set of instances, the method $BC^I(CCut, DFS)$ performed slightly better that the $BC^I(CCut^t, DFS)$. In fact, the first has solved one instance more than the latter, and executed faster in 9 out of 12 datasets.

4.4 Concluding remarks

In this chapter we have introduced CCut, a new mathematical formulation based on the concept of colorful cuts for solving both the MLSTP and the GMLSTP. In addition, we have proposed three branch-and-cut algorithms, separation procedures, and performed computational experiments to assess the new proposed methods.

The results have showed that the methods based on the CCut formulation outperformed the best exact methods described in the literature. Considering the benchmark used, the algorithm $BC^I(CCut, DFS)$ has solved to optimality 233 out of 240 input graphs. One reason for this good performance is the absence of edges, arcs, and flow variables, as well as the linking constraints that bound these variables to the label ones. For instance, considering a graph with $n = 200$ and $d = hd$, the EC formulation has 15920 edge variables and the same number of the linking constraints (2.10).

Further, the experiments suggest that it is not worth separating exactly the colorful cuts inequalities (4.2). Despite of that, we believe that the proposed methods can be further improved by adding new families of cuts to it. For instance, the addition of the tree search inequality (4.4) to the model was able to improve significantly its linear relaxation.

Table 4.2: Comparison between the proposed branch-and-cut strategies

	Instance	UB	LR	LB	t(s)	lrt(s)	Nodes	Cuts	Gapr
BC^A	**200-hd-50-0**	2	1.070	1.070	0.011	0.004	0	0	46.5%
BC^R		2	1.070	1.070	0.011	0.004	0	0	46.5%
BC^I		2	1.070	1.070	0.011	0.004	0	0	46.5%
BC^A	**200-hd-100-0**	2	1.288	1.587	1.777	0.015	2204	0	35.6%
BC^R		2	1.288	1.587	1.254	0.015	2204	0	35.6%
BC^I		2	1.288	1.591	**0.949**	**0.011**	2250	0	35.6%
BC^A	**200-hd-200-0**	4	1.776	3.000	287.672	0.022	409657	**1**	55.6%
BC^R		4	1.776	3.000	201.715	0.019	409719	0	55.6%
BC^I		4	1.776	3.000	**105.400**	**0.018**	**350081**	0	55.6%
BC^A	**200-hd-250-0**	4	2.024	3.000	67.946	0.032	69528	2	49.4%
BC^R		4	2.024	3.000	52.117	0.031	69528	2	49.4%
BC^I		4	2.024	3.000	**51.736**	0.031	69528	2	49.4%
BC^A	**200-md-50-0**	2	1.273	1.273	0.014	0.006	0	1	36.3%
BC^R		2	1.273	1.273	0.012	0.005	0	1	36.3%
BC^I		2	1.273	1.273	0.012	**0.004**	0	1	36.3%
BC^A	**200-md-100-0**	4	1.763	3.001	23.178	0.012	52835	**7**	55.9%
BC^R		4	1.763	3.001	15.305	0.012	52863	3	55.9%
BC^I		4	1.763	3.000	**8.603**	0.012	**48694**	3	55.9%
BC^A	**200-md-200-0**	5	2.649	4.000	140.025	0.015	190131	0	47.0%
BC^R		5	2.649	4.000	109.262	0.012	190131	0	47.0%
BC^I		5	2.649	4.000	**62.533**	0.012	**166906**	0	47.0%
BC^A	**200-md-250-0**	6	3.010	5.000	1917.250	0.015	2626788	5	49.8%
BC^R		6	3.010	5.000	1478.250	0.014	2545574	4	49.8%
BC^I		6	3.010	5.000	**844.391**	0.014	**2329440**	**6**	49.8%
BC^A	**200-ld-50-0**	5	3.020	4.248	0.813	0.003	1793	0	39.6%
BC^R		5	3.020	4.248	**0.607**	0.003	1793	0	39.6%
BC^I		5	3.020	4.783	1.097	0.003	5003	0	39.6%
BC^A	**200-ld-100-0**	8	5.010	7.263	27.750	0.009	53171	2	37.4%
BC^R		8	5.010	7.263	21.756	0.009	53171	2	37.4%
BC^I		8	5.010	7.000	**5.970**	**0.006**	**14769**	1	37.4%
BC^A	**200-ld-200-0**	13	8.451	12.000	2676.640	0.009	3522303	**12**	35.0%
BC^R		13	8.451	12.000	2255.690	0.009	3534963	2	35.0%
BC^I		13	8.451	12.000	**1509.730**	0.009	**3319290**	2	35.0%
BC^A	**200-ld-250-0**	14	10.006	13.264	3164.200	0.012	3507374	**21**	28.5%
BC^R		14	10.006	13.000	**655.298**	0.012	**792135**	2	28.5%
BC^I		14	10.006	13.000	796.199	0.012	1439630	1	28.5%

Table 4.3: Computational results for ELGs with |V|=100

	d	**l**	**Opt**	**UB**	**LR**	**LB**	**t(s)**	**lrt(s)**	**Nodes**	**Cuts**	**Gap**	**Gapr**
CCut	**hd**	**25**	10	1.8	1.074	1.074	0.035	0.008	0	0	0%	36%
CCut^t			10	1.8	1.074	1.074	**0.033**	**0.007**	0	0	0%	36%
DCut^{sn}			10	1.8	1.074	1.074	0.421	0.079	0	0	0%	36%
EC^{sn}			10	1.8	1.074	1.074	0.249	0.045	0	0	0%	36%
CCut	**hd**	**50**	10	2	1.293	1.293	**0.087**	0.032	0	1	0%	35%
CCut^t			10	2	1.293	1.293	0.089	**0.029**	0	1	0%	35%
DCut^{sn}			10	2	1.293	1.293	1.187	0.322	0	0	0%	35%
EC^{sn}			10	2	1.293	1.293	0.786	0.333	0	0	0%	35%
CCut	**hd**	**100**	10	3	1.766	2.266	4.021	0.057	18435	24	0%	41%
CCut^t			10	3	**2.010**	2.240	**1.997**	**0.011**	12191	24	0%	**33%**
DCut^{sn}			10	3	1.766	2.229	51.466	0.371	20562	28	0%	41%
EC^{sn}			10	3	1.766	2.192	25.449	0.286	12305	23	0%	41%
CCut	**hd**	**125**	10	4	1.991	3.000	37.688	0.051	167k	4	0%	50%
CCut^t			10	4	1.991	3.000	**37.376**	**0.048**	167k	4	0%	50%
DCut^{sn}			10	4	1.991	3.000	277.572	0.395	136k	9	0%	50%
EC^{sn}			10	4	1.991	3.000	166.031	0.211	135k	8	0%	50%
CCut	**md**	**25**	10	2	1.305	1.321	0.062	0.013	10	1	0%	35%
CCut^t			10	2	1.305	1.321	**0.059**	**0.012**	10	1	0%	35%
DCut^{sn}			10	2	1.305	1.321	0.639	0.183	10	2	0%	35%
EC^{sn}			10	2	1.305	1.321	0.380	0.127	10	0	0%	35%
CCut	**md**	**50**	10	3	1.775	2.233	0.660	0.025	1345	3	0%	41%
CCut^t			10	3	**1.825**	2.183	**0.556**	**0.011**	1437	5	0%	**39%**
DCut^{sn}			10	3	1.775	2.169	6.089	0.164	1008	2	0%	41%
EC^{sn}			10	3	1.775	2.188	3.685	0.137	1020	3	0%	41%
CCut	**md**	**100**	10	4.7	2.647	3.843	20.394	0.032	152k	51	0%	43%
CCut^t			10	4.7	**3.092**	3.811	**18.230**	**0.009**	148k	47	0%	**34%**
DCut^{sn}			10	4.7	2.647	3.777	187.047	0.175	135k	44	0%	43%
EC^{sn}			10	4.7	2.647	3.825	131.048	0.132	153k	44	0%	43%
CCut	**md**	**125**	10	5.2	3.040	4.232	23.863	0.036	137k	112	0%	42%
CCut^t			10	5.2	**3.671**	4.203	**9.079**	**0.012**	61k	49	0%	**29%**
DCut^{sn}			10	5.2	3.040	4.295	324.641	0.193	212k	121	0%	42%
EC^{sn}			10	5.2	3.040	4.267	195.871	0.117	202k	150	0%	42%
CCut	**ld**	**25**	10	4.5	2.859	3.545	0.175	0.011	380	5	0%	36%
CCut^t			10	4.5	2.869	3.561	**0.173**	**0.009**	391	5	0%	36%
DCut^{sn}			10	4.5	2.859	3.579	0.942	0.055	478	4	0%	36%
EC^{sn}			10	4.5	2.859	3.580	0.598	0.033	430	3	0%	36%
CCut	**ld**	**50**	10	6.7	4.550	5.794	**1.171**	0.018	7572	29	0%	32%
CCut^t			10	6.7	**4.694**	5.842	1.274	**0.012**	9037	12	0%	**30%**
DCut^{sn}			10	6.7	4.550	5.733	6.937	0.083	6940	26	0%	32%
EC^{sn}			10	6.7	4.550	5.784	5.096	0.048	8154	20	0%	32%
CCut	**ld**	**100**	10	9.7	7.183	8.951	26.957	0.031	153k	265	0%	26%
CCut^t			10	9.7	**7.593**	8.841	**16.576**	**0.015**	109k	214	0%	**22%**
DCut^{sn}			10	9.7	7.183	8.913	161.964	0.103	160k	196	0%	26%
EC^{sn}			10	9.7	7.183	8.875	104.659	0.064	146k	276	0%	26%
CCut	**ld**	**125**	10	11	8.141	10.164	507.745	0.026	428k	2267	0%	26%
CCut^t			10	11	**8.767**	10.146	**21.259**	**0.016**	109k	566	0%	**20%**
DCut^{sn}			10	11	8.141	10.213	1919.135	0.112	501k	1914	0%	26%
EC^{sn}			10	11	8.141	10.139	564.177	0.074	398k	1617	0%	26%

Table 4.4: Computational results for ELGs with |V|=200

	d	l	Opt	UB	LR	LB	t(s)	lrt(s)	Nodes	Cuts	Gap	Gapr
CCut	**hd**	**50**	10	2	1.080	1.080	**0.113**	**0.045**	0	0	0%	46%
CCut^t			10	2	1.080	1.080	0.122	0.051	0	0	0%	46%
DCut^{sn}			10	2	1.080	1.080	5.931	1.593	0	0	0%	46%
EC^{sn}			10	2	1.080	1.080	3.270	0.893	0	0	0%	46%
CCut	**hd**	**100**	10	2.6	1.301	1.850	10.789	0.133	28k	7	0%	48%
CCut^t			10	2.6	1.301	1.844	**10.713**	**0.113**	28k	7	0%	48%
DCut^{sn}			10	2.6	1.301	1.794	313.865	2.429	27k	7	0%	48%
EC^{sn}			10	2.6	1.301	1.789	166.352	1.156	27k	7	0%	48%
CCut	**hd**	**200**	10	4	1.815	3.000	**961.758**	0.202	3174k	3	0%	55%
CCut^t			10	4	**2.096**	3.000	975.084	**0.025**	3333k	3	0%	**48%**
DCut^{sn}			10	4	1.815	3.000	31958.389	4.657	3222k	16	0%	55%
EC^{sn}			10	4	1.815	3.000	14964.415	2.260	3150k	9	0%	55%
CCut	**hd**	**250**	10	4	2.068	3.024	**434.196**	**0.236**	601k	22	0%	48%
CCut^t			10	4	2.068	3.024	434.990	0.247	601k	22	0%	48%
DCut^{sn}			10	4	2.068	3.088	10246.673	5.020	1201k	36	0%	48%
EC^{sn}			10	4	2.068	3.087	5670.172	2.460	1175k	35	0%	48%
CCut	**md**	**50**	10	2.2	1.316	1.492	**0.516**	**0.066**	481	3	0%	39%
CCut^t			10	2.2	1.316	1.492	0.529	0.072	470	3	0%	39%
DCut^{sn}			10	2.2	1.316	1.495	18.209	0.956	872	5	0%	39%
EC^{sn}			10	2.2	1.316	1.488	8.431	0.556	772	4	0%	39%
CCut	**md**	**100**	10	3.4	1.835	2.480	**17.083**	0.126	66k	12	0%	46%
CCut^t			10	3.4	**1.902**	2.555	17.892	**0.055**	72k	8	0%	**44%**
DCut^{sn}			10	3.4	1.835	2.614	453.806	1.314	83k	9	0%	46%
EC^{sn}			10	3.4	1.835	2.563	214.743	0.655	71k	8	0%	46%
CCut	**md**	**200**	**10**	5.4	2.832	4.400	**2352.854**	0.139	6803k	36	**0%**	48%
CCut^t			**10**	5.4	**3.274**	4.420	2377.483	**0.033**	7077k	63	**0%**	**39%**
DCut^{sn}			9	5.4	2.832	4.349	21063.010	2.271	2921k	57	2%	48%
EC^{sn}			9	5.4	2.832	4.386	14097.577	1.161	3806k	68	2%	48%
CCut	**md**	**250**	**10**	6.3	3.285	5.300	**10557.906**	0.153	27m	57	**0%**	48%
CCut^t			**10**	6.3	**3.938**	5.335	12280.528	**0.034**	34m	22	**0%**	**37%**
DCut^{sn}			1	6.3	3.285	4.984	59653.040	2.556	7446k	39	20%	48%
EC^{sn}			2	6.3	3.285	5.156	54075.540	1.337	13m	34	15%	48%
CCut	**ld**	**50**	10	5.2	2.862	4.293	7.815	0.048	34k	3	0%	45%
CCut^t			10	5.2	**2.878**	4.294	**7.711**	**0.045**	35k	5	0%	**44%**
DCut^{sn}			10	5.2	2.862	4.249	96.178	0.377	34k	1	0%	45%
EC^{sn}			10	5.2	2.862	4.314	62.130	0.165	39k	2	0%	45%
CCut	**ld**	**100**	10	7.9	4.635	7.060	**466.704**	0.075	1786k	27	0%	41%
CCut^t			10	7.9	**4.772**	7.121	529.135	**0.047**	2032k	18	0%	**39%**
DCut^{sn}			10	7.9	4.635	7.033	5689.845	0.474	1475k	24	0%	41%
EC^{sn}			10	7.9	4.635	6.992	3002.548	0.245	1478k	18	0%	41%
CCut	**ld**	**200**	**8**	**12**	7.683	11.053	32303.712	0.096	66m	274	**3%**	36%
CCut^t			7	12.1	**8.117**	10.999	**29671.498**	**0.062**	62m	238	4%	**33%**
DCut^{sn}			1	12.3	7.683	10.567	67219.190	0.828	12m	105	13%	38%
EC^{sn}			5	12.1	7.683	10.716	54320.280	0.400	17m	179	7%	37%
CCut	**ld**	**250**	**5**	13.8	9.007	12.388	**48378.577**	0.113	88m	322	7%	35%
CCut^t			**5**	13.8	**9.606**	12.456	49853.001	**0.055**	92m	298	**6%**	**30%**
DCut^{sn}			1	13.8	9.007	11.822	68644.820	0.932	11m	375	14%	35%
EC^{sn}			3	13.8	9.007	11.971	65669.390	0.417	18m	560	11%	35%

Chapter 5

A Polyhedral Study on CCut Formulation

Studying the polyhedron derived from a MIP formulation is a powerful tool for solving NP-hard problems. This approach can lead to a deeper understanding of the problem and to the discovering of new families of *strong* valid inequalities, which stands for non-dominated and non-redundant ones. In its turn, this kind of cuts are very useful for the design of effective branch-and-cut algorithms. In the sequel, we formalize the concepts of dominated and redundant inequalities, following the definitions given by Wolsey (1998).

Given a set $P = \{x \in \mathbb{R}^n_+ \mid Ax \leq b\}$, where A is an m by n matrix, b an m-dimensional column vector, and x an n-dimensional column vector of variables.

Definition 5.1. *(Wolsey, 1998): An inequality $\mu x \leq \mu_0$ is a valid inequality for P if $\mu x \leq \mu_0$ for all $x \in P$.*

Definition 5.2. *(Wolsey, 1998): If $\mu x \leq \mu_0$ and $\mu' x \leq \mu'_0$ are two valid inequalities for P, the first dominates the second if there exists $u > 0$ such that $\mu \geq u\mu'$, $\mu_0 \leq u\mu'_0$, and $(\mu, \mu_0) \neq (u\mu', u\mu'_0)$.*

Definition 5.3. *(Wolsey, 1998): A valid inequality $\mu x \leq \mu_0$ is redundant in the description of P if there exists $k \geq 2$ valid inequalities $\mu^i x \leq \mu^i_0$. and weights $u^i > 0$, for $i = 1, \cdots, k$, such that $(\sum_{i=1}^k u^i \mu^i)x \leq (\sum_{i=1}^k u^i \mu^i_0)$ dominates $\mu x \leq \mu_0$.*

Figure 5.1 illustrates the concepts of dominated and redundant inequalities. In Fig. 5.1(a), the inequality $2x_1 + 4x_2 \leq 9$ is dominated by the inequality $x_1 + 3x_2 \leq 4$ (take $u = \frac{1}{2}$). In Fig. 5.1(b), the inequality $5x_1 - 2x_2 \leq 6$ is redundant because of the inequalities $9x_1 - 5x_2 \leq 6$ and $6x_1 - x_2 \leq 9$ (consider $u = (\frac{1}{3}, \frac{1}{3})$).

Given the Definitions 5.1, 5.2, and 5.3, we have that it is desirable to know which inequalities are non-redundant in the description of P, i.e., the *necessary* inequalities. In practice, it is important to avoid using an inequality when it is possible to find quickly one (or some) that

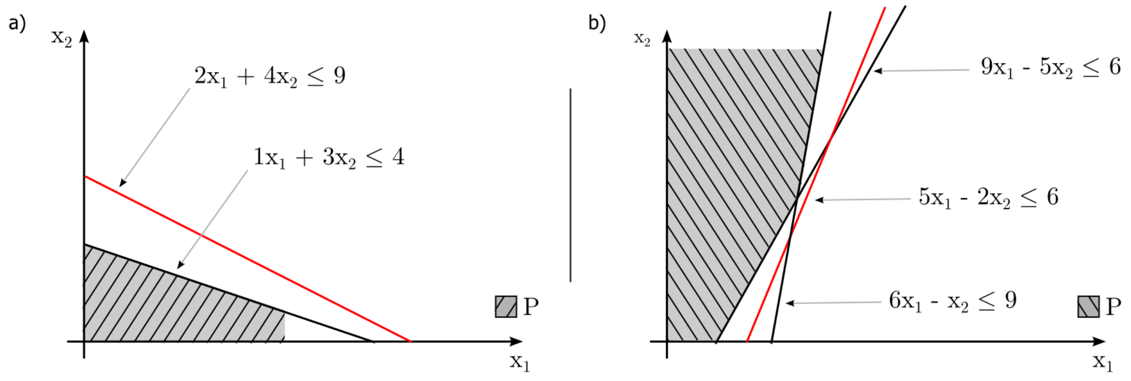


Figure 5.1: Adapted from Wolsey (1998): Example of dominance of inequalities. **(a)** A dominated inequality. **(b)** A redundant inequality

dominates it. The following proposition gives a way to search for strong valid inequalities in the description of polyhedrons.

Proposition 5.1. *(Wolsey, 1998): Given a polyhedron $P \subseteq \mathbb{R}^n$, a valid inequality $\mu x \leq \mu_0$ of P, and the face $F = \{x \in P \mid \mu x = \mu_0\}$. If P is full-dimensional, i.e. its dimension is n, $\mu x \leq \mu_0$ is necessary in the description of P if and only if F is a facet of P, which means the dimension of F is $n-1$.*

In the remaining of this chapter we define the polytope of the CCut formulation, prove that three new families of inequalities and the bounding constraints are facet defining under certain conditions, compare the polytope defined by the formulations described in this work, and present an example ELG that emphasizes the convex hull of the GMLSTP is not completely described. The polyhedral comparison results for the studied polytope show that the new model is theoretically superior to current state-of-the-art formulations. The following section investigate the polyhedron defined by the CCut formulation, proving it is full dimensional for non-trivial instances of the GMLSTP.

5.1 Definition of the CCut polytope

This section formally presents the polyhedron defined by the CCut formulation with the aim of proving it is full dimensional. We are not interested in trivial instances of the GMLSTP and therefore assume that the input ELG is connected and has at least tree vertices, refer to Propositions 3.6 and 3.7. We are neither interested in the case of an input ELG G with *monochromatic cuts*, as given by Definition 4.2. We have already discussed how to deal with monochromatic cuts in the Chapter 3. Refer to Proposition 3.3.

Let $m = |L|$, $Z = (z_l)_{l \in L}$, and let

$$P^{\text{CCut}}(G) := \text{conv}\{Z \in \{0,1\}^m \mid Z \text{ satisfies (4.2)}\} \tag{5.1}$$

be the polytope defined by the CCut formulation. We can now state the central result of this section:

Proposition 5.2. *If G is a monochromatic-cut-free graph, the polytope $P^{CCut}(G)$ is full dimensional, i.e.,*

$$dim(P^{CCut}(G)) = |L| = m. \tag{5.2}$$

Proof. *We prove the proposition by showing that the problem has $m+1$ affinely independent feasible solutions. In fact, since the input graph G is connected, there exists one vector Z^0, representing the feasible solution with all the labels of G. Furthermore, for each label $l \in L$, there exists a vector Z^l, representing the solution $L \backslash \{l\}$; this solution is feasible because there are no monochromatic cuts in the input graph.*

It is easy to verify that these solution vectors are affinely independent by choosing Z^0 as source and subtracting it from all other solutions. This operation leads to m vectors that are clearly linearly independent:

$$\begin{array}{ccccccc|ccccccc}
 & z_1 & z_2 & z_3 & \cdots & z_m & & & z_1 & z_2 & z_3 & \cdots & z_m & \\
Z^0 = [& 1 & 1 & 1 & 1... & 1 &] & & & & & & & \\
Z^1 = [& 0 & 1 & 1 & 1... & 1 &] & Z^1 - Z^0 = [& -1 & 0 & 0 & 0... & 0 &] \\
Z^2 = [& 1 & 0 & 1 & 1... & 1 &] & Z^2 - Z^0 = [& 0 & -1 & 0 & 0... & 0 &] \\
\vdots & & & & & & & \vdots & & & & & & \\
Z^m = [& 1 & 1 & 1 & 1... & 0 &] & Z^m - Z^0 = [& 0 & 0 & 0 & 0... & -1 &].
\end{array}$$

□

In the following sections, we discuss cases in which the variable bounding inequalities are facet defining and examine in detail the colorful cut inequalities (4.2) with respect to the polytope $P^{\text{CCut}}(G)$. We also introduce new concepts to support the propositions.

5.2 Variable bounding inequalities

Given that every polytope $P^{\text{CCut}}(G)$ is contained in a unitary m-dimensional hypercube, the variable bounding inequalities (5.3) and (5.4) are clearly valid for the GMLSTP. In this section, we argue that the bounding inequalities (5.3) are always facet defining and that the non-negativity inequalities (5.4) also define facets under certain conditions.

$$z_l \leq 1, \qquad \forall l \in L. \tag{5.3}$$

$$z_l \geq 0, \qquad \forall l \in L. \tag{5.4}$$

Theorem 5.1. *For any $l \in L$, the associated variable bounding inequality $z_l \leq 1$ defines a facet of the polytope $P^{CCut}(G)$.*

Proof. *For every label $l \in L$, we can use all the solution vectors described in Proposition 5.2, except Z^l. Since these solutions are a subset of an affinely independent set of points, they are also affinely independent.* □

Theorem 5.2. *The inequality $z_l \geq 0$, for $l \in L$, defines a facet of the polytope $P^{CCut}(G)$ if and only if l is not part of any colorful cut $K(S)$ of G with $|K(S)| = 2$.*

Proof. *($\Leftarrow$) For any $l \in L$, let Z^0 be the solution vector corresponding to the set of labels $L\backslash\{l\}$, and let Z^k, $k \in \{1,2,\cdots,m-1\}$, be the solution vector corresponding to $L\backslash\{k,l\}$. Since l is not part of any colorful cut with $|K(S)| = 2$, these solution vectors are feasible and affinely independent. In fact, choosing Z^0 as the source and subtracting it from all other solutions yields $m-1$ linearly independent vectors:*

$$\begin{array}{rcccccc}
 & z_1 & z_2 & \cdots & z_{m-1} & z_m & \\
Z^0 = [& 1 & 1 & 1\ldots & 1 & 0 &] \\
Z^1 = [& 0 & 1 & 1\ldots & 1 & 0 &] \\
Z^2 = [& 1 & 0 & 1\ldots & 1 & 0 &] \\
\vdots & & & & & & \\
Z^{m-1} = [& 1 & 1 & 1\ldots & 0 & 0 &]
\end{array}
\quad
\begin{array}{rcccccc}
 & z_1 & z_2 & \cdots & z_{m-1} & z_m & \\
 & & & & & & \\
Z^1 - Z^0 = [& -1 & 0 & 0\ldots & 0 & 0 &] \\
Z^2 - Z^0 = [& 0 & -1 & 0\ldots & 0 & 0 &] \\
\vdots & & & & & & \\
Z^{m-1} - Z^0 = [& 0 & 0 & 0\ldots & -1 & 0 &].
\end{array}$$

($\Rightarrow$) If $K(S) = \{k,l\}$ is a colorful cut of G, and the associated inequality (4.2) $z_k + z_l \geq 1$ is valid, then $(z_l + z_k \geq 1) - (z_k \leq 1)$ is exactly $z_l \geq 0$, which is redundant. □

5.3 The colorful cut inequalities

The main objective of this section is to prove that the colorful cut inequalities (4.2) are facet defining in many cases. However, to do so we must first define the concepts of *minimal colorful cuts* and *T-labels* as well as some of their properties.

Definition 5.4. *A colorful cut $K(S)$ is minimal if there does not exist a disconnecting set of labels K' such that $K' \subset K(S)$.*

Property 5.1. *Let $K(S)$ be a minimal colorful cut of the graph G; thus $G[L\backslash L']$ is connected for every subset $L' \subset K(S)$.*

Proof. *If $G[L\backslash L']$ is disconnected for some subset $L' \subset K(S)$, then L' is a disconnecting set of labels, and $K(S)$ is not a minimal colorful cut.* $\square$

Note that Property 5.1 can be used to verify whether a given colorful cut $K(S)$ is minimal. Indeed, if $G[L\backslash L']$ is connected for every $L' \subset K(S)$ such that $|L'| = |K(S)| - 1$, then $K(S)$ is minimal.

Definition 5.5. *Given a minimal colorful cut $K(S)$ and a label $X \in L\backslash K(S)$, we say that X is a T-label induced by $K(S)$ if X is part of a monochromatic cut in* **every** *graph $G[L\backslash L']$ such that $L' \subset K(S)$ and $|L'| = |K(S)| - 1$.*

In other words, if a minimal colorful cut $K(S)$ induces a T-label X, then every solution that has just one label of $K(S)$ must also have X because it becomes a monochromatic cut. The implicit interaction between the colorful cut $K(S)$ and its induced T-label X leads to the valid inequality

$$\Big(\sum_{l\in K(S)} z_l\Big) + z_X \geq 2, \qquad X \in \mathcal{T}(S), \tag{5.5}$$

where $\mathcal{T}(S)$ denotes the set of T-labels induced by $K(S)$.

Property 5.2. *Let $K(S)$ be a colorful cut of G for $S \subset V$, and let $\overline{K}(S) = L\backslash K(S)$. If $X \in \overline{K}(S)$ is not a T-label induced by $K(S)$, then there exists a label $k \in K(S)$, denoted by $U(S,X)$, such that the solution $L' = U(S,X) \cup \overline{K}(S,X)$ is feasible, where $\overline{K}(S,X) = \overline{K}(S)\backslash\{X\}$.*

Proof. *Let us assume the contrary. Suppose that $X \notin \mathcal{T}(S)$ and $\nexists k \in K(S)$ such that the solution $\{k\} \cup \overline{K}(S,X)$ is feasible. In this case, X is part of a monochromatic cut in $G[\{k\} \cup \overline{K}(S)]$, $\forall k \in K(S)$. According to Definition 5.5, $X \in \mathcal{T}(S)$. This is a contradiction.* $\square$

Figure 5.2 illustrates the concept of a T-label induced by a colorful cut. The example in Fig. 5.2a presents the graph G, in which $F \in \mathcal{T}(\{1\})$; Fig. 5.2b shows that for $G[L \backslash \{A,B\}]$, $K(\{1,4\}) = \{F\}$ is a monochromatic cut; in Fig. 5.2c $K(\{7,8\}) = \{F\}$ is a monochromatic cut for $G[L \backslash \{A,C\}]$; and Fig. 5.2d shows that, for $G[L \backslash \{B,C\}]$, $K(\{4\}) = \{F\}$ is a monochromatic cut. Observe that $D \notin \mathcal{T}(\{1\})$. In fact, D is not a monochromatic cut in the graph $G[L \backslash \{A,C\}]$; thus $U(\{1\},D) = B$.

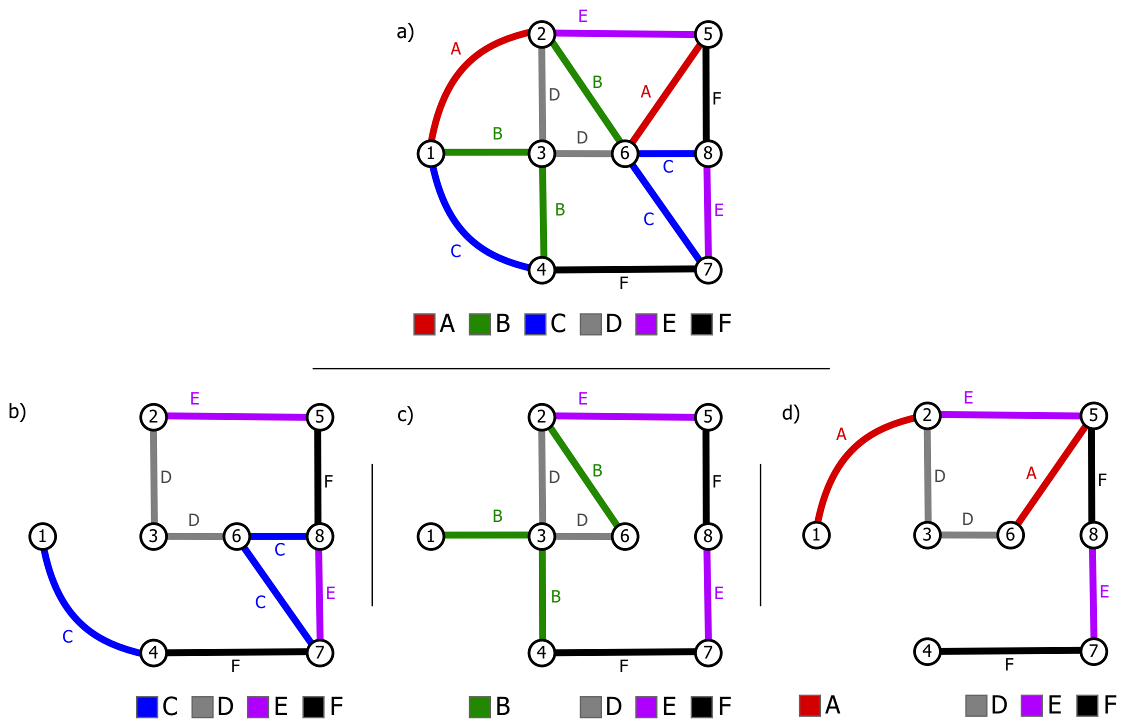


Figure 5.2: The T-label concept. **(a)** Example graph G, in which $\mathcal{T}(\{1\}) = \{F\}$. **(b)** F is a monochromatic cut if we remove A and B from G. **(c)** F is a monochromatic cut if we remove A and C. **(d)** F is a monochromatic cut if we remove B and C

Let $K(S)$ be a minimal colorful cut of the graph G, $Z = (z_l)_{l \in L}$, and let

$$F^{\text{CCut}}(S) := \{Z \in P^{\text{CCut}}(G) \mid \sum_{l \in K(S)} z_l = 1\} \tag{5.6}$$

be the face induced by the respective colorful cut inequality (4.2). To prove that $F^{\text{CCut}}(S)$ is a facet of the polytope $P^{\text{CCut}}(G)$, we use the following steps (known as indirect proof, see Wolsey and Nemhauser (2014)).

First, we introduce $m = |L|$ solutions that are in $F^{\text{CCut}}(S)$. Then, we suppose that these solutions lie on a generic hyperplane $\mu z = \mu_0$. Finally, we prove that this hyperplane is exactly a multiple of the associated colorful cut inequality (4.2).

For any $S \subset V$, let $s = |K(S)|$. Without loss of generality, let the indices of the labels in $K(S)$ be $1, 2, \cdots, s$. Furthermore, let Z^l, for $l \in \{1, 2, \cdots, s, s+1, \cdots, m\}$, be the solution vectors built as follows.

For any $l \in \{1, 2, \cdots, s\}$, Z^l is the solution vector corresponding to the set of labels $\{l\} \cup \overline{K}(S)$; and, for any $l \in \{s+1, \cdots, m\}$, Z^l is the solution vector corresponding to the set of labels $U(S,l) \cup \overline{K}(S,l)$, where $U(S,l)$ is underlined:

$$\begin{array}{r cccc cccc l}
 & \multicolumn{4}{c}{K(S)} & \multicolumn{4}{c}{\overline{K}(S)} & \\
\cline{2-5}\cline{6-9}
 & z_1 & z_2 & \cdots & z_s & z_{s+1} & z_{s+2} & \cdots & z_m & \\
Z^1 = [& 1 & 0 & 0\ldots & 0 & 1 & 1 & 1\ldots & 1 &] \\
Z^2 = [& 0 & 1 & 0\ldots & 0 & 1 & 1 & 1\ldots & 1 &] \\
 & & & & & \vdots & & & & \\
Z^s = [& 0 & 0 & 0\ldots & 1 & 1 & 1 & 1\ldots & 1 &] \\
 & & & & & & & & & \\
Z^{s+1} = [& 0 & 0 & 0\ldots & \underline{1} & 0 & 1 & 1\ldots & 1 &] \\
Z^{s+2} = [& \underline{1} & 0 & 0\ldots & 0 & 1 & 0 & 1\ldots & 1 &] \\
 & & & & & \vdots & & & & \\
Z^m = [& 0 & \underline{1} & 0\ldots & 0 & 1 & 1 & 1\ldots & 0 &].
\end{array}$$

According to Property 5.1, any solution in the form $\{l\} \cup \overline{K}(S)$ is feasible for $l \in K(S)$; according to Property, 5.2 any solution in the form $U(S,l) \cup \overline{K}(S,l)$ is feasible for $l \in \overline{K}(S)$. In addition, it is evident that by construction

$$Z^l \in F^{\text{CCut}}(S), \qquad \forall l \in \{1, 2, \cdots, m\}. \tag{5.7}$$

Moreover, we have the following lemma:

Lemma 5.1. *Let $e^s \in \mathbb{R}^m$ be a vector representing the left-hand-side coefficients of the corresponding inequality (4.2), defined as follows:*

$$e^s_l = \begin{cases} 1 & \text{for } 1 \le l \le s, \\ 0 & \text{for } s+1 \le l \le m. \end{cases}$$

With respect to the already defined set of solution vectors Z, consider the following linear system:

$$\mu \bullet Z^l = \mu_0, \qquad \forall l \in \{1, 2, \cdots, m\}, \tag{5.8}$$

where $\mu \in \mathbb{R}^m$, $\mu_0 \in \mathbb{R}$, and '$\bullet$' stands for the usual scalar product. The unique solution of (5.8) verifies

$$(\mu, \mu_0) = \alpha(e^s, 1). \tag{5.9}$$

Proof. *For any $l \in \{1, 2, \cdots, s-1\}$,*

$$\mu \bullet Z^l = \mu_0 = \mu \bullet Z^{l+1} \Rightarrow \mu \bullet Z^l = \mu \bullet Z^{l+1} \Rightarrow \mu_l = \mu_{l+1}.$$

Consequently,

$$\mu_1 = \mu_2 = \cdots = \mu_s = \alpha.$$

In addition, for any $l \in \{s+1, s+2, \cdots, m\}$,

$$\mu \bullet Z^{U(S,l)} = \mu_0 = \mu \bullet Z^l \Rightarrow \mu \bullet Z^{U(S,l)} = \mu \bullet Z^l \Rightarrow \mu_l = 0.$$

Therefore, we also have

$$\alpha = \mu_0,$$

and finally,

$$(\mu, \mu_0) = \alpha(e^s, 1).$$

□

Theorem 5.3. *The face $F^{CCut}(S)$ is a facet of $P^{CCut}(G)$ if and only if $K(S)$ is a minimal colorful cut and it does not induce any T-labels, i.e., $\mathcal{T}(S) = \emptyset$.*

Proof. *($\Leftarrow$) It follows from Lemma 5.1 that the unique solution of (5.8) verifies (5.9). In this case, the provided solution vectors Z^l, for $l \in \{1, 2, \cdots, m\}$, are affinely independent. Therefore, $dim(F^{CCut}(S)) = m-1$, and $F^{CCut}(S)$ is a facet of $P^{CCut}(G)$.*

($\Rightarrow$) If the colorful cut $K(S)$ is not minimal, then there exists a disconnecting set of labels $K' \subset K(S)$, and the valid inequality

$$\sum_{l \in K'} z_l \geq 1$$

clearly dominates the colorful cut inequality associated with $K(S)$.

On the other hand, if $X \in \mathcal{T}(S)$, then the T-label inequality (5.5) associated with $K(S)$ and X is valid, and

$$\Big((\sum_{l \in K(S)} z_l) + z_X \geq 2 \Big) + (-z_X \geq -1)$$

is exactly the colorful cut inequality associated with $K(S)$, which is redundant. In this case, $F^{CCut}(S)$ is not a facet of $P^{CCut}(G)$. □

5.4 The T-label inequalities

In the previous section, we have shown that inequality (5.5) is valid if a minimal colorful cut $K(S)$ induces a T-label X. In this section, we present the entire family of these inequalities, and we prove that they define facets of $P^{\mathrm{CCut}}(G)$ under certain conditions. To this end, we must first extend the T-label concept and discuss some of its new properties.

Definition 5.6. *Given a minimal colorful cut $K(S)$ and a label $X \in \overline{K}(S)$, we say that X is a k-T-label induced by $K(S)$ if X is part of a monochromatic cut in **every** graph $G[L \backslash L']$ such that $L' \subset K(S)$ and $|L'| = |K(S)| - k$.*

The concept of k-T-labels extends the concept of T-labels in such a way that even if we keep k labels of $K(S)$, then we must also use the induced k-T-label because it becomes a monochromatic cut. Likewise, for T-labels, if a colorful cut $K(S)$ induces a k-T-label X, then the following inequality is valid:

$$(\sum_{l \in K(S)} z_l) + k \cdot z_X \geq 1 + k, \qquad X \in \mathcal{T}^k(S), \tag{5.10}$$

where $\mathcal{T}^k(S)$ denotes the set of k-T-labels induced by $K(S)$.

Figure 5.3 illustrates the concept of a 2-T-label induced by a colorful cut. The example given in Fig. 5.3a presents the graph G, in which $F \in \mathcal{T}^2(\{1\})$; Fig. 5.3b shows that for

$G[L\backslash\{A\}]$, $K(\{5\}) = \{F\}$ is a monochromatic cut; in Fig. 5.3c, $K(\{7,8\}) = \{F\}$ is a monochromatic cut for $G[L\backslash\{C\}]$; and Fig. 5.3d shows that, for $G[L\backslash\{B\}]$, $K(\{1,2,3,4\}) = \{F\}$ is a monochromatic cut.

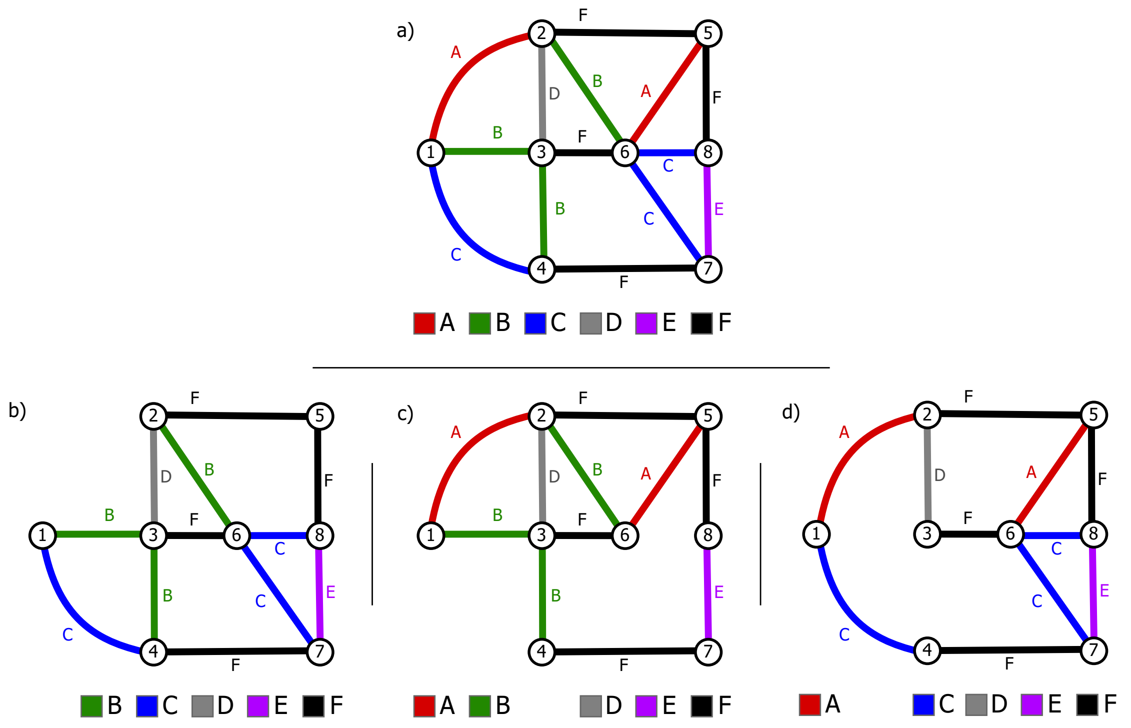


Figure 5.3: The k-T-label concept. **(a)** Example graph G, in which $\mathcal{T}^2(\{1\}) = \{F\}$. **(b)** F is a monochromatic cut if we remove A from G. **(c)** F is a monochromatic cut if we remove C. **(d)** F is a monochromatic cut if we remove B

In the following, we refine the characterization of the k-T-labels and their inducing colorful cuts. For the following properties, let $K(S)$ be a minimal colorful cut and let $s = |K(S)|$.

Property 5.3. *If $k \geq s$, then $\mathcal{T}^k(S) = \emptyset$.*

Property 5.4. *If a label $X \in \overline{K}(S)$ is a $(k+1)$-T-label induced by $K(S)$, then X is also a k-T-label induced by $K(S)$, for $k \geq 1$. It follows that*

$$\mathcal{T}^{s-1}(S) \subseteq \mathcal{T}^{s-2}(S) \subseteq \cdots \subseteq \mathcal{T}^2(S) \subseteq \mathcal{T}(S).$$

Note that, even when $X \in \overline{K}(S)$ is a $(k+1)$-T-label induced by $K(S)$, the corresponding inequality (5.5) remains valid (but redundant). Starting from (5.10) and adding to it $k \cdot (-z_X \geq -1)$, we arrive exactly at inequality (5.5).

Property 5.5. *If $X \in \overline{K}(S)$ is **not** a k-T-label induced by $K(S)$ for $k < |K(S)|$, then there exists a set $U^k(S,X) \subset K(S)$ with exactly k labels such that the solution $U^k(S,X) \cup \overline{K}(S,X)$ is feasible, where $\overline{K}(S,X) = \overline{K}(S)\backslash\{X\}$.*

Proof. *Suppose not. Suppose that $X \notin \mathfrak{T}^k(S)$ and $\nexists K' \subset K(S)$, $|K'| = k$ such that the solution $K' \cup \overline{K}(S,X)$ is feasible. In this case, X is part of a monochromatic cut in the graph $G[K' \cup \overline{K}(S)]$, $\forall K' \subset K(S)$, and $X \in \mathfrak{T}^k(S)$ according to Definition 5.6.* □

Lemma 5.2. *Let $K(S)$ be a minimal colorful cut of a graph G such that $X,Y \in \mathfrak{T}(S)$. The inequality*

$$(\sum_{l \in K(S)} z_l) + z_X + z_Y \geq 3, \qquad\qquad X,Y \in \mathfrak{T}(S), \tag{5.11}$$

is valid with respect to $P^{CCut}(G)$ if one of these four cases holds:

(a) The inequality $z_X + z_Y \geq 1$ is valid.

(b) $\nexists$ $\{k,w\}$ such that $\{k,w\} = U^2(S,X) = U^2(S,Y)$.

(c) For any $\{k,w\} = U^2(S,X) = U^2(S,Y)$, there exists a set $K' \subseteq K(S) \backslash \{k,w\}$ such that $\{X,Y\} \cup K'$ is a disconnecting set of labels.

(d) $X \in \mathfrak{T}^2(S)$ or $Y \in \mathfrak{T}^2(S)$.

Proof. *Let $L' \subseteq L$ be a feasible solution of the GMLSTP for the graph G. Consider the following cases:*

(1) If $|K(S) \cap L'| = 1$, then inequality (5.11) is clearly valid because $X,Y \in \mathfrak{T}(S)$.

(2) If $|K(S) \cap L'| = 2$, then neither X nor Y is needed. However, if (a), then either X or Y is needed and (5.11) is valid. Let $\{k,w\} = K(S) \cap L'$. The same occurs if (b). If $\{k,w\} = U^2(S,X)$, then $Y \in L'$. If $\{k,w\} = U^2(S,Y)$, then $X \in L'$. If (c), even with $\{k,w\} = U^2(S,X) = U^2(S,Y)$, then either X or Y is needed. This occurs because (a) is valid in every graph $G[\{k,w\} \cup \overline{K}(S)]$. Finally, (d) reduces to (b).

(3) If $|K(S) \cap L'| \geq 3$, then inequality (5.11) is also valid. □

Property 5.6. *If $X,Y \in \mathfrak{T}(S)$ and the inequality $(\sum_{l \in K(S)} z_l) + z_X + z_Y \geq 3$ is not valid (refer to Lemma 5.2), then there exists a valid solution of the form $U^2(S,Y) \cup \overline{K}(S,Y) \backslash \{X\}$.*

Proof. *Let $\{k,w\} = U^2(S,X) = U^2(S,Y)$ such that $\nexists K' \subseteq K(S) \backslash \{k,w\} \mid \{X,Y\} \cup K'$ is a disconnecting set of labels. The solution $\{k,w\} \cup \overline{K}(S,Y) \backslash \{X\}$ is clearly valid.* □

In the following, we address the T-label family of inequalities (5.12).

$$\Big(\sum_{l\in K(S)} z_l\Big) + z_X \geq 2, \qquad \begin{array}{r}\forall S \subset V, S \neq \emptyset,\\ \forall X \in \mathcal{T}(S).\end{array} \tag{5.12}$$

Let $K(S)$ be a minimal colorful cut of the graph G, $X \in \mathcal{T}(S)$, and let

$$F^{\text{T-label}}(S,X) := \{Z \in P^{\text{CCut}}(G) \mid \Big(\sum_{l\in K(S)} z_l\Big) + z_X = 2\} \tag{5.13}$$

be the face induced by the respective T-label inequality (5.12). To prove that $F^{\text{T-label}}(S,X)$ is a facet of the polytope $P^{\text{CCut}}(G)$, we again use the indirect proof. First, we introduce $m \geq |L|$ solutions that are in $F^{\text{T-label}}(S,X)$. We then suppose that these solutions lie on a generic hyperplane $\mu z = \mu_0$. Finally, we prove that this hyperplane is exactly a multiple of the associated T-label inequality (5.12).

Let $K(S)$ be a minimal colorful cut, where $s = |K(S)|$, $t = |\mathcal{T}(S)|$, $X \in \mathcal{T}(S)$, $\mathcal{T}^2(S) = \emptyset$, and Lemma 5.2 does not hold for any $Y \in \mathcal{T}(S)\backslash\{X\}$. Without loss of generality, let the indices of the labels in $K(S)$ be $1, 2, \cdots, s$; let the indices of the labels in $\mathcal{T}(S)$ be $m-t+1, m-t+2, \cdots, m$; and let the index of the label X be m. Furthermore, consider the Z^l, for $l \in \{1, 2, \cdots, s, s+1, \cdots, m-1, m, m+1, \cdots, m+t-1\}$, solution vectors:

$$
\begin{array}{rcccccccccccccl}
 & & \multicolumn{4}{c}{K(S)} & \multicolumn{4}{c}{\overline{K}(S)\setminus\mathcal{T}(S)} & \multicolumn{4}{c}{\mathcal{T}(S)} & \\
 & & z_1 & z_2 & \cdots & z_s & z_{s+1} & z_{s+2} & \cdots & z_{m-t} & z_{m-t+1} & z_{m-t+2} & \cdots & z_m & \\
Z^1 = & [& 1 & 0 & 0... & 0 & 1 & 1 & 1... & 1 & 1 & 1 & 1... & 1 &] \\
Z^2 = & [& 0 & 1 & 0... & 0 & 1 & 1 & 1... & 1 & 1 & 1 & 1... & 1 &] \\
 & & & & & & \vdots & & & & \vdots & & & & \\
Z^s = & [& 0 & 0 & 0... & 1 & 1 & 1 & 1... & 1 & 1 & 1 & 1... & 1 &] \\
Z^{s+1} = & [& 0 & 0 & 0... & \underline{1} & 0 & 1 & 1... & 1 & 1 & 1 & 1... & 1 &] \\
Z^{s+2} = & [& \underline{1} & 0 & 0... & 0 & 1 & 0 & 1... & 1 & 1 & 1 & 1... & 1 &] \\
 & & & & & & \vdots & & & & \vdots & & & & \\
Z^{m-t} = & [& 0 & \underline{1} & 0... & 0 & 1 & 1 & 1... & 0 & 1 & 1 & 1... & 1 &] \\
Z^{m-t+1} = & [& 0 & \underline{1} & 0... & \underline{1} & 1 & 1 & 1... & 1 & 0 & 1 & 1... & 0 &] \\
Z^{m-t+2} = & [& \underline{1} & 0 & 0... & \underline{1} & 1 & 1 & 1... & 1 & 1 & 0 & 1... & 0 &] \\
 & & & & & & \vdots & & & & \vdots & & & & \\
Z^m = & [& 0 & \underline{1} & 0... & \underline{1} & 1 & 1 & 1... & 1 & 1 & 1 & 1... & 0 &] \\
Z^{m+1} = & [& \underline{1} & 0 & 0... & \underline{1} & 1 & 1 & 1... & 1 & 1 & 1 & 1... & 0 &] \\
 & & & & & & \vdots & & & & \vdots & & & & \\
Z^{m+t-1} = & [& \underline{1} & \underline{1} & 0... & 0 & 1 & 1 & 1... & 1 & 1 & 1 & 1... & 0 &].
\end{array}
$$

For any $l \in \{1,2,\cdots,s\}$, Z^l is the solution vector corresponding to the set of labels $\{l\} \cup \overline{K}(S)$. For any $l \in \{s+1,\cdots,m-t\}$, Z^l is the solution vector corresponding to the set of labels $U(S,l) \cup \overline{K}(S,l)$, where $U(S,l)$ is underlined. For $l \in \{m-t+1, m-t+2, \cdots, m-1\}$, Z^l is the solution vector corresponding to the set of labels $\big(U^2(S,l) = U^2(S,X)\big) \cup \overline{K}(S,l)\setminus\{X\}$, $l \in \mathcal{T}(S)\setminus\{X\}$, and any label $w \in U^2(S,l)$ is underlined. Finally, Z^m, $Z^{m+1}, \cdots, Z^{m-t+1}$, are the solution vectors $\big(U^2(S,l) = U^2(S,X)\big) \cup \overline{K}(S,X)$, $\forall l \in \mathcal{T}(S)$.

According to Property 5.1, any solution of the form $\{l\} \cup \overline{K}(S)$ is feasible for $l \in K(S)$; according to Property 5.2, any solution of the form $U(S,l) \cup \overline{K}(S,l)$ is feasible for $l \in \overline{K}(S)$. According to Property 5.6, the solutions $U^2(S,l) \cup \overline{K}(S,l)\setminus\{X\}$ are feasible because Lemma 5.2 does not hold. Finally, the solutions $U^2(S,X) \cup \overline{K}(S,X)$ are also feasible according to Property 5.5. In addition, it is evident that from the construction

$$
Z^l \in F^{\text{T-label}}(S,X), \qquad \forall l \in \{1,2,\cdots,m,m+1,\cdots,m-t+1\}. \tag{5.14}
$$

Lemma 5.3. *Let $e^s \in \mathbb{R}^m$ be a vector representing the left-hand-side coefficients of the corresponding inequality (5.12), defined as follows:*

$$e^s_l = \begin{cases} 1 & \textit{for } 1 \leq l \leq s, \\ 0 & \textit{for } s+1 \leq l \leq m-1, \\ 1 & \textit{for } l = m. \end{cases}$$

With respect to the already defined set of solution vectors Z, consider the following linear system:

$$\mu \bullet Z^l = \mu_0, \qquad \forall l \in \{1, 2, \cdots, m, m+1, \cdots, m-t+1\}, \tag{5.15}$$

where $\mu \in \mathbb{R}^m$, $\mu_0 \in \mathbb{R}$, and '$\bullet$' stands for the usual scalar product. The unique solution of (5.15) verifies

$$(\mu, \mu_0) = \alpha(e^s, 2). \tag{5.16}$$

Proof. *For any $l \in \{1, 2, \cdots, s-1\}$,*

$$\mu \bullet Z^l = \mu_0 = \mu \bullet Z^{l+1} \Rightarrow \mu \bullet Z^l = \mu \bullet Z^{l+1} \Rightarrow \mu_l = \mu_{l+1}.$$

It follows that,

$$\mu_1 = \mu_2 = \cdots = \mu_s = \alpha.$$

In addition, for any $l \in \{s+1, s+2, \cdots, m-t\}$,

$$\mu \bullet Z^{U(S,l)} = \mu_0 = \mu \bullet Z^l \Rightarrow \mu \bullet Z^{U(S,l)} = \mu \bullet Z^l \Rightarrow \mu_l = 0.$$

For $l \geq m$ and $w \in U^2(S,l)$,

$$\mu \bullet Z^w = \mu_0 = \mu \bullet Z^l \Rightarrow \mu \bullet Z^w = \mu \bullet Z^l \Rightarrow \mu_w = \mu_l = \alpha.$$

For $l \in \{m-t+1, m-t+2, \cdots, m-1\}$,

$$\mu.Z^l = \mu_0 = \mu.Z^m, \big(U^2(S,l) = U^2(S,X)\big) \Rightarrow \mu.Z^l = \mu.Z^m \Rightarrow \mu_l = 0.$$

Therefore, we also have

$$2\alpha = \mu_0.$$

Finally,

$$(\mu, \mu_0) = \alpha(e^s, 2).$$

□

We can now state the central theorem of this section:

Theorem 5.4. *The face $F^{\text{T-label}}(S,X)$ is a facet of $P^{CCut}(G)$ if and only if the following three conditions are verified:*

(a) $K(S)$ is a minimal colorful cut.

(b) Inequality (5.11) is not valid for any $Y \in \mathcal{T}(S)\backslash\{X\}$.

(c) $\mathcal{T}^2(S) = \emptyset$. It follows that, X is not a 2-T-label induced by $K(S)$.

Proof. *($\Leftarrow$) It follows from Lemma 5.3 that the unique solution of (5.15) verifies (5.16). In this case, the provided solution vectors Z^l, for $l \in \{1,2,\cdots,m,m+1,\cdots\}$, are affinely independent. Therefore, $dim(F^{\text{T-label}}(S,X)) = m-1$, and $F^{\text{T-label}}(S,X)$ is a facet of $P^{CCut}(G)$.*

((a) $\Rightarrow$) If the colorful cut $K(S)$ is not minimal, then there exists a disconnecting set $K' \subset K(S)$, and the valid inequality

$$(\sum_{l \in K'} z_l) + z_X \geq 2$$

clearly dominates the T-label inequality associated with $K(S)$.

((b) $\Rightarrow$) If inequality (5.2) is valid for any additional label $Y \in \mathcal{T}(S)$, then

$$\Big((\sum_{l \in K(S)} z_l) + z_X + z_Y \geq 3\Big) + (-z_Y \geq -1)$$

is exactly the T-label inequality associated with $K(S)$, which is redundant.

((c) ⇒) If $X \in \mathcal{T}^2(S)$, then the 2-T-label inequality (5.10) associated with $K(S)$ and X is valid and

$$\Big(\big(\sum_{l\in K(S)} z_l\big) + 2\cdot z_X \geq 3\Big) + (-z_X \geq -1)$$

is exactly the T-label inequality associated with $K(S)$, which is redundant.

For any of the cases (a), (b), or (c), $F^{\text{T-label}}(S,X)$ is not a facet of $P^{\text{CCut}}(G)$. □

5.5 The k-T-Label inequalities

This section presents the k-T-label family of inequalities. We extend Theorem 5.4 and prove that these inequalities also define facets of the polytope $P^{\text{CCut}}(G)$ under some conditions. Let

$$\big(\sum_{l\in K(S)} z_l\big) + k\cdot z_X \geq 1+k, \qquad \begin{array}{r}\forall S\subset V, S\neq\emptyset,\\ \forall k\in\{1,...,K(S)-1\},\\ \forall X\in\mathcal{T}^k(S),\end{array} \tag{5.17}$$

be the family of k-T-label inequalities. Moreover, let $K(S)$ be a minimal colorful cut of the graph G, $s = |K(S)|$, $X \in \mathcal{T}^k(S)$, and let

$$F^{\text{k-T-label}}(S,X) := \{Z\in P^{\text{CCut}}(G) \mid \big(\sum_{l\in K(S)} z_l\big) + k\cdot z_X = k+1\} \tag{5.18}$$

be the face induced by the respective k-T-label inequality (5.17).

We prove that $F^{\text{k-T-label}}(S,X)$ is a facet of the polytope $P^{\text{CCut}}(G)$ by indirect proof. Without loss of generality, let the indices of the labels in $K(S)$ be $1,2,\cdots,s$, and let the index of the label X be m. Furthermore, let Z^l, for $l\in\{1,2,\cdots,s,s+1,\cdots,m-1,m\}$, be the solution vectors built as follows.

For any $l\in\{1,2,\cdots,s\}$, Z^l is the solution vector corresponding to the set of labels $\{l\}\cup\overline{K}(S)$. For any $l\in\{s+1,\cdots,m-1\}$, Z^l is the solution vector corresponding to the set of labels $U(S,l)\cup\overline{K}(S,l)$. For $l=m$, Z^l is the solution vector corresponding to the set of labels $U^k(S,X)\cup\overline{K}(S,X)$.

It is evident that the proposed solution vectors represent feasible solutions according to Properties 5.1, 5.2, and 5.5. In addition, it is evident that

$$Z^l \in F^{\text{k-T-label}}(S,X), \qquad \forall l \in \{1,2,\cdots,m\}. \tag{5.19}$$

Thus, we can state the following lemma:

Lemma 5.4. *Let $e^s \in \mathbb{R}^m$ be a vector representing the left-hand-side coefficients of the corresponding inequality (5.17), defined as follows:*

$$e^s_l = \begin{cases} 1 & \text{for } 1 \leq l \leq s, \\ 0 & \text{for } s+1 \leq l \leq m-1, \\ k & \text{for } l = m. \end{cases}$$

With respect to the already defined set of solution vectors Z, consider the following linear system:

$$\mu \bullet Z^l = \mu_0, \qquad \forall l \in \{1,2,\cdots,m\}, \tag{5.20}$$

where $\mu \in \mathbb{R}^m$, $\mu_0 \in \mathbb{R}$, and '$\bullet$' stands for the usual scalar product. The unique solution of (5.20) verifies

$$(\mu,\mu_0) = \alpha(e^s, k+1). \tag{5.21}$$

Proof. *For any $l \in \{1,2,\cdots,s-1\}$,*

$$\mu \bullet Z^l = \mu_0 = \mu \bullet Z^{l+1} \Rightarrow \mu \bullet Z^l = \mu \bullet Z^{l+1} \Rightarrow \mu_l = \mu_{l+1}.$$

It follows that,

$$\mu_1 = \mu_2 = \cdots = \mu_s = \alpha.$$

In addition, for any $l \in \{s+1,s+2,\cdots,m-1\}$,

$$\mu \bullet Z^{U(S,l)} = \mu_0 = \mu \bullet Z^l \Rightarrow \mu \bullet Z^{U(S,l)} = \mu \bullet Z^l \Rightarrow \mu_l = 0.$$

Finally, for $l = m$ and $w \in U^{k+1}(S,l)$,

$$\mu \bullet Z^w = \mu_0 = \mu \bullet Z^l \Rightarrow \mu \bullet Z^w = \mu \bullet Z^l \Rightarrow k \cdot \mu_w = \mu_l \Rightarrow \mu_l = k \cdot \alpha.$$

Therefore, we also have

$$(k+1) \cdot \alpha = \mu_0.$$

Finally,

$$(\mu, \mu_0) = \alpha(e^s, k+1).$$

□

Theorem 5.5. *If*

(a) $K(S)$ is a minimal colorful cut;

(b) $K(S)$ induces exactly one T-label X, i.e. $\mathcal{T}(S) = \{X\}$; and

(c) X is a k-T-label, but it is not a (k+1)-T-label induced by $K(S)$, i.e. $X \in \mathcal{T}^k(S)$, $X \notin \mathcal{T}^{k+1}(S)$;

then the face $F^{\text{k-T-label}}(S,X)$ is a facet of $P^{CCut}(G)$.

Proof. *It follows from Lemma 5.4 that the unique solution of (5.20) verifies (5.21). In this case, the provided solution vectors Z^l, for $l \in \{1,2,\cdots,m\}$, are affinely independent and verify the conditions (a), (b), and (c). Therefore, dim($F^{\text{k-T-label}}(S,X)) = m-1$, and $F^{\text{k-T-label}}(S,X)$ is a facet of $P^{CCut}(G)$.* □

5.6 Polyhedral comparisons

This section aims to achieve the following two objectives: 1) to examine the effect of adding the tree search (4.4), T-label (5.12), and k-T-label (5.17) inequalities to the polytope $P^{\text{CCut}}(G)$; and 2) to compare $P^{\text{CCut}}(G)$ with the polytopes described by the formulations EC (2.8-2.12) and DCut (2.2-2.7), presented respectively in Sections 2.6 and 2.5. The remainder of this section uses superscripts to identify when a formulation is strengthened by some cut. Table 5.1 shows the superscripts and their associated constraints.

Table 5.1: Superscripts and constraints associated with each formulation

		Constraint/Formulation		
Superscript	**Constraint Name**	**DCut**	**EC**	**CCut**
t	Tree search	(2.14)	(2.15)	(4.4)
s	Strong linkage	(2.16)	(2.17)	-
n	Node label	(2.18)	(2.18)	-
l	T-label	-	-	(5.12)
k	k-T-label	-	-	(5.17)

Theorem 5.6.

$$P^{CCut^{tlk}}(G) \subset P^{CCut^{tl}}(G) \subset P^{CCut^{t}}(G) \subset P^{CCut}(G). \tag{5.22}$$

Proof. *Since CCut^t is exactly the CCut formulation with the addition of the tree search inequalities (4.4), we have $P^{CCut^t}(G) \subseteq P^{CCut}(G)$. By the same reasoning, $P^{CCut^{tlk}}(G) \subseteq P^{CCut^{tl}}(G) \subseteq P^{CCut^t}(G)$. The remainder of the proof uses the three graphs shown in Fig. 5.4.*

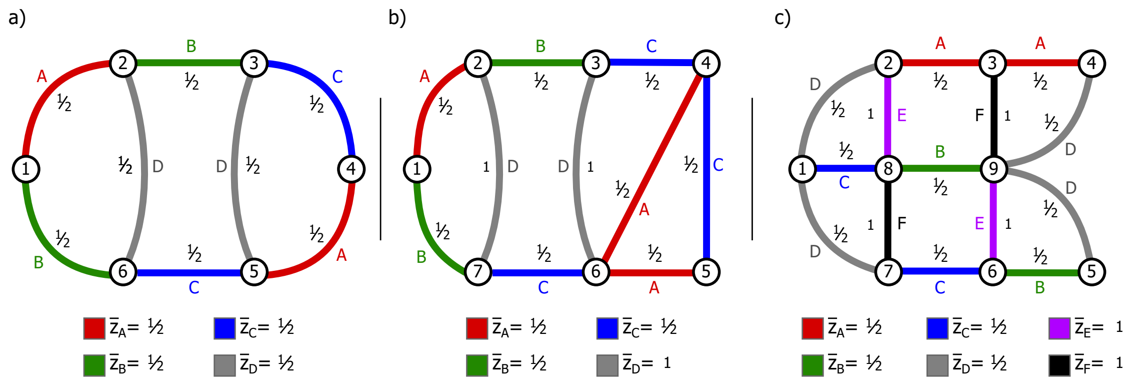


Figure 5.4: LR solutions that are feasible for the CCut formulation but are either **(a)** infeasible for CCut^t, **(b)** infeasible for CCut^{tl}, or **(c)** infeasible for CCut^{tlk}

The solution given in Fig. 5.4a is feasible for the CCut formulation; however, it violates the associated tree search inequalities (4.4). Thus, $P^{CCut^t}(G) \subset P^{CCut}(G)$. The solution given in Fig. 5.4b is feasible for the formulation CCut^t; however, the label C is a T-label induced by the colorful cut $K(\{1\}) = \{A,B\}$, and the solution violates the associated T-label constraint (5.12). In this case, $P^{CCut^{tl}}(G) \subset P^{CCut^t}(G)$. The solution given in Fig. 5.4c is feasible for the formulation CCut^{tl}; however, the label D is a 2-T-label induced by the colorful cut $K(\{1,2,7,8\}) = \{A,B,C\}$, and the solution violates the associated k-T-label constraint (5.17). Thus, $P^{CCut^{tlk}}(G) \subset P^{CCut^{tl}}(G)$, and the proof is concluded. □

Let P_z denote the projection of some polytope P on the Z variable space. For polytopes defined in the (Z,X) variable space, $P_z := \{Z \in \mathbb{R}^{|L|} \mid (Z,X) \in P\}$, whereas for polytopes defined in the (Z,Y) variable space, $P_z := \{Z \in \mathbb{R}^{|L|} \mid (Z,Y) \in P\}$. In the following, the solutions presented in Fig. 5.5 are used to prove Lemmas 5.5 and 5.6. Observe that the inequality associated with the colorful cut $K(\{3,4\}) = \{A,B\}$ is violated in both solutions given in Fig. 5.5. In fact, $\bar{z}_A + \bar{z}_B = \frac{1}{4} + \frac{1}{4} = \frac{1}{2} < 1$.

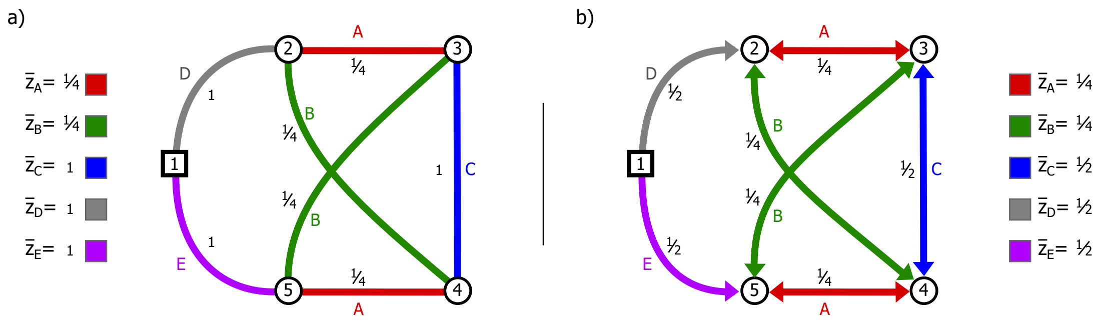


Figure 5.5: LR solutions that are infeasible for the CCut formulation but feasible for EC **(a)** and DCut **(b)**

Lemma 5.5.

$$P_z^h(G) \not\subseteq P^{CCut}(G),\ \forall h \in \{EC, EC^s, EC^n, EC^t, EC^{sn}, EC^{nt}\}. \tag{5.23}$$

Proof. *Consider the LR solution presented in Fig. 5.5a. We have shown that the solution is not feasible for the CCut formulation; yet it does not violate any constraints of the formulations EC, EC^s, EC^n, EC^t, EC^{sn}, or EC^{nt}.* □

Lemma 5.6.

$$P_z^h(G) \not\subseteq P^{CCut}(G),\ \forall h \in \{DCut, DCut^s, DCut^n, DCut^t, DCut^{sn}, DCut^{nt}\}. \tag{5.24}$$

Proof. *Consider the LR solution presented in Fig. 5.5b. We have shown that the solution is not feasible for the CCut formulation, yet it does not violate any constraints of the formulations DCut, $DCut^s$, $DCut^n$, $DCut^t$, $DCut^{sn}$, or $DCut^{nt}$.* □

Let $P_{zx}^{\text{CCut}}(G) := conv\{(Z,X) \in \mathbb{R}^{|L|+|E|} \mid Z \in P^{\text{CCut}}(G)$ and $x_e = z_{l(e)}, \forall e \in E\}$ be the extension of the polytope $P^{\text{CCut}}(G)$ to the (Z,X) variable space.

Theorem 5.7.

$$P_{zx}^{CCut}(G) \subset P^h(G),\ \forall H \in \{EC, EC^s, EC^n, EC^{sn}\}. \tag{5.25}$$

Proof. *The first part of the proof is given by Lemma 5.5. Additionally, we need to prove that (Z,X) respects all EC constraints as well as those of the strong linkage and node labels. Con-*

straints (2.10) and strong linkage (2.17) are respected by the definition of $P_{zx}^{CCut}(G)$. The node label constraints (2.18) are respected because they are a subset of the colorful cut inequalities (4.2). Finally, since $x_e = z_{l(e)}, \forall e \in E$, we have

$$\sum_{e\in\delta(S)} x_e \geq \sum_{l\in K(S)} z_l \geq 1 \geq \varepsilon, \qquad \forall S \subset V, S \neq \emptyset,$$

and inequalities (2.9) are respected. □

Let $P_{zy}^{\text{CCut}}(G) := conv\{(Z,Y) \in \mathbb{R}^{|L|+|A|} \mid Z \in P^{\text{CCut}}(G) \text{ and } y_a = z_{l^a(a)}, \forall a \in A\}$ be the extension of the polytope $P^{\text{CCut}}(G)$ to the (Z,Y) variable space.

Theorem 5.8.

$$P_{zy}^{CCut}(G) \subset P^h(G),\ \forall h \in \{DCut^s, DCut^{sn}\}. \tag{5.26}$$

Proof. *The first part of this proof is given by Lemma 5.6. Additionally, we need to prove that (Z,Y) respects all DCut constraints as well as those of the strong linkage and node labels. Note that the strong linkage constraints and inequalities (2.5) cannot coexist. Constraints (2.4) and strong linkage (2.16) are respected by the definition of $P_{zy}^{CCut}(G)$. The node label constraints (2.18) are respected because they are a subset of the colorful cut inequalities (4.2). Finally, since $y_a = z_{l^a(a)}, \forall a \in A$, we have*

$$\sum_{a\in\delta^-(S)} y_a \geq \sum_{l\in K(S)} z_l \geq 1, \qquad \forall S \subseteq V\backslash\{r\}, S \neq \emptyset,$$

and inequalities (2.3) are respected. □

Before proposing the next theorem, we provide an algorithm that extends the polytope $P^{\text{CCut}}(G)$ with the aim to show that every (extended) feasible solution for the CCut formulation is also feasible for the DCut formulation. Furthermore, we discuss some properties of the procedure and introduce the notation used to prove Theorem 5.9.

Let $v_0, v_1, \cdots, v_{n-1}$ be any arbitrary ordering of the nodes in the set V, for $n = |V|$. For convenience, let $r = v_0$ be the root node of the DCut formulation. Consider Algorithm 5.1, where $Z = (z_l)_{l\in L} \in P^{\text{CCut}}(G)$.

Algorithm 5.1: Extension of Z to (Z,Y^*)

```
Procedure Ext($G=(V,E,L)$, $Z$)
    $D^1=(V,A,Y^1) \leftarrow$ BuildDigraph($G,Z$);
    for $i \leftarrow 1,2,\cdots,n-1$ do
        $F^i \leftarrow$ MaxFlow( $r \rightarrow v_i$, $D^i$ );
        $F^i \leftarrow$ RemovePositiveFlowCircuits( $F^i$ );
        for each $a \in A$ do $y_a^{i+1} \leftarrow \mathrm{Min}(y_a^i, 1-f_{\bar{a}}^i)$;
        $D^{i+1} \leftarrow (V,A,Y^{i+1})$;
    for each $a \in A$ do $y_a^* \leftarrow f_a^{m(a)} = \mathrm{Max}(f_a^1, f_a^2, \cdots, f_a^{n-1})$;
    return $(Z,Y^*)$;
```

Let A be a set of arcs, and let $Y=(y_a)_{a\in A}$ denote a vector of arc capacities. On line 2, $D^1=(V,A,Y^1)$ denotes the directed graph derived from G in which each edge $e\in E$ is replaced by two opposite arcs $a,\bar{a}\in A$, with respective weights of $0\le y_a^1=y_{\bar{a}}^1=z_{l^a(a)}=z_{l^a(\bar{a})}=z_{l(e)}\le 1$. The loop of lines 3 through 7 aims to find a set of maximum flows $F^i=(f_a^i)_{a\in A}$, from r to v_i, $\forall v_i\in V\backslash\{r\}$.

On line 4, MaxFlow$(r\rightarrow v_i, D^i)$ is a procedure that computes one optimal flow vector F^i of the maximum flow problem with the node source r and the destination v_i in the digraph D^i, considering y_a^i as the capacity of the arc a. For the case in which F^i contains any positive flow circuits, it is well known that they may be removed without changing the maximum flow value. Note that because of the positive flow circuit removal, for any $a\in A$, either $f_a^i=0$, $f_{\bar{a}}^i=0$, or $f_a^i=f_{\bar{a}}^i=0$.

For any $a\in A$, let $m(a)=\underset{i=1,2,\cdots,n-1}{argmax}\ (f_a^i)$ be the index of the greater flow that passed through the arc a. On line 6, all arc capacities are updated, and the digraph of the next iteration is created on line 7. Observe that the y series

$$\begin{cases} y_a^1=z_{l(e)}, \\ y_a^{i+1}=\mathrm{Min}(y_a^i, 1-f_{\bar{a}}^i), \quad \text{for } i\ge 1, \end{cases}$$

are non-increasing since $y_a^{i+1}\le y_a^i$. Finally, the return value (Z,Y^*) is the CCut solution vector Z extended with the variables Y^*. In the following, we state Lemmas 5.7 and 5.8 and use them to prove Theorem 5.9.

Lemma 5.7. *For each* $i=1,2,\cdots,n-1$,

$$\sum_{a\in\delta^-(S)} y_a^i \geq 1, \qquad \forall S \subseteq V\backslash\{r\}, v_i \in S \;\Rightarrow \sum_{a\in\delta^-(S)} y_a^* \geq 1, \qquad \forall S \subseteq V\backslash\{r\}, v_i \in S.$$

Proof. *If the left side of the lemma is true, then $F^i \leftarrow MaxFlow(r \rightarrow v_i, D^i)$, and from the min-cut max-flow theorem, it follows that the maximum flow from r to v_i is greater than 1. Since $y_a^* \leftarrow f_a^{m(a)} = Max(f_a^1, f_a^2, \cdots, f_a^i, \cdots, f_a^{n-1})$, $\forall a \in A$, the right side of the lemma is also true.* □

Lemma 5.8. *$y_a^j \geq f_a^i$, $\forall j$ such that $n-1 > j > i$.*

Proof. *If $f_a^i = 0$, the lemma holds. The lemma also holds if $j = i+1$ since $f_a^i \neq 0 \Rightarrow f_{\overline{a}}^i = 0 \Rightarrow y_a^{i+1} = y_a^i$.*

Once $f_a^i \neq 0$, $y_{\overline{a}}^{i+1} \leq 1 - f_a^i$, and from the non-increasing property of the y series $f_{\overline{a}}^k \leq y_{\overline{a}}^{i+1} \leq 1 - f_a^i$, for any $k \geq i+1$. Because of the nature of the function $Min(y_a^i, 1 - f_{\overline{a}}^i)$, the minimum possible value of y_a^{i+1} is attained when $f_{\overline{a}}^i$ is at maximum. Because $f_{\overline{a}}^k \leq 1 - f_a^i$, $y_a^{k+1} \geq 1 - 1 + f_a^i \Rightarrow y_a^{k+1} \geq f_a^i$. Once $k \geq i+1$, replacing k with j results in the lemma also holding for any $j \geq i+2$. □

Let $P_{zy^*}^{\text{CCut}}(G) := conv\{Ext(G,Z) \mid Z \in P^{\text{CCut}}(G)\}$ be the extension of the polytope $P^{\text{CCut}}(G)$ to the (Z,Y) variable space.

Theorem 5.9.

$$P_{zy^*}^{CCut}(G) \subset P^{DCut}(G). \tag{5.27}$$

Proof. *The first part of the proof is given by Lemma 5.6; the second part consists in proving that $Ext(G,Z) = (Z,Y^*) \in P^{DCut}$, $\forall Z \in P^{CCut}(G)$. To this end, we must show that (Z,Y^*) respects all DCut constraints (see Section 2.5).*

- ***Constraints (2.4):*** *We have $f_a^{m(a)} \leq y_a^{m(a)}$ because of the capacity constraint of the flow. Thus, from the non-increasing property of the y series,*

  $$y_a^* = f_a^{m(a)} \leq y_a^{m(a)} \leq y_a^{m(a)-1} \leq \cdots \leq y_a^1 = z_{l^a(a)}, \forall a \in A,$$

  and Constraints (2.4) are respected.

- ***Constraints (2.5):*** *If $m(a) = m(\overline{a})$, then $f_a^{m(a)} + f_{\overline{a}}^{m(\overline{a})} = 0 \leq 1$ because positive flow circuits are removed. Thus, $y_a^* + y_{\overline{a}}^* = 0 \leq 1$.*

  This remains the case when $m(a) < m(\overline{a})$ (the case $m(a) > m(\overline{a})$ is symmetric). For *this case, $m(a)+1 \leq m(\overline{a})$. However, $f_{\overline{a}}^{m(\overline{a})} \leq y_{\overline{a}}^{m(\overline{a})} \leq y_{\overline{a}}^{m(a)+1}$ due to the flow capacity constraint and the non-increasing property of the y series.*

From the definition of the y series, we also have $y_{\overline{a}}^{m(a)+1} \leq 1 - f_a^{m(a)}$. Thus,

$$f_{\overline{a}}^{m(\overline{a})} \leq 1 - f_a^{m(a)} \Rightarrow f_{\overline{a}}^{m(\overline{a})} + f_a^{m(a)} \leq 1 \Rightarrow y_{\overline{a}}^* + y_a^* \leq 1,$$

and Constraints (2.5) are satisfied.

- ***Constraints (2.3):*** *We need to prove that*

$$\sum_{a\in\delta^-(S)} y_a^* \geq 1, \qquad \forall S \subseteq V\backslash\{r\}, S \neq \emptyset. \tag{5.28}$$

To this end, we use mathematical induction on Algorithm 5.1, from which we derived the following inequalities:

$$\sum_{a\in\delta^-(S)} y_a^i \geq 1, \qquad \forall S \subseteq V\backslash\{r\}, v_i \in S. \tag{5.29}$$

Observe that, if (5.29) is true for every $i = 1, 2, \cdots, n-1$, then it follows from Lemma 5.7 that (5.28) is also true.

Since $y_a^1 = z_{l(e)}$, $\forall a \in A$, and Z is feasible for CCut, the result holds for $i = 1$. Now let us assume, by induction hypothesis, that (5.29) holds for every $i \leq k-1$, for $k \leq n-1$, and we prove that it is still true for $i = k$. Let S^k denote any set of vertices such that $S^k \subseteq V\backslash\{r\}$ and $v_k \in S^k$. If

$$\sum_{a\in\delta^-(S^k)} y_a^k = \sum_{a\in\delta^-(S^k)} y_a^1,$$

then the result holds for the same reason it holds for $i = 1$. Furthermore, if any $v_i \in S^k$, for $i < k$, then the result holds by induction hypothesis. For the remaining case we have

$$\sum_{a\in\delta^-(S^k)} y_a^k < \sum_{a\in\delta^-(S^k)} y_a^1, \text{ and } v_i \notin S^k, \forall i < k.$$

In this case, at least one arc capacity y_a^k, $a \in \delta^-(S)$ was changed compared with y_a^1. Consider the largest value for w, with $w < k$, such that the flow F^w changes some capacity y_a^{w+1}, $a \in \delta^-(S)$. In this case, based on flow conservation, we have

$$\sum_{a \in \delta^-(S^k)} f_a^w = \sum_{\overline{a} \in \delta^+(S^k)} f_{\overline{a}}^w$$

because $v_w \notin S^k$. Let $b \in \delta^-(S^k)$ be the arc whose capacity y_b^{w+1} was changed. This means that $f_{\overline{b}}^w \neq 0$ and $f_b^w = 0$. Then, from Lemma 5.8, it follows that

$$\sum_{\substack{a \in \delta^-(S^k) \\ a \neq b}} y_a^k \geq \sum_{\substack{a \in \delta^-(S^k) \\ a \neq b}} f_a^w = \sum_{a \in \delta^-(S^k)} f_a^w = \sum_{\overline{a} \in \delta^+(S^k)} f_{\overline{a}}^w .$$

Moreover, $y_b^{w+1} = 1 - f_{\overline{b}}^w$ and $f_{\overline{b}}^w = 1 - y_b^{w+1}$. Thus,

$$\sum_{\substack{a \in \delta^-(S^k) \\ a \neq b}} y_a^k \geq f_{\overline{b}}^w + \sum_{\substack{\overline{a} \in \delta^+(S^k) \\ \overline{a} \neq \overline{b}}} f_{\overline{a}}^w \Rightarrow \sum_{\substack{a \in \delta^-(S^k) \\ a \neq b}} y_a^k \geq 1 - y_b^{w+1} + \sum_{\substack{\overline{a} \in \delta^+(S^k) \\ \overline{a} \neq \overline{b}}} f_{\overline{a}}^w .$$

Because $w < k$ is the index of the last iteration, which changed the capacity y_b^{w+1}, $y_b^k = y_b^{w+1}$. Finally,

$$\sum_{a \in \delta^-(S^k)} y_a^k = \sum_{\substack{a \in \delta^-(S^k) \\ a \neq b}} y_a^k + y_b^{w+1} \geq 1 + \sum_{\substack{\overline{a} \in \delta^+(S^k) \\ \overline{a} \neq \overline{b}}} f_{\overline{a}}^w \geq 1.$$

□

Corollary 5.1.

$$P_{zy^*}^{CCut}(G) \subset P^{DCut^n}(G). \tag{5.30}$$

Proof. *From Theorem 5.9, we have $P_{zy^*}^{CCut}(G) \subset P^{DCut}(G)$. Furthermore, the node label constraints (2.18) are a subset of the colorful cut inequalities (4.2).* □

The section above compared the polytope $P^{\text{CCut}}(G)$ with the polytopes defined by the EC and DCut formulations as well as with their variations. Figure 5.6 summarizes the results described in this section.

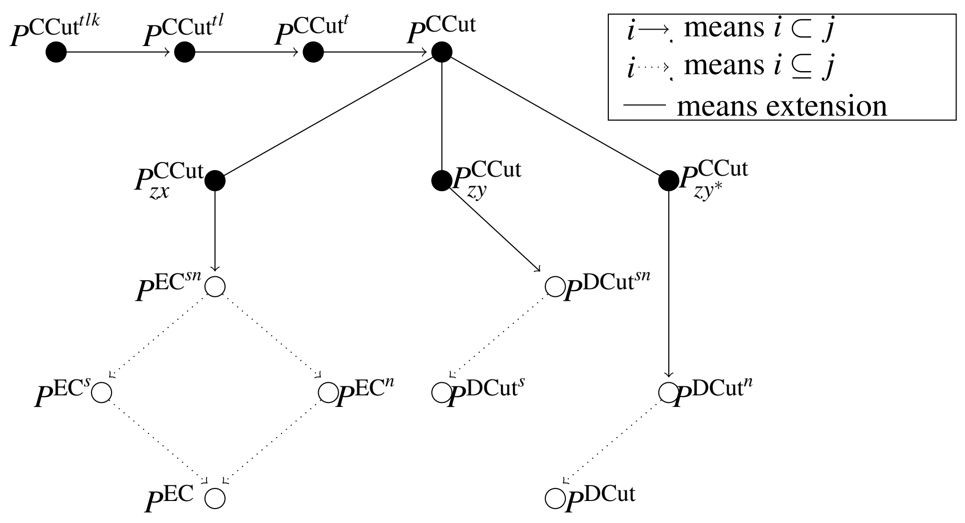


Figure 5.6: Polytope inclusion diagram summarizing the results of Section 5.6

5.7 CCut convex hull study

Taking into account all inequalities introduced in the previous sections, it may be interesting to determine whether a complete description of the polytope $P^{\text{CCut}}(G)$ is attained. Figure 5.7 shows that it is not the case. Indeed, the basic feasible solution of this example satisfies all inequalities (colorful cuts, tree search, T-labels, and k-T-labels), yet still contains fractional variables. However, the graph in Fig. 5.7 might serve as a starting point in the search for new families of facet-defining inequalities.

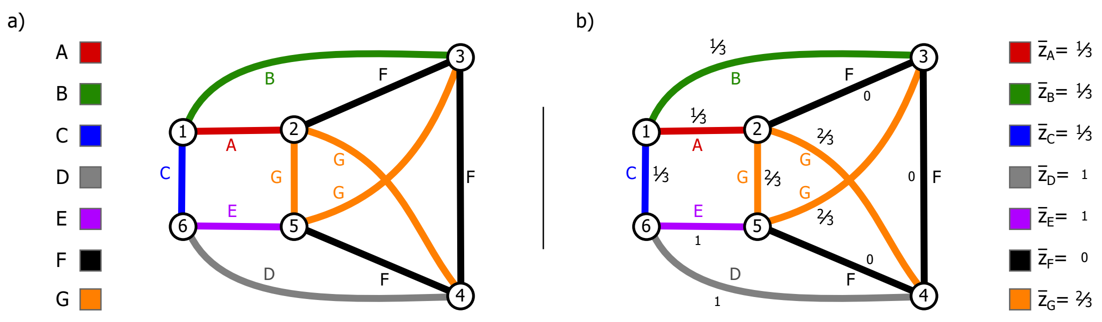


Figure 5.7: Small convex hull counter-example. **(a)** Example graph. **(b)** A basic feasible solution with fractional label variables

Furthermore, expressions (5.31) through (5.39) present the inequalities corresponding to the instance given in Fig. 5.7. Inequalities (5.31) through (5.37) are facet defining for the minimal colorful cuts of this graph. Inequality (5.38) represents the colorful cut $K(\{1,6\}) = \{A,B,D,E\}$ that induces the T-label C. Inequality (5.39) is the corresponding tree search inequality.

colorful cut: $(S = \{1\})$ $$z_A + z_B + z_C \geq 1 \tag{5.31}$$

colorful cut: $(S = \{2\})$ $$z_A + z_F + z_G \geq 1 \tag{5.32}$$

colorful cut: $(S = \{3\})$ $$z_B + z_F + z_G \geq 1 \tag{5.33}$$

colorful cut: $(S = \{1,2,3\})$ $$z_C + z_F + z_G \geq 1 \tag{5.34}$$

colorful cut: $(S = \{4\})$ $$z_D + z_F + z_G \geq 1 \tag{5.35}$$

colorful cut: $(S = \{5\})$ $$z_E + z_F + z_G \geq 1 \tag{5.36}$$

colorful cut: $(S = \{6\})$ $$z_C + z_D + z_E \geq 1 \tag{5.37}$$

T-label C: $(S = \{1,6\})$ $$z_A + z_B + z_C + z_D + z_E \geq 2 \tag{5.38}$$

tree search: $$z_A + z_B + z_C + z_D + z_E + 3z_F + 3z_G \geq 5 \tag{5.39}$$

5.8 Concluding remarks

In the present chapter, we have provided some interesting results for the CCut polytope, in particular concerning its dimension and its facet compositions. New valid inequalities were introduced, and the conditions in which they define facets have been given. We have proposed polyhedral comparisons between the polytope associated with the state-of-the-art formulations—DCut and EC—and their variations. Our results show that the CCut formulation theoretically performs better with respect to its polytope than all currently available mathematical formulations for the GMLSTP and MLSTP. On the other hand, in Section 5.7, we have shown that all facets introduced here were insufficient to reach the CCut polytope convex hull.

Chapter 6

Improved Exact Methods

Chapter 4 has presented studies on MIP-based exact methods for solving the GMLSTP, such as CCut, a new mathematical formulation for the problem, as well as branch-and-cut algorithms for solving this model. In this chapter we discuss two improvements for the methods previously addressed in Chapter 4. First, we propose a new mathematical model that extends CCut by using the concept of *partitioning cuts*. Then, we introduce a new branching strategy for solving CCut. Lastly, we combine these two approaches into a new branch-and-cut algorithm. We can observe from the computational experiments performed that the new approaches were able to achieve the best results regarding exact methods for the MLSTP so far.

6.1 The partitioning cuts formulation

This section presents the partitioning cuts formulation (PCut), a new mathematical model for solving both the MLSTP and the GMLSTP. As well as the CCut formulation, PCut defines only $|L|$ binary variables. Moreover, as discussed later, PCut can be seen as an extension of CCut. Before introducing the model, it is necessary to formalize the concepts of *proper partitionings*, *partitioning graphs*, and *partitioning cuts*, as well as to discuss an important property related to these concepts.

Definition 6.1. *Given an ELG $G = (V,E,L)$, $P = \{S_1, S_2, \cdots, S_p\}$ is a proper partitioning of the vertices of G, for short partitioning, if (i) $S \subset V$, $S \neq \emptyset$, $\forall S \in P$, (ii) $\cup_{S \in P} S = V$, and (iii) $S_a \cap S_b = \emptyset$, $\forall S_a, S_b \in P, S_a \neq S_b$.*

Definition 6.2. *Given an ELG $G = (V,E,L)$ and a proper partitioning of its vertices $P = \{S_1, S_2, \cdots, S_p\}$, the partitioning graph $G((P)) = (V', E', L)$ is the graph built as follows: for each partition $S \in P$ there is a vertex $v(S) \in V'$, and for each edge $e = (v_1, v_2) \in E$, $v_1 \in S'$,*

$v_2 \in S''$, $S', S'' \in P$, there is an edge $e' = (v(S'), v(S'')) \in E'$, such that $l(e') = l(e)$.

Proposition 6.1. *Let $G = (V, E, L)$ be an edge-labeled graph. G is connected if and only if $G((P))$ is connected for any proper partitioning $P = \{S_1, S_2, \cdots, S_p\}$ of V, the vertices of G.*

Proof. *($\Rightarrow$) Suppose that $G = (V, E, L)$ is connected and $G((P)) = (V', E', L)$ is not connected for some proper partitioning $P = \{S_1, S_2, \cdots, S_p\}$ of V. In this case, $G((P))$ has an empty cut-set $[S', V' \backslash S']$ with $S' = \{v(S_a), v(S_b), \cdots, v(S_w)\}$. Then, by the definition of partitioning graphs, the cut-set $[S'', V \backslash S'']$, for $S'' = S_a \cup S_b \cup \cdots \cup S_w$, is empty on G and G is disconnected. It is a contradiction.*

($\Leftarrow$) Suppose that $G((P)) = (V', E', L)$ is connected for any proper partitioning P of the vertices of G, and $G = (V, E, L)$ is disconnected, that is to say G has an empty cut-set $[S, \overline{S} = V \backslash S]$. Considering the partitioning $P = \{S, \overline{S}\}$, we have that $G((P))$ has an edge $e' = (v(S), v(\overline{S}))$, otherwise it is not connected. However, from the definition of partitioning graphs, the edge e' could only be originated from an edge of G that has one endpoint in S and other in $\overline{S}$. In this case, the cut-set $[S, \overline{S}]$ is not empty and G is connected. $\square$

As a consequence of Proposition 6.1, we have that the connectivity of an ELG G could be verified by checking the connectivity of $G((P))$ for every possible proper partitioning of the vertices of G. In addition, observe that $G((P))$ itself is an ELG (possibly an edge-labeled multigraph) and thence, all the properties that hold for this kind of graph also holds for $G((P))$. We are particularly interested in Propositions 3.2 and 3.8, respectively the monochromatic cycle removal property and the lower bound on the number of edges necessary to connect an ELG.

Given an ELG $G = (V, E, L)$ and a proper partitioning of its vertices $P = \{S_1, S_2, \cdots, S_p\}$, let $G^*((P))$ denote the monochromatic-cycles free ELG obtained by applying the procedure MCR (refer to Algorithm 3.1) on $G((P))$.

Definition 6.3. *A partitioning cut (or a PCut inequality) is the tree search inequality (4.4) derived from a partitioning graph $G^*((P))$.*

Figure 6.1 illustrates the concept of partitioning cuts as well as the definitions necessary to its understanding. Fig. 6.1a presents a small edge-labeled graph $G = (V, E, L)$ and a partitioning $P = \{S_1 = \{1,2\}, S_2 = \{3,4\}, S_3 = \{5,6\}, S_4 = \{7,8\}\}$. Fig. 6.1b depicts the graph $G((P)) = (V', E', L)$, $V' = \{s1 = v(S_1), s2 = v(S_2), s3 = v(S_3), s4 = v(S_4)\}$. Fig. 6.1c shows the graph $G^*((P))$, obtained after applying the monochromatic-cycle removal procedure on $G((P))$. The tree search inequality (4.4) derived from the graph $G^*((P))$, thence a partitioning cut, is $2z_A + z_B + z_D + z_F \geq 3$.

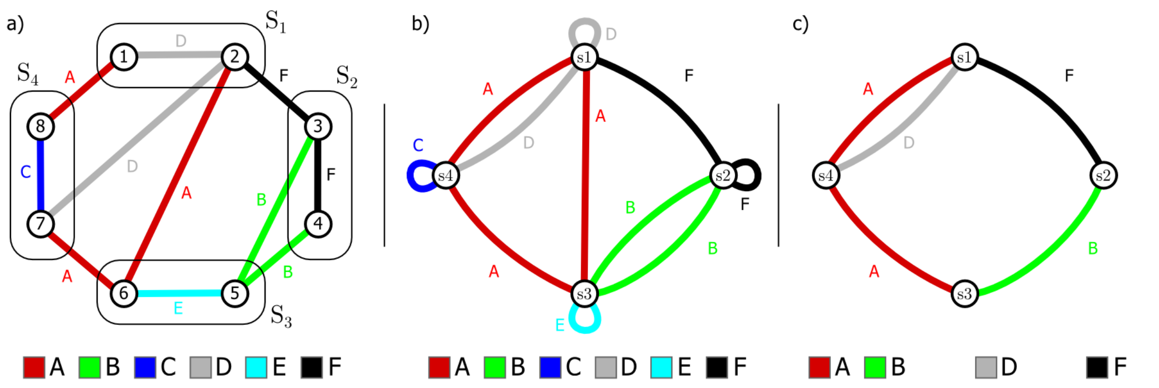


Figure 6.1: Proper partitioning, partitioning graphs, and partitioning cuts. **(a)** An edge-labeled graph G and a partitioning $P = \{S_1 = \{1,2\}, S_2 = \{3,4\}, S_3 = \{5,6\}, S_4 = \{7,8\}\}$. **(b)** The graphs $G((P))$ and **(c)** $G^*((P))$. The partitioning cut derived from $G^*((P))$ is $2z_A + z_B + z_D + z_F \geq 3$

The partitioning cuts formulation (PCut), presented in the program (6.1) through (6.3), is derived directly from Proposition 6.1 and Definition 6.3. As well as CCut, the model also defines only the group of binary variables $z_l \in \{0,1\}$, for which $z_l = 1$ means that every edge with the label l is in the solution. The objective function (6.1) minimizes the number of labels in the solution. Let $E(K,H)$, $K \subseteq L$, denote the set of edges with label in K on the ELG H, and $\mathcal{P}(G)$ be the set of all possible proper partitionings of the vertices of the ELG G. Each constraint of the set (6.2) ensures that $G^*((P))$ has a minimum number of edges to be connected. Finally, the set of constraints (6.3) defines the domain of the variables.

$$\text{Minimize} \sum_{l \in L} z_l \tag{6.1}$$

$$\text{s.t.} \quad \sum_{l \in L} |E\left(\{l\}, G^*((P))\right)| \cdot z_l \geq |P| - 1, \qquad \forall P \in \mathcal{P}(G), \tag{6.2}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{6.3}$$

Alternatively, we can express the partitioning cuts inequalities (6.2) with respect to the cardinality of the partitioning set P, as presented in the inequalities (6.4) to (6.8). Let PCut^n denote the set of partitioning cuts with $|P| = n$.

$$(PCut^2) \qquad \sum_{l \in L} |E(\{l\}, G^*((P)))| \cdot z_l \geq 1, \qquad \forall P \in \mathcal{P}(G), |P| = 2, \tag{6.4}$$

$$(PCut^3) \qquad \sum_{l \in L} |E(\{l\}, G^*((P)))| \cdot z_l \geq 2, \qquad \forall P \in \mathcal{P}(G), |P| = 3, \tag{6.5}$$

$$(PCut^4) \qquad \sum_{l \in L} |E(\{l\}, G^*((P)))| \cdot z_l \geq 3, \qquad \forall P \in \mathcal{P}(G), |P| = 4, \tag{6.6}$$

$$\vdots$$

$$(PCut^{|V|-1}) \qquad \sum_{l \in L} |E(\{l\}, G^*((P)))| \cdot z_l \geq |V| - 2, \qquad \forall P \in \mathcal{P}(G), |P| = |V| - 1, \tag{6.7}$$

$$(PCut^{|V|}) \qquad \sum_{l \in L} |E(\{l\}, G^*((P)))| \cdot z_l \geq |V| - 1, \qquad \forall P \in \mathcal{P}(G), |P| = |V|. \tag{6.8}$$

Observe that PCut^2 is exactly the colorful cuts inequalities (4.2). In this sense, we have that the PCut formulation can be seen an extension of the CCut model. Moreover, since all PCut inequalities are valid for both the MLSTP and the GMLSTP (from Propositions 6.1, 3.2 and 3.8), we can ensure the corretude of the formulation. In addition, note that PCut$^{|V|}$ is exactly the tree search inequality (4.4) derived from G.

Further, since the set of inequalities PCut^2 and the CCut inequalities (4.2) are equivalent, they can be separated exactly as discussed in Section 4.2. Unfortunately, we were not able to provide a MIP-based exact separation algorithm for the remaining PCut^n inequalities. We leave this as an open question in this work. Notwithstanding, we have made an effort to evaluate the effectiveness of the PCut inequalities. In this sense, we have generated 100 graphs with $|V| = 14$ and separated the PCut inequalities by enumeration.

Table 6.1 shows the results of the experiments on the impact of PCut inequalities on the linear relaxation of CCut. The first column identifies the formulation, while each of the remaining ones presents the results for one dataset, which is a set of 10 ELGs with the same edge density d. Further, we also provide the average number of non-empty labels $\overline{|L|}$ of the dataset. Each line of the table reports the average results over the ten graphs in each dataset. The first three lines are the reference values: The lines *OPT*, *CCut^t*, and *Gap*, refer, respectively, to the optimal integer solution, to the linear relaxation of CCut formulation with the tree search constraint but without any separation method, and to the relative difference between them ($Gap = (OPT - CCut^t)/OPT$). The remaining lines are divided in groups of three and each group reports the results of separating a subset of PCut inequalities[1]. The first line of the group gives the average linear relaxation obtained, the second line reports its gap

[1]Unfortunately, we were not able to separate the sets PCut^n for $n \in \{6, 7, 8, 9\}$ due to its size.

with relation to the value in *OPT*, and the third line report the number of cuts added to the model. PCut* means the separation of all PCut inequalities. PCut^s refers to the subgroup of all PCut inequalities composed only by unitary partitions, except for one big partition, e.g. $P = \{\{v_1, v_2, v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}\}$.

From the results of the Table 6.1 we have that the PCut* inequalities have a substantial impact on the linear relaxation of CCut. Indeed, for sets with $d \leq 0.4$ the Gap is $\leq 1.1\%$. Another point that caught our attention was the reduced number of cuts added to the model. For instance, 3.4 cuts (in average) were enough to lead the gap to 0% in line PCut* and column $d = 0.20$. Considering the groups individually, we have that PCut^{10} performed better for low density graphs while PCut^s obtained the best results for medium-to-high density graphs. Therefore, the results presented justifies further studies on the PCut inequalities.

Table 6.1: Impact of PCut inequalities on the linear relaxation of CCut

	Dataset identification									
	d=0.20	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	**0.60**	**0.80**	**1.0**
	$\overline{\vert L\vert}$=12.9	**16.4**	**19.8**	**23.5**	**28.1**	**31.4**	**31.6**	**35.4**	**40.8**	**44.7**
OPT	9.9	9.4	8.1	8.5	7.9	8.1	6.3	5.9	4.7	4.0
CCut^t	7.900	7.550	6.450	6.517	6.308	6.317	4.900	4.503	3.464	2.831
Gap	20.2%	19.7%	20.4%	23.3%	20.1%	22.0%	22.2%	23.7%	26.3%	29.2%
PCut^2	8.758	7.900	6.883	6.980	6.569	6.492	5.290	4.702	3.586	2.917
Gap	11.5%	16.0%	15.0%	17.9%	16.8%	19.9%	16.0%	20.3%	23.7%	27.1%
Cuts	4.2	3.5	3.2	3.9	3.8	2.6	3.5	3.5	3.1	2.6
PCut^3	9.336	8.284	7.164	7.274	6.841	6.820	5.515	4.876	3.666	3.002
Gap	5.7%	11.9%	11.6%	14.4%	13.4%	15.8%	12.5%	17.4%	22.0%	25.0%
Cuts	9.6	15.1	11.8	14.1	15.4	11.5	11.6	12.1	10.7	9.5
PCut^4	9.606	8.538	7.390	7.496	7.007	7.006	5.620	4.963	3.729	3.034
Gap	3.0%	9.2%	8.8%	11.8%	11.3%	13.5%	10.8%	15.9%	20.7%	24.1%
Cuts	15.1	23.0	21.8	27.4	25.7	25.6	27.4	18.2	24.1	18.8
PCut^5	9.731	8.761	7.545	7.664	7.166	7.209	5.696	5.057	3.758	3.058
Gap	1.7%	6.8%	6.9%	9.8%	9.3%	11.0%	9.6%	14.3%	20.0%	23.6%
Cuts	16.5	36.4	28.5	46.7	38.4	35.5	34.0	27.0	31.6	26.4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
PCut^{10}	9.900	9.400	7.917	8.500	7.729	7.545	5.465	5.063	3.835	3.076
Gap	**0.0%**	**0.0%**	**2.3%**	**0.0%**	**2.2%**	6.8%	13.3%	14.2%	18.4%	23.1%
Cuts	13.2	9.1	4.7	13.0	15.4	15.7	15.3	22.7	15.6	21.7
PCut^{11}	9.900	9.400	7.600	8.129	7.208	7.165	5.410	4.998	3.738	3.041
Gap	**0.0%**	**0.0%**	6.2%	4.4%	8.8%	11.5%	14.1%	15.3%	20.5%	24.0%
Cuts	6.1	3.4	3.3	5.0	5.2	9.4	6.4	10.8	7.5	13.9
PCut^{12}	9.600	8.900	7.100	7.480	6.960	6.815	5.271	4.893	3.512	3.027
Gap	3.0%	5.3%	12.3%	12.0%	11.9%	15.9%	16.3%	17.1%	25.3%	24.3%
Cuts	2.9	1.9	2.0	3.1	2.7	3.3	2.5	4.8	5.1	7.7
PCut^{13}	8.900	8.000	6.650	7.050	6.625	6.450	5.150	4.718	3.140	2.966
Gap	10.1%	14.9%	17.9%	17.1%	16.1%	20.4%	18.3%	20.0%	33.2%	25.9%
Cuts	0.9	0.9	0.4	1.1	0.5	1.2	1.2	1.4	2.1	2.3
PCut^s	9.650	9.267	7.900	8.250	7.450	7.823	5.858	5.270	3.871	3.126
Gap	2.5%	1.4%	2.5%	2.9%	5.7%	**3.4%**	**7.0%**	**10.7%**	**17.6%**	**21.8%**
Cuts	2.9	3.3	4.1	7.8	7.5	11.5	20.4	23.0	17.1	23.2
PCut*	9.900	9.400	8.007	8.500	7.825	7.903	5.919	5.319	3.908	3.159
Gap	**0.0%**	**0.0%**	**1.1%**	**0.0%**	**0.9%**	**2.4%**	**6.0%**	**9.9%**	**16.8%**	**21.0%**
Cuts	3.4	5.1	6.3	8.2	12.1	14.7	26.4	26.0	20.8	27.2

Due to its exponential size, separating the PCut set of inequalities (6.2) by enumeration is not practical for medium to large size input graphs. In this sense, we have proposed a greedy heuristic, denominated PCut^h, for separating these cuts. It is a deterministic multistart procedure that uses $|V|-2$ partitionings as starting points. Though, before introducing the complete heuristic, we need to introduce the *generatePartitioning* routine and to define two neighborhood structures: $\mathcal{NS}$ and $\mathcal{NM}$.

Given an integer parameter $2 \leq k \leq |V|-1$, the routine *generatePartitioning* is responsible for generating an initial partitioning $P = \{S_1, S_2, \cdots, S_p\}$, with $p = |V|-k+1$, of the vertices of G. All partitions are unitary, except for S_1, that has k elements. The selection of vertices is made by *id*: the first k vertices are placed in S_1 while each of the remaining ones are put in a separate partition.

Given a partitioning P and two partitions $S_a, S_b \in P$ such that $S_a \neq S_b$, $v_a \in S_a$, and $v_b \in S_b$, performing a *swap* move on P is to make $S_a \leftarrow (S_a \backslash \{v_a\}) \cup \{v_b\}$ and $S_b \leftarrow (S_b \backslash \{v_b\}) \cup \{v_a\}$. In the same sense, performing a *migrate* move on P is to make $S_a \leftarrow S_a \backslash \{v_a\}$ and $S_b \leftarrow S_b \cup \{v_a\}$. Given that, let $\mathcal{NS}$(P) and $\mathcal{NM}$(P) denote, respectively, the set of all possible swap and migration moves performed from the partitioning P.

Consider $G = (V, E, L)$ the input ELG, and Z^* a solution for the linear relaxation of CCut, the heuristic PCut^h is presented in Algorithm 6.1. The loop of the lines (2 to 10) controls the multistart procedure where the first partitioning with positive violations found is returned. The initial partitioning is generated on line 3. The loop of the lines (5 to 10) controls the number of moves without positive violation that are allowed. Lastly, the loops of the lines 6 and 8 evaluates the solutions from $\mathcal{NS}$ and $\mathcal{NM}$. We have added a memory to the method (line 10) to try to avoid cycles between the partitionings.

Algorithm 6.1: Separation heuristic for PCut inequalities

```
1  procedure PCut$^h$($G = (V, E, L)$, $Z^*$)
2      for bigPartitionSize $\leftarrow 2$ to $|V|-1$ do
3          $s_0 \leftarrow s_1 \leftarrow$ generatePartitioning( bigPartitionSize );
4          if $s_0$ is violated then return $s_0$ ;
5          for $i \leftarrow 1$ to $4 \cdot |V|$ do
6              foreach $s' \in \mathcal{NS}(s_i)$ do
7                  if $s'$ is violated then return $s'$ ;
8              foreach $s' \in \mathcal{NM}(s_i)$ do
9                  if $s'$ is violated then return $s'$ ;
10             $s_{i+1} \leftarrow$ the solution that is closer to be violated in $(\mathcal{NS}(s_i) \cup \mathcal{NM}(s_i)) \backslash \{s_{i-1}\}$;
11     return No cuts found!;
```

Figure 6.2 illustrates the swap and migrate moves. Fig. 6.2a presents an example edge-labeled graph $G = (V,E,L)$ and a partitioning $P = \{S_1 = \{1,2\}, S_2 = \{3,4\}, S_3 = \{5,6\}, S_4 = \{7,8\}\}$. In Fig. 6.2b, there was a migration of the vertex 6 from S_3 to S_4. In Fig. 6.2c there was a swap between the vertices 3 and 5. The PCut inequalities associated with the partitionings of Fig. 6.2a, b, and c are, respectively, $2z_A + z_B + z_D + z_F \geq 3$, $z_A + z_B + z_D + z_E + z_F \geq 3$, and $z_A + z_B + z_D + z_E + 2z_F \geq 3$.

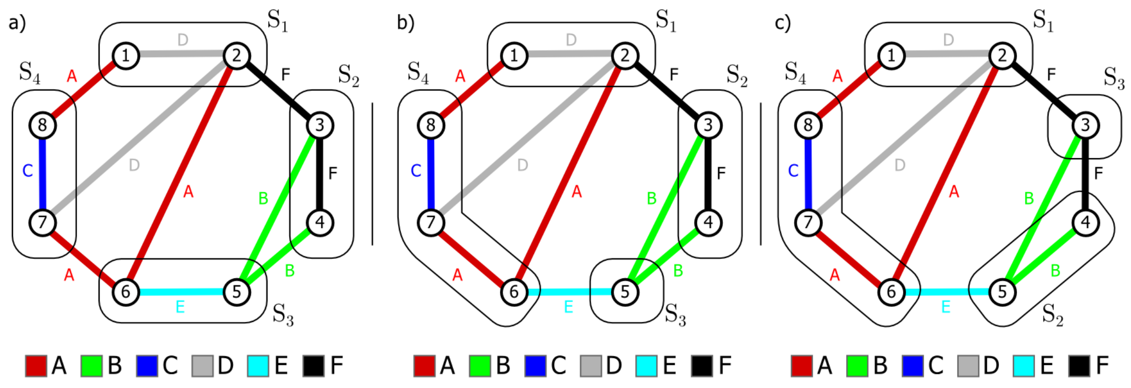


Figure 6.2: Example of swap and migrate moves. **(a)** An edge-labeled graph G and a partitioning $P = \{S_1 = \{1,2\}, S_2 = \{3,4\}, S_3 = \{5,6\}, S_4 = \{7,8\}\}$. **(b)** The partitioning after migrating the vertex 6: $P = \{S_1 = \{1,2\}, S_2 = \{3,4\}, S_3 = \{5\}, S_4 = \{6,7,8\}\}$. **(c)** The partitioning after swapping the vertices 3 and 5: $P = \{S_1 = \{1,2\}, S_2 = \{4,5\}, S_3 = \{3\}, S_4 = \{6,7,8\}\}$

We have evaluated the behavior of the separation heuristic PCut^h on the same graphs of the previous experiment. The results are reported in Table 6.2, which has the same structure of the Table 6.1. The results showed that PCut^h has obtained a satisfactory performance, mainly for instances with low edge density. Observe that the performance deteriorates as the density of the input graphs grows. Fortunately, the worst performance of the CCut formulation, according to the experiments discussed in Section 4.3, is when the input graph has a small density. More detailed experiments are presented in Section 6.3.

Table 6.2: Impact of separating heuristicly the PCut inequalities

	Dataset identification									
	d=0.20	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	**0.60**	**0.80**	**1.0**
	$\overline{\lvert L\rvert}$=12.9	**16.4**	**19.8**	**23.5**	**28.1**	**31.4**	**31.6**	**35.4**	**40.8**	**44.7**
OPT	9.9	9.4	8.1	8.5	7.9	8.1	6.3	5.9	4.7	4.0
PCut^h	9.900	9.233	7.900	8.167	7.340	7.717	5.833	5.298	3.902	3.146
Gap	0.0%	1.8%	2.5%	3.9%	7.1%	4.7%	7.4%	10.2%	17.0%	21.3%
Cuts	3.2	3.2	3.8	7.9	6.9	12.1	20.9	23.3	29.1	27.9

6.2 Branch-and-bound strategies

The Simplex method (Dantzig et al., 1955) is one of the most efficient approaches for solving linear programs in which all the variables are continuous, i.e. they are not required to be integers or binaries. If on the one hand the Simplex method has an exponential worst-case time complexity, on the other hand it performs very well in practice. In contrast, it is necessary to use more sophisticated techniques, such as cutting-planes or branch-and-bound, to deal with integrality constraints. The first consists in solving the linear relaxation of the program and refine it iteratively by adding new linear inequalities, termed cuts, until a solution without fractional variables is found. The second consists in partitioning the solution space into disjoint subspaces and solving the resulting models, possibly repartitioning them. In the traditional branch-and-bound method, after solving the linear relaxation of the model, a variable x with a fractional value $\bar{x}$ is selected and two subproblems are generated: one for $x \leq \lfloor \bar{x} \rfloor$ and another for $x \geq \lceil \bar{x} \rceil$ ($x = 0$ and $x = 1$ when x is binary), what leads to a binary branching tree. Figure 6.3 illustrates a binary branch-and-bound tree on the z variables of CCut formulation.

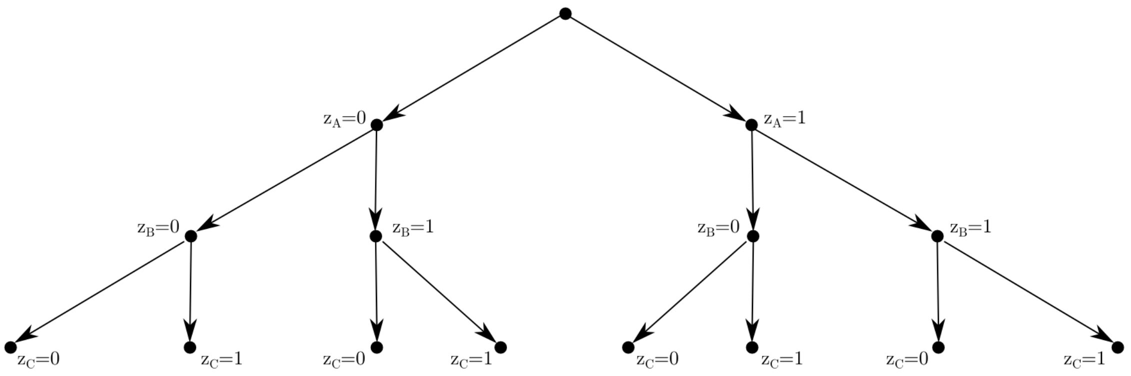


Figure 6.3: A binary branch-and-bound tree on the z variables of CCut formulation

Combining branch-and-bound and cutting-planes leads to the branch-and-cut method, a very powerful technique for solving (mixed) integer linear programs. We have studied three branch-and-cut algorithms for solving the CCut model in Section 4.2, namely BC^A, BC^R, and BC^I. In this section, we discuss alternative branching strategies which can improve the performance of CCut for solving both the MLSTP and the GMLSTP.

From the results reported in Section 4.3 it is possible to observe that the number of nodes visited in the branch-and-bound tree is too big. It is due to the fact that in CCut formulation, setting a variable $z = 1$ has much more impact on the subproblem than setting $z = 0$. Indeed, in the first case a set of edges is added to the solution whereas in the second case it is easy to find a substitute for the label that is forbidden to be in the solution. It could be reinforced by the fact that the number of labels in the solution is commonly much smaller than the number of labels of the input graph. For instance, the average best solution found for the dataset 200-*ld*-250 is 13.8 (refer to Table 4.4), which stands for only 5.52% from the total labels in the input ELGs. Such difference between setting a binary variable to 1 or to 0 leads to highly unbalanced branch-and-bound trees. Figure 6.6(a) depicts the structure of a highly unbalanced branch-and-bound tree.

As an alternative to the traditional branch-and-bound method, based on a binary decision tree, we propose the *colorful cuts branch-and-bound* (CCutBB), which uses the CCut branching strategy, performing the branching phase using a colorful cut inequality (4.2) as a pivot. Given a colorful cut $K(S) = \{A_1, A_2, \cdots, A_k\}$, recall from Definition 4.2 and Proposition 4.1 that at least one label $l \in K(S)$ must be in the solution (and thence to have $z_l = 1$), otherwise the resulting graph will be disconnected. In this sense, we propose to use $K(S)$ as a pivot to partition the problem into k subproblems such that the variables of the subproblem k are set in the following way: $z_k = 1$, and $z_x = 0$, $\forall x < k$. Follows that three questions have to be answered in order to implement the colorful cuts branch-and-bound: (i) which colorful cut should be used as a pivot? (ii) how the labels in the selected pivot should be sorted? And (iii) in which sequence the nodes should be processed?

We have answered these questions empirically. The resulting procedure, namely the colorful cuts branch-and-bound, is presented in Algorithms 6.2, 6.3, and 6.4. Algorithm 6.2 describes the procedure CCutBB, which is the main function. The necessary initializations are performed in lines 2 to 4. The main loop (lines 5 to 17) performs a DFS traversal in the branch-and-bound tree. The next node to be processed is selected in the line 6 and its linear relaxation is solved in line 7, returning a lower bound (LB) and the solution vector Z^*. Two light variants of the MVCA are used in the lines 8 and 9 in order to obtain better upper bounds, making possible to close some nodes earlier. The procedure MVCA' adds all labels fixed to 1 to the solution and then performs the original MVCA. The procedure MVCA" uses Z^*, the solution vector of the linear relaxation of the node, to guide the MVCA. At each iteration, it adds to the solution the label l that has the maximum value of Z_l^*. Once all the remaining labels have $z_l^* = 0$, it turns into the original MVCA. The lines 10 to 13 present the bound conditions and if these conditions are not met, it is necessary to perform a branch on this node (lines 14 to 17) using the procedure *CCutBranch*.

Algorithm 6.3 describes the procedure *CCutBranch*, which is responsible for partitioning the given node into smaller subproblems. The necessary initialization is given in line 2. Then, the procedure *CCutSelect* (described later) is called in order to choose a colorful cut as a pivot (line 3), which is sorted (line 4) with respect to the number of edges possessed by the label. The loop of the lines 5 to 10 creates the new nodes by copying the current one and fixing some of its variables according to the rules described previously. Finally, the new set of nodes is returned (line 11) as a stack.

The procedure *CCutSelect*, presented in Algorithm 6.4, describes the strategy for selecting the colorful cut which will be used as a pivot for the CCut branching. It consists in contracting all the labels of the graph which have been fixed to 1 (lines 3 and 4), removing all edges whose label was fixed to 0 (lines 5 and 6), and selecting the colorful cut with the minimum number of labels, considering only the singletons, i.e., the unitary sets of vertices (lines 7 to 9).

Algorithm 6.2: Colorful cuts branch and bound

```
1  procedure CCutBB($G = (V,E,L)$, LPModel, GlobalUB)
2      nodeStack ← emptyStack;
3      rootNode ← createNode( LPModel );
4      nodeStack.push( rootNode );
5      while nodeStack is not empty do
6          currenNode ← nodeStack.pop();
7          Z*, LB ← solveLR( currentNode );
8          GlobalUB ← MIN( GlobalUB, MVCA'(currentNode) );
9          GlobalUB ← MIN( GlobalUB, MVCA"(Z*) );
10         if currentNode is infeasible then
11             bound( currentNode );
12         else if GlobalUB - LB < 1 then
13             bound( currentNode );
14         else
15             newNodeStack ← CCutBranch( crrentNode ) ;
16             while newNodeStack is not empty do
17                 nodeStack.push( newNodeStack.pop() );
18     return GlobalUB ;
```

The Figure 6.4 illustrates a *colorful cuts branch-and-bound tree* on the z variables of CCut formulation. Observe that this new branching tree is wider than the traditional one since it could create more than two subproblems at each branching phase. On the other hand it is shorter and more balanced, given that each subproblem has a variable set to 1. Figure 6.6(b) depicts the structure of a wide and short colorful cuts branching tree.

In addition, observe that the structure of the colorful cuts branch-and-bound tree makes it very suitable for applying the domination rule (refer to Definition 3.10 and Proposition 3.5) and the monochromatic cuts removal (refer to Definition 3.7 and Proposition 3.3). Indeed, from

Algorithm 6.3: Colorful cuts branching strategy

```
procedure CCutBranch(currentNode)
    newNodeStack ← emptyStack;
    K ← CCutSelect();
    Sort K so that $|E(K_i)| \leq |E(K_{i+1})|$, for $i = 1$ to $size(K) - 1$;
    for k = 1 to size(K) do
        node$_k$ ← copy( currentNode );
        fix $z_k$ = 1 in node$_k$;
        for x = 1 to k-1 do
            fix $z_x$ = 0 in node$_k$;
        newNodeStack.push( node$_k$ );
    return newNodeStack ;
```

Algorithm 6.4: Strategy for quick selection of the pivot CCut

```
procedure CCutSelect($G = (V,E,L)$, currentNode)
    $H = (V',E',L') \leftarrow G = (V,E,L)$;
    foreach $z_l$ fixed to 1 in currentNode do
        $H \leftarrow H /\!/ l$;
    foreach $z_l$ fixed to 0 in currentNode do
        remove all the edges with label $l$ from $E'$;
    $K_{min} \leftarrow K(\{v\})$ for any $v \in V'$;
    foreach $v \in V'$ do
        if size(K({v} < size($K_{min}$))) then $K_{min} \leftarrow K(\{v\})$ ;
    return $K_{min}$ ;
```

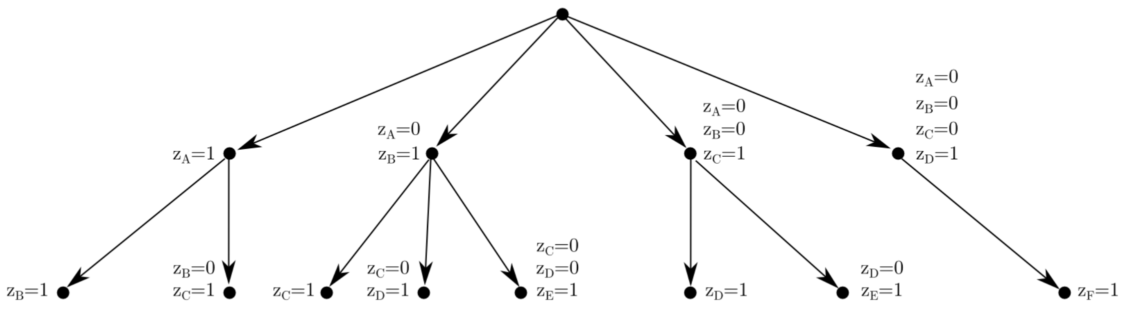


Figure 6.4: A colorful cuts branch-and-bound tree on the z variables of CCut formulation

Theorem 3.1 and Proposition 3.2, once a variable z_A is set to 1, it is possible to contract all the edges $e \in E$ with $l(e) = A$, remove the formed monochromatic cycles, and solve the problem on the resulting graph. This fact allied with the big number of variables set to 0 makes possible to find dominated labels as well as monochromatic cuts. Figure 6.5 illustrates the impact of the colorful cuts branching on an example ELG. Fig. 6.5(a) presents the ELG $G = (V,E,L)$

which has the colorful cut $K(\{8\}) = \{A, B\}$, used as a pivot for the colorful cuts branching. Fig. 6.5(b) shows the first subproblem, where z_A is set to 1, the label A is contracted, and its monochromatic cycles are removed. Observe that the label E became dominated by the label F. Fig. 6.5(c) shows the second subproblem, where z_A is set to 0, z_B is set to 1, the label B is contracted, and its monochromatic cycles are removed. Observe that the monochromatic cut $K(\{4\}) = \{F\}$ was created.

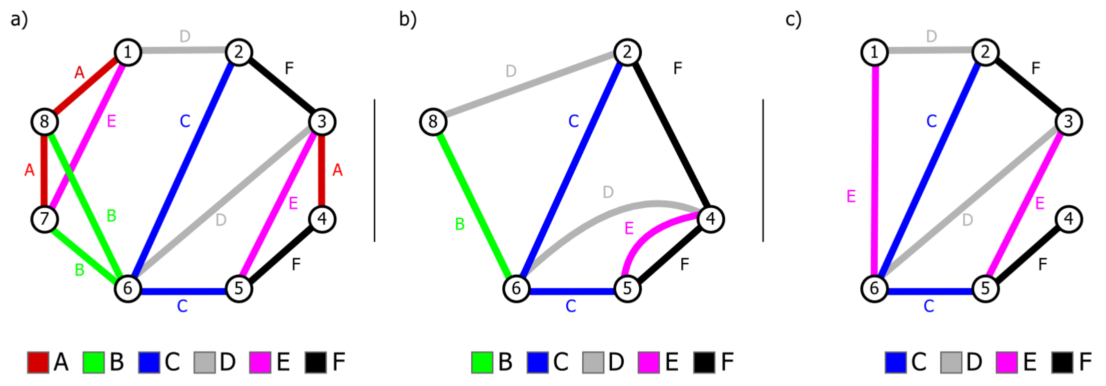


Figure 6.5: The impact of the colorful cuts branching on an example ELG. **(a)** The ELG $G = (V, E, L)$ and the colorful cut $K(\{8\}) = \{A, B\}$, used as pivot. **(b)** The subproblem generated by setting $z_A = 1$. **(c)** The subproblem generated by setting $z_A = 0$, and $z_B = 1$

We have performed an experiment to verify the quality of the CCut branching strategy in comparison to the branch-and-cut algorithm $BC^I(CCut^t, DFS)$, which uses the traditional binary branching strategy. This experiment is presented in Section 6.3. Unfortunately, as reported in Tables 6.5 and 6.6, we have observed that the CCut branching strategy presents serious convergence problems. Probably, it is due the fact that the number of nodes in the branch-and-bound tree grows too fast with its height.

Aiming to combine the balance provided by the colorful cuts branch-and-bound tree with the convergence power of the traditional one, we propose a new hybrid branching strategy, namely the *colorful cuts hybrid branch-and-bound* (CCutHB). Given an integer parameter $h_0 > 0$, it consists in using the colorful cuts branching in the first few h_0 levels of the tree[2] and then switch to the traditional binary branching. This strategy allows a good use of the contraction and dominance properties at the beginning of the tree whereas it delivers much smaller subproblems to the traditional branch and bound. Further, the traditional branch-and-bound phase of this method could be solved by calling directly the Cplex solver.

Figure 6.6 illustrates the limitations of the branching strategies described previously and presents the structure of the proposed hybrid branch-and-bound tree. Fig. 6.6(a) depicts an

[2]Consider the root node is at level 0

unbalanced traditional branch-and-bound tree. Fig. 6.6(b) shows a wide and short colorful cuts branch-and-bound tree. Fig. 6.6(c) presents the structure of the proposed hybrid branching strategy. The next section provides more detailed computational experiments in order to assess the quality of the new proposed strategy.

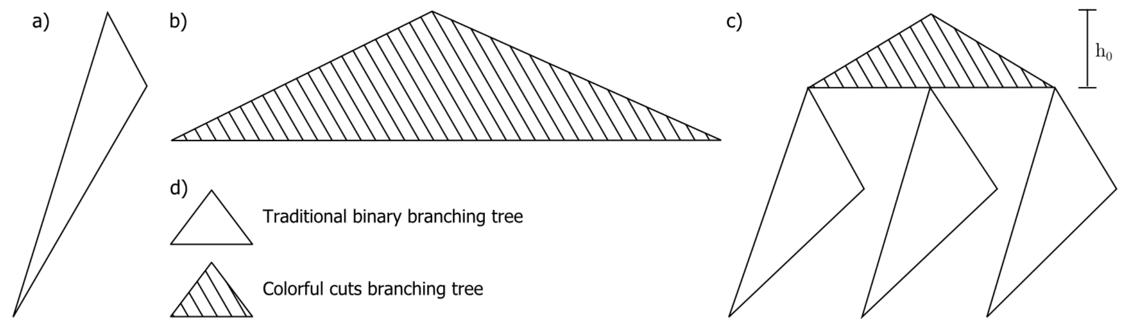

a)
b)
c)
h_0
d)
Traditional binary branching tree
Colorful cuts branching tree

Figure 6.6: The limitations of the branching strategies and the structure of the hybrid branch-and-bound tree. **(a)** An unbalanced traditional branch-and-bound tree. **(b)** A wide and short colorful cuts branch-and-bound tree. **(c)** The structure of the hybrid branching strategy. **(d)** The subtitles

6.3 Computational experiments

This section reports the computational experiments performed in order to evaluate the methods proposed in this chapter, namely the PCut formulation (specifically the PCut inequalities) and the branch-and-bound strategies CCutBB and CCutHB. First, we study the impact of adding the PCut inequalities to the branch-and-cut algorithms studied in Chapter 4. Next, we test the new proposed branch-and-bound strategies. Lastly, we combine these two approaches into a new branch-and-cut algorithm and verify its performance.

All experiments reported in this chapter were implemented in C++ language and compiled by using g++ 4.6.3 with the optimization flag -O3. The formulations and all of its derived procedures were implemented using the Concert library and Cplex 12.51 as the solver. The experiments were performed on a computer with Intel(R) Core(TM) i7-4790K CPU, 3.4GHz, 16 GB of RAM, and Ubuntu 14.04 as the operating system. Although the processor of this device has more than one core, the algorithms were executed using a single core and a single thread within a time limit of 2 hours. Further, we turned off all presolve features and all automatic cutting-plane generation procedures while all other parameters of the Cplex were set to their respective default values. Moreover, it is possible to observe from the experiments reported in Chapter 4 that the branch-and-cut algorithm $BC^I(CCut^t, DFS)$, CCut^t for short, has obtained very consistent results with respect both to running times and linear relaxation quality. For this reason, we use this method as a reference to assess the quality of the improvements proposed in this chapter.

The first experiment performed studies the impact of separating heuristically the PCut inequalities in the root node of the branch-and-cut algorithm $BC^I(CCut^t, DFS)$. For this experiment, we have considered the graphs with number of vertices $n = |V| \in \{100, 200\}$ from the group 2 of ELGs generated by Cerulli et al. (2005) (refer to Section 4.3 for more details). However, due to the size of the input graphs, we had to modify slightly the heuristic PCut^h, presented in Algorithm 6.1. In the lines 6 and 8, instead of evaluating all the sets $\mathcal{NS}(s_i)$ and $\mathcal{NM}(s_i)$, we evaluate just n solutions taken randomly from each set. Further, we have changed the limits for the generation of the initial solution (line 2) to $[n-60, n-10]$. We call this modified version of PCut^h as PCut^{rh}.

The results of this experiment are reported in Tables 6.3 and 6.4. Each line of these tables represents a dataset, which is a set with 10 instances with the same *n-l-d* configuration. The first two columns identify the input instances. The next column reports the average optimal solution for the dataset. It was obtained from the results reported in Chapter 4[3], and the remaining columns show the computational results of the methods considered. The next six columns refer to the results of CCut^t, while the last six refer to the results of adding the separation heuristic PCut^{rh} on the root node of $BC^I(CCut^t, DFS)$. The column '$\star$' reports the number of instances the method has failed in solving to optimality within each dataset. The columns *UB*, *gap* and *gapr* report, respectively, the average upper bound, the average gap ($(UB-LB)/UB$), and the average gap on the root node of the model ($(UB-LR)/UB$), considering each instance within each dataset. The column *cuts* refers to the total number of cuts added to the model. Lastly, the column *t(s)* reports the total time (in seconds) necessary to solve all instances in each dataset.

The results for instances with $n = 100$ show that separating the PCut inequalities improves slightly the linear relaxation of the model. However, it could not obtain better results than CCut^t. Despite both methods were able to solve all instances of this group to optimality, seems that adding the PCut inequalities to the model makes it harder to solve. Indded, CCut^t has achieved the best time for 11 out of 12 datasets. For instances with $n = 200$, PCut^{rh} still improves the linear relaxation of CCut^t. Besides, for this group, adding the PCut inequalities has shortened the time needed to solve 7 datasets out of 12. Observe that the performance of the new method gets better as the number of label increases and the edge density get smaller. Indeed, for instances with $d = 0.2$ and $l = |L| = 200$, the method CCut^t + PCut^{rh} was able to obtain a better upper bound as well as to solve more instances to the optimality.

[3]The methods reported in Chapter 4 were not able to find all the optimal solutions for the instances with $|V| = 200$. The missing values were taken from (Silva et al., 2015)

Table 6.3: Computational results for CCut^t + PCut^{rh} on instances with $|V| = 100$

					CCut^t						**CCut^t + PCut^{rh}**			
d	**l**	**Opt**	$\star$	**UB**	**t(s)**	**cuts**	**gap**	**gapr**	$\star$	**UB**	**t(s)**	**cuts**	**gap**	**gapr**
0.8	25	1.8	0	1.8	0.033	0	0%	36%	0	1.8	**0.014**	0	0%	36%
	50	2.0	0	2.0	**0.089**	1	0%	35%	0	2.0	14.210	8	0%	35%
	100	3.0	0	3.0	**1.997**	24	0%	33%	0	3.0	67.645	90	0%	33%
	125	4.0	0	4.0	**37.376**	4	0%	50%	0	4.0	78.578	107	0%	**40%**
0.5	25	2.0	0	2.0	**0.059**	1	0%	35%	0	2.0	2.508	0	0%	35%
	50	3.0	0	3.0	**0.556**	5	0%	39%	0	3.0	60.531	99	0%	39%
	100	4.7	0	4.7	**18.230**	47	0%	34%	0	4.7	41.760	181	0%	**33%**
	125	5.2	0	5.2	**9.079**	49	0%	29%	0	5.2	51.809	264	0%	29%
0.2	25	4.5	0	4.5	**0.173**	5	0%	36%	0	4.5	8.622	20	0%	36%
	50	6.7	0	6.7	**1.274**	12	0%	30%	0	6.7	8.241	80	0%	**29%**
	100	9.7	0	9.7	**16.576**	214	0%	22%	0	9.7	21.129	428	0%	**20%**
	125	11.0	0	11.0	**21.259**	566	0%	20%	0	11.0	32.954	648	0%	**19%**

Table 6.4: Computational results for CCut^t + PCut^{rh} on instances with $|V| = 200$

					CCut^t						**CCut^t + PCut^{rh}**			
d	**l**	**Opt**	$\star$	**UB**	**t(s)**	**cuts**	**gap**	**gapr**	$\star$	**UB**	**t(s)**	**cuts**	**gap**	**gapr**
0.8	50	2.0	0	2.0	0.122	0	0%	46%	0	2.0	**0.064**	0	0%	46%
	100	2.6	0	2.6	**10.7**	7	0%	48%	0	2.6	9863.0	534	0%	48%
	200	4.0	0	4.0	975.0	3	0%	48%	0	4.0	**732.8**	68	0%	48%
	250	4.0	0	4.0	**435.0**	22	0%	48%	0	4.0	839.1	110	0%	**37%**
0.5	50	2.2	0	2.2	**0.5**	3	0%	39%	0	2.2	27.8	3	0%	**37%**
	100	3.4	0	3.4	**17.9**	8	0%	44%	0	3.4	421.6	113	0%	**43%**
	200	5.4	0	5.4	2377.5	63	0%	39%	0	5.4	**1774.8**	187	0%	39%
	250	6.3	0	6.3	12280.5	22	0%	37%	0	6.3	**12115.0**	156	0%	37%
0.2	50	5.2	0	5.2	**7.7**	5	0%	44%	0	5.2	46.0	23	0%	44%
	100	7.9	0	7.9	529.1	18	0%	39%	0	7.9	**399.4**	83	0%	39%
	200	11.9	3	12.1	29671.5	238	4%	33%	**2**	**11.9**	**26719.2**	480	**2%**	**31%**
	250	13.7	5	13.8	49853.0	298	**6%**	30%	5	13.8	**48435.0**	424	7%	30%

The second experiment aims to verify the quality of the branching strategy CCutBB (introduced in Section 6.2), by comparing it with the branch-and-cut algorithm $BC^I(CCut^t, DFS)$, for short CCut^t, which uses the traditional binary branching strategy. Again, we have carried out both methods for each instance from the group 2 of ELGs generated by Cerulli et al. (2005) with number of vertices $n = |V| \in \{100, 200\}$. The results of this experiment are reported in Tables 6.5 and 6.6. These tables follow the same structure of tables 6.3 and 6.4, except that we have removed the columns *cuts* and *gapr* and added the column *nodes*, which stands for the total number of nodes solved in the branch-and-bound tree.

Table 6.5: Computational results for CCutBB on instances with $|V| = 100$

					CCut^t					CCutBB		
d	**l**	**Opt**	$\star$	**UB**	**t(s)**	**nodes**	**gap**	$\star$	**UB**	**t(s)**	**nodes**	**gap**
0.8	25	1.8	0	1.8	0.033	**0**	0%	0	1.8	**0.007**	**0**	0%
	50	2.0	0	2.0	0.089	**0**	0%	0	2.0	**0.022**	**0**	0%
	100	3.0	0	3.0	**1.997**	12191	0%	0	3.0	2.613	**1157**	0%
	125	4.0	0	4.0	**37.376**	167241	0%	0	4.0	47.693	**19465**	0%
0.5	25	2.0	0	2.0	0.059	**10**	0%	0	2.0	**0.016**	14	0%
	50	3.0	0	3.0	0.556	1437	0%	0	3.0	**0.528**	**493**	0%
	100	4.7	0	4.7	18.230	148403	0%	0	4.7	**16.368**	**9591**	0%
	125	5.2	0	5.2	**9.079**	61949	0%	0	5.2	34.515	**16099**	0%
0.2	25	4.5	0	4.5	0.173	391	0%	0	4.5	**0.095**	**203**	0%
	50	6.7	0	6.7	**1.274**	9037	0%	0	6.7	1.820	**2428**	0%
	100	9.7	0	9.7	**16.576**	109809	0%	0	9.7	53.747	**29315**	0%
	125	11.0	0	11.0	**21.259**	109374	0%	0	11.0	531.093	**71576**	0%

Table 6.6: Computational results for CCutBB on instances with $|V| = 200$

					CCut^t					CCutBB		
d	**l**	**Opt**	$\star$	**UB**	**t(s)**	**nodes**	**gap**	$\star$	**UB**	**t(s)**	**nodes**	**gap**
0.8	50	2.0	0	2.0	0.122	**0**	0%	0	2.0	**0.049**	**0**	0%
	100	2.6	0	2.6	10.7	28313	0%	0	2.6	**5.7**	**621**	0%
	200	4.0	0	4.0	975.0	3333282	0%	0	4.0	**718.6**	**67729**	0%
	250	4.0	0	4.0	**435.0**	601875	0%	0	4.0	1548.5	**117481**	0%
0.5	50	2.2	0	2.2	0.5	470	0%	0	2.2	**0.3**	**85**	0%
	100	3.4	0	3.4	**17.9**	72969	0%	0	3.4	19.8	**3832**	0%
	200	5.4	0	5.4	**2377.5**	7077886	0%	0	5.4	4408.5	**526126**	0%
	250	6.3	0	6.3	**12280.5**	34418801	0%	0	6.3	17639.8	**1439063**	0%
0.2	50	5.2	0	5.2	**7.7**	35236	0%	0	5.2	11.6	**7245**	0%
	100	7.9	0	7.9	**529.1**	2032335	0%	0	7.9	960.3	**343314**	0%
	200	11.9	**3**	**12.1**	**29671.5**	62381046	**4%**	8	12.4	61915.0	**8881646**	25%
	250	13.7	**5**	**13.8**	**49853.0**	92983303	**6%**	10	15.0	72000	**8439352**	34%

Considering the set of instances with 100 vertices, it is possible to observe that CCutBB is very competitive for graphs from medium to high edge densities. Indeed, it has solved all instances to optimality as well as obtained the best times for 5 out of 8 datasets. Notwithstanding, its performance decreases considerably for low density graphs. Specially, CCutBB has used more than 500s for solving all 10 instances of the dataset 100-125-0.2, while CCut^t has used just 21s. Further, observe that the number of nodes solved by the branch and bound tree of the CCutBB procedure is smaller for 11 out of 12 datasets. As expected, it is due to the fact that the CCutBB tree is more balanced than the CCut^t one.

For the set of instances with 200 vertices, despite the fact CCutBB still visits less nodes, it did not performed well. It was able to solve the problems faster for high density input graphs ($d = 0.8$) with $l \leq 200$ and for the dataset 200-50-0.5. However, for the remaining datasets, its running times are too big in comparison to the CCut^t ones. Furthermore, CCutBB was not able to solve any instance from the dataset 200-250-0.2 as well as managed to solve only two instances from the dataset 200-200-0.2. Along with the results for instances with 100 vertices, it suggests CCutBB presents serious convergence problems for ELGs with low edge density ($l = 0.2$) and high number of labels ($l \geq n$).

Lastly, the next experiment aims to evaluate three versions of the method CCutHB (refer to Section 6.2), namely CCutHB$(h_0 = 1)$, CCutHB$(h_0 = 2)$, and CCutHB$(h_0 = 2)$ + PCut^{rh} (for short PCutHB$(h_0 = 2)$). The latter stands for using the heuristic PCut^{rh} for separating the PCut inequalities at the root node of the method CCutHB$(h_0 = 2)$, which has performed better than CCutHB$(h_0 = 1)$. We have executed the methods for each instance from the group 2 of ELGs generated by Cerulli et al. (2005) with number of vertices $n = |V| = 200$ and the results of this experiment are reported in Table 6.7. Each line of this table reports the results obtained by one method. The lines are grouped four by four and each group represents a dataset, which is a set with ten input ELGs with the same *n-l-d* configuration. The first column identifies the algorithm, while the next two identify the dataset. The columns '$\star$', *UB*, *t(s)*, *cuts*, *gap*, and *gapr* have the same meaning as in Table 6.3. The column *lrt(s)* reports the total time (in seconds) necessary to solve the root node of all instances in a dataset. Finally, the columns *nodesT* and *nodesC* stand, respectively, for the total number of nodes solved in the traditional branch-and-bound tree phase and in the CCut branching one.

From the results of this experiment it is possible to observe that all approaches derived from CCutHB have outperformed CCut^t with respect to the number of instances solved to optimality. It is important to highlight that both CCutHB$(h_0 = 2)$ and PCutHB$(h_0 = 2)$ have only failed to solve 2 out of 120 instances to optimality. It is the best known result so far. Further, these methods have also achieved the best UB values for all datasets, with the exception of CCutHB$(h_0 = 1)$ for the dataset 200-250-0.2.

Considering the graphs with edge density $d \in \{0.5, 0.8\}$, the method CCutHB$(h_0 = 1)$ has achieved the best results. Indeed, although all approaches have solved the four datasets to optimality, this method was able to complete the tasks in much smaller times, specially for instances with $l \geq 200$. For this group of instances, it seems that separating the PCut inequalities only increases the completion time of the method. On the other hand, CCutHB$(h_0 = 2)$ has achieved the best results for input ELGs with edge density $d = 0.2$. In fact, it has obtained the best times for 3 out of 4 groups and, along with PCutHB$(h_0 = 2)$, it solved more instances to optimality than the other methods. Regarding separating heuristically the PCut inequalities on the root node of PCutHB$(h_0 = 2)$, the results suggest that it becomes relevant as the density of the input graphs decreases and its number of labels increases. Particularly for the dataset 200-250-0.2, the hardest one in this set of experiments, this method was able to improve the running time of CCutHB$(h_0 = 2)$ and to reduce the total number of nodes in the branch-and-bound tree by a factor of 3.57.

6.4 Concluding remarks

In this chapter we have introduced two improvements for the CCut formulation. The first one is based on a new mathematical model, namely PCut, that extends CCut by using the concept of *partitioning cuts*. The second is a new hybrid branching strategy for solving the CCut model which is based on the concept of colorful cuts. In addition, we have proposed branch-and-cut algorithms and separation procedures, as well as performed computational experiments to evaluate the new proposed methods. The improvements proposed lead to four new branch-and cut algorithms, namely CCutBB, CCutHB$(h_0 = 1)$, CCutHB$(h_0 = 2)$, and PCutHB$(h_0 = 2)$. We can observe from the computational experiments carried out that the new approaches were able to overperform CCut, achieving the best results regarding exact methods for both the MLSTP and the GMLSTP so far.

Regarding to PCut model and to the partitioning cuts inequalities, the results have showed that they were able to improve slightly the results of the CCut formulation. Moreover, separating these cuts on the root node of the CCutHB$(h_0 = 2)$ has showed to worth for instances with low density and a large number of labels. In its turn, CCutBB, the branch-and-bound algorithm that uses only the CCut branching strategy did not performed well. As evidenced by the experiments, it has presented serious convergence problems. On the other hand, the hybrid branching strategies proposed have successfully combined the balance of the new branching strategy with the convergence power of the traditional one. It is relevant to evidence that the resulting methods CCutHB$(h_0 = 1)$ and CCutHB$(h_0 = 2)$ have improved the performance of CCut formulation, being able to solve to optimality 238 out of 240 input graphs.

We believe the proposed methods can be further improved with the development of new heuristics for separating the PCut inequalities. Other improvements would be to provide a smart way to select which of the methods would be used based on the characteristics of the input ELG, or to change dynamically the parameter h_0 accordingly to the node to be solved.

Table 6.7: Computational results for CCutHB on ELGs with |V|=200

	d	**l**	$\star$	**UB**	**t(s)**	**lrt(s)**	**nodesT**	**nodesC**	**cuts**	**gap**	**gapr**
CCut^t	**0.8**	**50**	0	2.0	0.122	0.051	0	0	0	0%	46%
CCutHB($h_0 = 1$)			0	2.0	**0.049**	0.042	0	0	0	0%	46%
CCutHB($h_0 = 2$)			0	2.0	**0.049**	0.042	0	0	0	0%	46%
PCutHB($h_0 = 2$)			0	2.0	0.052	0.040	0	0	0	0%	46%
CCut^t	**0.8**	**100**	0	2.6	10.713	0.112	28313	0	7	0%	48%
CCutHB($h_0 = 1$)			0	2.6	5.931	0.096	0	621	0	0%	48%
CCutHB($h_0 = 2$)			0	2.6	**5.703**	0.096	9	621	9	0%	48%
PCutHB($h_0 = 2$)			0	2.6	10280.1	10270.7	9	621	540	0%	48%
CCut^t	**0.8**	**200**	0	4.0	975.1	0.024	3333282	0	3	0%	48%
CCutHB($h_0 = 1$)			0	4.0	**11.5**	0.030	0	938	0	0%	48%
CCutHB($h_0 = 2$)			0	4.0	1012.8	0.031	0	67729	0	0%	48%
PCutHB($h_0 = 2$)			0	4.0	1566.9	515.0	0	67729	65	0%	48%
CCut^t	**0.8**	**250**	0	4.0	435.0	0.246	601875	0	22	0%	37%
CCutHB($h_0 = 1$)			0	4.0	**64.9**	0.033	1011	994	14	0%	37%
CCutHB($h_0 = 2$)			0	4.0	1750.8	0.033	0	77522	0	0%	37%
PCutHB($h_0 = 2$)			0	4.0	2319.5	489.4	0	77522	87	0%	37%
CCut^t	**0.5**	**50**	0	2.2	0.5	0.072	470	0	3	0%	39%
CCutHB($h_0 = 1$)			0	2.2	**0.2**	0.046	0	85	0	0%	39%
CCutHB($h_0 = 2$)			0	2.2	0.3	0.046	0	85	0	0%	39%
PCutHB($h_0 = 2$)			0	2.2	28.4	28.1	0	85	0	0%	**37%**
CCut^t	**0.5**	**100**	0	3.4	17.9	0.055	72969	0	8	0%	44%
CCutHB($h_0 = 1$)			0	3.4	**3.8**	0.038	0	454	0	0%	44%
CCutHB($h_0 = 2$)			0	3.4	26.4	0.038	0	4920	0	0%	44%
PCutHB($h_0 = 2$)			0	3.4	473.7	445.0	0	4920	112	0%	44%
CCut^t	**0.5**	**200**	0	5.4	2377.5	0.033	7077886	0	63	0%	39%
CCutHB($h_0 = 1$)			0	5.4	**609.9**	0.039	16093	541	23	0%	39%
CCutHB($h_0 = 2$)			0	5.4	781.0	0.039	0	26450	0	0%	39%
PCutHB($h_0 = 2$)			0	5.4	1032.5	213.1	0	26450	157	0%	39%
CCut^t	**0.5**	**250**	0	6.3	12280.5	0.034	34418801	0	22	0%	37%
CCutHB($h_0 = 1$)			0	6.3	**925.2**	0.037	89811	565	16	0%	37%
CCutHB($h_0 = 2$)			0	6.3	724.9	0.037	385	29716	9	0%	37%
PCutHB($h_0 = 2$)			0	6.3	937.3	207.8	332	29716	156	0%	37%
CCut^t	**0.2**	**50**	0	5.2	7.7	0.045	35236	0	5	0%	44%
CCutHB($h_0 = 1$)			0	5.2	9.3	0.025	3	132	0	0%	44%
CCutHB($h_0 = 2$)			0	5.2	**3.6**	0.026	0	1281	0	0%	44%
PCutHB($h_0 = 2$)			0	5.2	40.5	36.9	0	1281	19	0%	44%
CCut^t	**0.2**	**100**	0	7.9	529.1	0.047	2032335	0	18	0%	39%
CCutHB($h_0 = 1$)			0	7.9	310.5	0.039	212667	150	12	0%	39%
CCutHB($h_0 = 2$)			0	7.9	**261.2**	0.039	1077	2306	2	0%	39%
PCutHB($h_0 = 2$)			0	7.9	313.3	36.1	1130	2306	77	0%	39%
CCut^t	**0.2**	**200**	3	12.1	29671.5	0.062	62381046	0	238	4%	33%
CCutHB($h_0 = 1$)			**0**	**11.9**	18507.2	0.051	8657165	159	172	**0%**	**31%**
CCutHB($h_0 = 2$)			**0**	**11.9**	**13435.0**	0.051	241539	2704	47	**0%**	**31%**
PCutHB($h_0 = 2$)			**0**	**11.9**	16650.3	117.4	42631	2704	450	**0%**	**31%**
CCut^t	**0.2**	**250**	5	**13.8**	49853.0	0.055	92983303	0	298	**6%**	**30%**
CCutHB($h_0 = 1$)			4	14.1	44858.2	0.053	11616250	161	299	13%	31%
CCutHB($h_0 = 2$)			**2**	**13.8**	27907.1	0.053	3182721	2478	120	**6%**	**30%**
PCutHB($h_0 = 2$)			**2**	**13.8**	**25679.9**	72.2	891156	2463	342	**6%**	**30%**

Chapter 7

Heuristic Methods

In this chapter we propose a new heuristic approach for both MLSTP and GMLSTP. First, we present a revised version of the maximum vertex covering algorithm, the most successful constructive heuristic for these problems, and provide a tighter bound to its time complexity. Further, a new MIP-based metaheuristic is proposed for solving the GMLSTP, the multi-start local branching (MSLB). It combines the efficiency of the proposed constructive heuristics with the capacity of exploration of a new local search method based on MIP. The computational experiments performed show that the MSLB is superior to the current state-of-the-art metaheuristics in respect to quality of the solutions and processing times.

The remainder of this chapter is structured as follows: Section 7.1 introduces a modified version of the maximum vertex covering algorithm and provides a tighter bound on its time complexity. Section 7.2 presents the multi-start local branching, a new MIP-based metaheuristic for solving the MLSTP. Section 7.3 discuss an experimental analysis of the proposed method in comparison of the state-of-the-art metaheuristics for the problem.

7.1 Revised MVCA

As discussed in Section 2.2, one can say the MVCA is the most successful constructive heuristic for both MLSTP and GMLSTP. It has been used to provide initial solutions or to complete partial ones by many metaheuristic-based methods. Moreover, even other constructive heuristics rely on MVCA to build or to rebuild solutions.

This section proposes a revised version of MVCA, namely rMVCA, with the aim of providing a best upper bound on its running time complexity. This new version is based on Theorem 3.1 and on the fact that $G \mathbin{/\!\!/} L'$ has exactly the same number of vertices as the number of connected components of $G[L']$, for $L' \in L$, as discussed in Section 3.1. The main idea is to carry out fewer operations to determine the next label to be added to the solution. To do so, at each iteration, the method keeps a precomputed version of the graph for each possible choice of label, as well as update these auxiliary data structures dynamically.

Assume that G is a monochromatic-cycle-free ELG. As discussed in Section 3.2, the monochromatic cycles can be broken by using Algorithm 3.1. The rMVCA is presented in Algorithm 7.1: in the line 2, C is defined to carry the set of labels of the solution; the loop of line 3 computes the graph $G_l = G /\!/ \{l\}$, $\forall l \in L$, that stands for the resulting graph if the label l is added to the current solution; the loop of lines 4-14 represents one iteration of the method, while its condition assures the connectivity of the final graph; the greedy choice of the rMVCA is performed in the line 5; the loop of lines 7-10 updates the input graph to reflect the label chosen to enter the solution; in the line 11 the solution is updated; and in the loop of lines 12-14 each graph G_l, $l \in L \backslash C$, is updated such that $G_l = G /\!/ C \cup \{l\}$. Recall, from Definition 3.5, that $\xi(e, E_l)$ stands for the projection of the edge e on the set of edges of the graph G_l.

Algorithm 7.1: Revised MVCA

```
1  procedure rMVCA($G = (V,E,L)$)
2      Let $C \leftarrow \emptyset$ be the set of labels of the solution;
3      foreach $l \in L$ do $G_l \leftarrow G /\!/ \{l\} = (V_l, E_l, L)$ ;
4      while $|V| > 1$ do
5          $c \leftarrow argmin_{l \in L \backslash C}(|V_l|)$;
6          $E^* \leftarrow \emptyset$ ;
7          foreach $e = (v1, v2) \in E(\{c\})$ do
8              if $v1 \neq v2$ then
9                  $E^* \leftarrow E^* \cup \{e\}$ ;
10                 $G \leftarrow G/e$ ;
11         $C \leftarrow C \cup \{c\}$;
12         foreach $l \in L \backslash C$ do
13             foreach $e \in E^*$ do
14                 $G_l \leftarrow G_l/\xi(e, E_l)$ ;
15     return $C$;
```

It is easy to see that the rMVCA follows exactly the definition of the MVCA: it starts with the solution $C = \emptyset$, and, while the number of vertices of $G /\!/ C$ is greater than 1, iteratively adds to C the label that minimizes the number of vertices of $G /\!/ C$. At each iteration of rMVCA, $G /\!/ C$ is the current solution graph, G_l is $G /\!/ C \cup \{l\}$, $\forall l \in L \backslash C$, and the label with minimum $|V_l|$ is the one to be added to the solution. Once the label c is chosen to enter the solution, it is necessary to update the current solution graph to $G /\!/ C \cup \{c\}$ and each graph G_l to $G /\!/ C \cup \{l\} \cup \{c\}$. Remark that if an edge is not able to contract two vertices of $G /\!/ C$, it cannot contract two vertices of $G \mathbin{/\!\!/} C \cup \{l\}$. Thence, when updating each graph G_l, it is only necessary to consider the edges that were able to contract vertices of $G \mathbin{/\!\!/} C$.

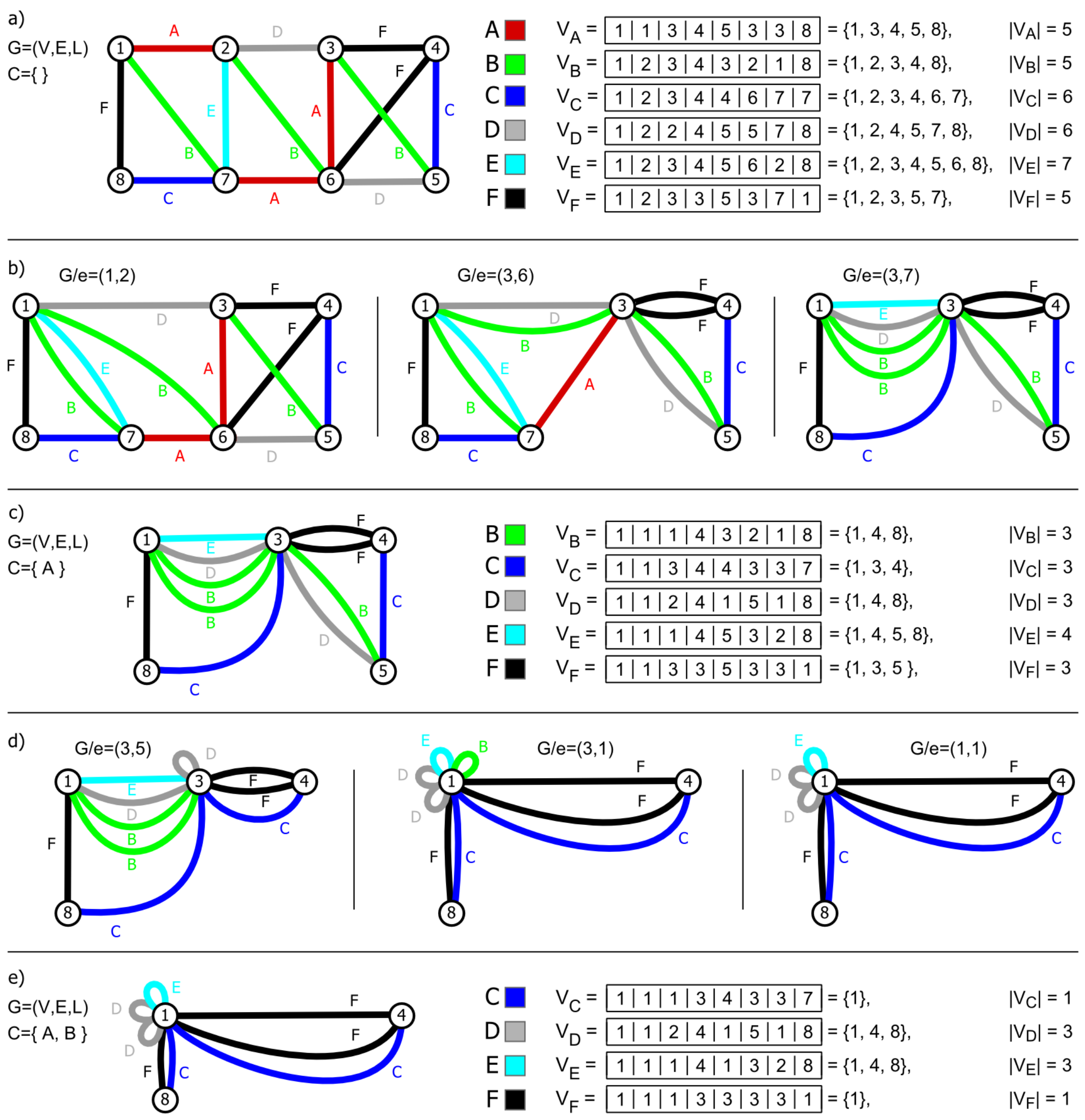
Theorem 7.1. *The time complexity of the rMVCA is $O(\alpha_n kn)$, where $n = |V|$, $k = |L|$, $\alpha_n = \alpha(n,n)$, and α is the inverse of the Ackerman's function.*

Proof. *First, rMVCA assumes that the input graph does not have monochromatic cycles. Let $G = (V,E,L)$ be the input graph. This implementation represents the set of vertices of the graphs G and G_l, $\forall l \in L$, by using data structures for disjoint sets (Cormen et al., 2009). Since $O(n)$ union-find operations are necessary to compute each graph G_l, the loop of the line 3 takes $O(\alpha_n kn)$ time.*

Let p be the number of iterations of the main loop. We have that $p < n$, because at most $n-1$ labels are added to the final solution, as well as $p < k$, because all the labels are added to the solution in the worst case. Remark that the number of vertices of each graph is given by n minus the number of successful union operations performed on that graph, what can be computed while contracting it. Then, the greedy choice of the line 5 is performed in $O(pk) \subseteq O(kn)$ total time. Also, the condition of the loop in the line 4 and the operations of lines 6 and 11 can be carried out in constant time, resulting in $O(p) \subseteq O(n)$ total time.

The update of the current solution graph, lines 7-10, requires $O(n)$ union-find operations per iteration, in a total time of $O(\alpha_n pn) \subseteq O(\alpha_n kn)$. Finally, let $q_i = |E^*|$ be the cardinality of E^* at the ith iteration. Thus, the total number of union-find operations necessary to update each graph G_l, $l \in L \backslash C$ (lines 12-14) is $q = \sum_{i=1}^{p} q_i$. Since E^* stands for the set of edges that actually contracted two vertices of G, and only $n-1$ edges do contract vertices of G during the complete execution of the rMVCA, it follows that $q = n-1$. Thus, as the method keeps k graphs updated, this step takes an overall time of $O(\alpha(q,n) \cdot kq) \subseteq O(\alpha_n kn)$. As a consequence, the time complexity of the rMVCA is $O((\alpha_n kn) + (kn) + (n) + (\alpha_n kn) + (\alpha_n kn)) \subseteq O(\alpha_n kn)$. $\square$

Figure 7.1 illustrates an execution of the rMVCA by using data structures for disjoint sets to represent the vertices of the graphs G_l, $\forall l \in L$. For the sake of clarity, the union-find operations do not use path compression and union by rank, as well as the unions are done from the larger ID to the smaller one. The Fig.7.1a brings the input graph, the set of vertices of the graphs G_l, and their cardinalities. The Fig.7.1b presents the update of the current solution graph after the choice of the label A. Fig.7.1c shows the graph at the beginning of the second iteration, as well as the set of vertices of G_l, $\forall l \in L \backslash C$, updated at the end of the previous iteration by using the set $E^* = \{(1,2),(3,6),(3,7)\}$. In Fig.7.1d it is showed the update of the current solution graph after choosing the label B. Note that the edge $e = (1,1) \in E(\{B\})$ was



A 🟥	V_A =	1 | 1 | 3 | 4 | 5 | 3 | 3 | 8	= {1, 3, 4, 5, 8},	$\|V_A\|$ = 5
B 🟩	V_B =	1 | 2 | 3 | 4 | 3 | 2 | 1 | 8	= {1, 2, 3, 4, 8},	$\|V_B\|$ = 5
C 🟦	V_C =	1 | 2 | 3 | 4 | 4 | 6 | 7 | 7	= {1, 2, 3, 4, 6, 7},	$\|V_C\|$ = 6
D ⬜	V_D =	1 | 2 | 2 | 4 | 5 | 5 | 7 | 8	= {1, 2, 4, 5, 7, 8},	$\|V_D\|$ = 6
E 🟦	V_E =	1 | 2 | 3 | 4 | 5 | 6 | 2 | 8	= {1, 2, 3, 4, 5, 6, 8},	$\|V_E\|$ = 7
F ⬛	V_F =	1 | 2 | 3 | 3 | 5 | 3 | 7 | 1	= {1, 2, 3, 5, 7},	$\|V_F\|$ = 5

B 🟩	V_B =	1 | 1 | 1 | 4 | 3 | 2 | 1 | 8	= {1, 4, 8},	$\|V_B\|$ = 3
C 🟦	V_C =	1 | 1 | 3 | 4 | 4 | 3 | 3 | 7	= {1, 3, 4},	$\|V_C\|$ = 3
D ⬜	V_D =	1 | 1 | 2 | 4 | 1 | 5 | 1 | 8	= {1, 4, 8},	$\|V_D\|$ = 3
E 🟦	V_E =	1 | 1 | 1 | 4 | 5 | 3 | 2 | 8	= {1, 4, 5, 8},	$\|V_E\|$ = 4
F ⬛	V_F =	1 | 1 | 3 | 3 | 5 | 3 | 3 | 1	= {1, 3, 5 },	$\|V_F\|$ = 3

C 🟦	V_C =	1 | 1 | 1 | 3 | 4 | 3 | 3 | 7	= {1},	$\|V_C\|$ = 1
D ⬜	V_D =	1 | 1 | 2 | 4 | 1 | 5 | 1 | 8	= {1, 4, 8},	$\|V_D\|$ = 3
E 🟦	V_E =	1 | 1 | 1 | 4 | 1 | 3 | 2 | 8	= {1, 4, 8},	$\|V_E\|$ = 3
F ⬛	V_F =	1 | 1 | 1 | 3 | 3 | 3 | 3 | 1	= {1},	$\|V_F\|$ = 1

Figure 7.1: Execution of rMVCA. **(a)** The input graph and the set of vertices of G_l, $\forall l \in L$. **(b)** $G \leftarrow G /\!\!/ \{A\}$. **(c)** The solution graph and the set of vertices of G_l, $\forall l \in L \backslash \{A\}$, at the 2nd iteration. **(d)** $G \leftarrow G /\!\!/ \{B\}$. **(e)** The solution graph and the set of vertices of G_l, $\forall l \in L \backslash \{A, B\}$, at the 3rd iteration

not capable to contract its endpoints on G, and thence it is not necessary to consider e while updating the graphs G_l. For this iteration $E^* = \{(3,5),(3,1)\}$. Lastly, from Fig.7.1e, that brings the graphs G and G_l, $\forall l \in L \backslash C$, at the beginning of the third iteration of the method, we have that choosing the labels C or F connects the solution.

7.2 Multi-start local branching

In this section, we introduce the multi-start local branching procedure (MSLB), a new hybrid metaheuristic for the GMLSTP. The MSLB incorporates two new constructive heuristics: a parametrized version of MVCA (pMVCA) and a MIP-based procedure denominated round and contract (R&C), and a local search family of heuristics based on a local branching approach (LB). The remaining of this section describes the methods pMVCA, R&C, and LB, as well as the metaheuristic MSLB.

7.2.1 Parametrized MVCA

The first few choices made by greedy heuristics, such as the rMVCA, play an important role in the quality of the solution they yield. This is due to the fact that, initially, these methods do not have enough information to make good decisions, while these bad decisions could lead the solution towards a local minimum. In fact, an experiment performed by Cerrone et al. (2017) on 10.000 randomly generated ELGs has demonstrated that from the first 25% selections of MVCA, less than 50% of the labels were part of the optimal solution.

This section proposes a parametrized version of rMVCA, denominated pMVCA, in order to minimize the impact of the initial choices of the method while keeping the greediness of the last selections. To this end, at each iteration, the label to enter into the solution is chosen randomly from the *Restricted Candidate List* (RCL) in the same way as in GRASP (Resende, 2009), where the RCL is a list with the $|RCL|$ best label selections for that iteration. Given $\delta, \theta \in (0,1]$, two parameters of pMVCA, the size of the RCL at iteration $i \in \{0,1,\cdots\}$ is given by $\max(1, \lfloor |L| \cdot \delta \cdot \theta^i \rfloor)$. Note that δ defines the initial size of the RCL, while θ controls the gradual reduction of its size through the iterations, what increases the greediness of the algorithm.

Some characteristics of pMVCA make it very suitable for multi-start procedures: it is based on an efficient constructive heuristic, the rMVCA; the randomness of the first choices allows the method to generate diversified solutions, as well as different configurations of δ and θ allow it to explore varied regions of the solution space; and the greediness of its last choices does not allow the method to generate very bad solutions.

7.2.2 Round & Contract heuristic

As discussed previously, the majority of the successful heuristic algorithms proposed for the MLSTP rely on MVCA. In this section we present the *Round & Contract* (R&C) procedure, a MIP-based constructive heuristic proposed as an alternative to this fact.

Let $Z^* \leftarrow \text{CCut}^*(G)$ represent the solution of the linear relaxation (LR) of the CCut formulation for the graph G, such that Z^*_l is the value of the variable associated with the label l in Z^*. The R&C starts with the solution $C = \emptyset$, and, while $G \mathbin{/\!/} C$ is not connected, at each iteration, adds to C the label $c \in L \backslash C$ with the greater Z^*_c. Given that CCut has an exponential number of constraints, it is executed with the singleton inequalities (4.5) in the place of the set of colorful cuts ones (4.2). Despite the fact that the constraints 4.2 could be dynamically separated, for the sake of performance, it is not carried out in the R&C method.

The R&C method is described in Algorithm 7.2. Each iteration of the main loop (lines 3-7) solves the linear relaxation of the CCut formulation for G (line 4); selects the label c with greater Z^*_c (line 5); add it to the solution (line 6); and contracts the current solution graph G to reflect the inclusion of c in C (line 7). The loop stops when the graph has only one vertex, then the solution C is returned (line 8).

Algorithm 7.2: Round and contract heuristic

```
1 procedure R&C($G = (V, E, L)$)
2     Let $C \leftarrow \emptyset$ be the set of selected labels;
3     while $|V| > 1$ do
4         $Z^* \leftarrow \text{CCut}^*(G)$;
5         $c \leftarrow argmax_{l \in L \backslash C}(Z^*_l)$;
6         $C \leftarrow C \cup \{c\}$;
7         $G \leftarrow G \mathbin{/\!/} \{c\}$ ;
8     return $C$;
```

Figure 7.2 brings an example when the R&C heuristic leads to a solution better the one generated by the MVCA. Fig 7.2(a) presents the input ELG and the LR of CCut. Fig 7.2(b) and (c) illustrates the choices of MVCA, which lead to the solution $\{A, B, C\}$. In its turn, Fig 7.2(d) and (e) illustrates the choices of R&C, which lead to the solution $\{A, B\}$.

7.2.3 A local search heuristic based on local branching techniques

The local branching technique, as proposed by Fischetti and Lodi (2003), is a generic strategy designed to improve the heuristic behavior of MIP solvers. It consists to solve the MIP with

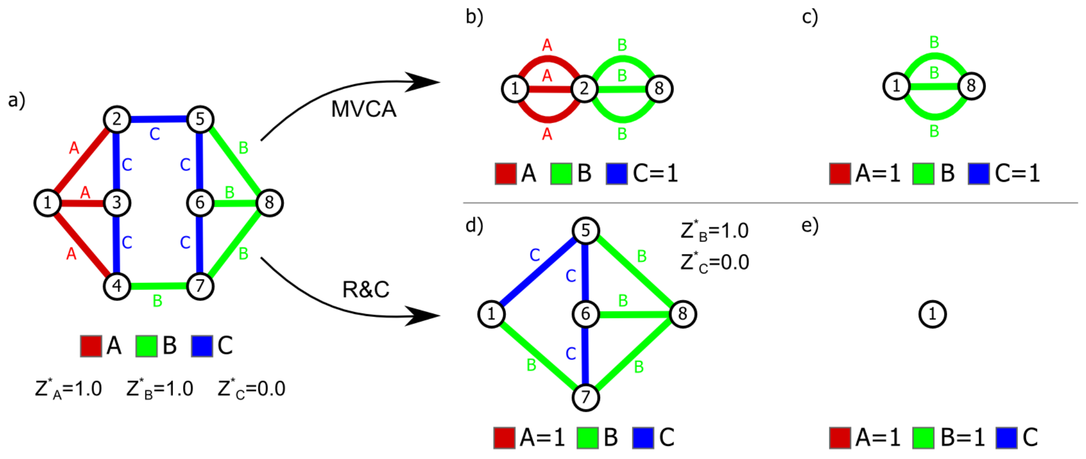


Figure 7.2: Comparison between the heuristics MVCA and R&C. **(a)** Input graph. **(b-c)** Execution of the MVCA that produces a solution with three labels. **(d-e)** Execution of the R&C that produces a solution with two colors. From (a) to (b) there is a contraction for edges with the chosen label A. After the selection of the label B, the graph in (d) becomes trivial

an additional constraint in order to achieve good incumbent solutions at very early stages of the computation. More formally, let x be a set of binary decision variables for a MIP, I be the indices of x, $\hat{x}$ be a reference solution, and $\Delta(x,\hat{x}) = \sum_{i \in I} |x_i - \hat{x}_i|$. Given an integer $k > 0$, the method partitions the solution space by means of the disjunction $\Delta(x,\hat{x}) \leq k$ (left branch) or $\Delta(x,\hat{x}) \geq k+1$ (right branch). Then, the left branch subproblem is solved first to search for better incumbent solutions in the subspace near to $\hat{x}$, while the right branch subproblem is responsible for solving the remaining problem.

This section proposes a new family of local search heuristics for the GMLSTP, denominated LB, inspired on the local branching technique. Since there is no compromise with optimality, there is no interest in solving the right branch subproblem. Furthermore, to have more control of the method, we have re-partitioned the left branch into the yet smaller subproblems: $\Delta(x,\hat{x}) = 1$, $\Delta(x,\hat{x}) = 2$, $\cdots$, $\Delta(x,\hat{x}) = k$.

Given $C \subseteq L$, a solution for the GMLSTP, B an upper bound, and an integer $2 \leq k \leq |C|$, the local search heuristic LB$_k$ consists in solving the CCut formulation along with the constraints (7.1) and (7.2) within a time limit of k seconds. The constraint (7.1) forces CCut to search only for solutions that use exactly $|C| - k$ labels of C, while the inequality (7.2) discards solutions equal to or worse than the given upper bound. In other words, unless the timeout is reached, an execution of the procedure LB$_k$ for a reference solution C returns a solution that uses $|C| - k$ labels of C and improves the given upper bound, if such a solution exists.

$$\sum_{l \in C} z_l = |C| - k. \tag{7.1}$$

$$\sum_{l \in L} z_l \leq B - 1. \tag{7.2}$$

Remark that, as in the R&C heuristic, the CCut formulation is initialized with the singleton constraints (4.5) in the place of colorful cuts inequalities (4.2). However, the LB procedure has to care with the feasibility of the returned solutions. To this end, whenever an integer solution C' is found, it is executed a depth-first search procedure on $G[C']$ and if it is disconnected, the inequality (4.2) related to each component of $G[C']$ is added to the model. This was carried out by using the *Lazy Callbacks* feature of Cplex. Further, note that it is advantageous to configure the solver in order to prioritize feasibility over optimality.

7.2.4 The multi-start local branching metaheuristic

In this section, we present the multi-start local branching (MSLB) procedure. It is a new MIP-based hybrid metaheuristic for the MLSTP that uses the constructive heuristics rMVCA, pMVCA and R&C to yield good solutions fast, and the LB family of local search heuristics in order to improve their quality. The MSLB can be divided into four phases, namely Initialization, Tuning, Roulette, and MIP-Based, and the latter three are executed in sequence within a multi-start loop until a stop condition is reached.

The Initialization phase is responsible for performing the preliminary operations necessary for the execution of the MSLB, such as the preparation of data structures and the removal of monochromatic cycles from the input graph. Further, due to the deterministic behavior of the heuristic rMVCA, it is carried out in this phase in order to achieve an initial upper bound fast.

The Tuning phase of MSLB consists in a big number of calls to the constructive heuristic pMVCA. Recall that pMVCA is expected to be very efficient because it is based on rMVCA and that its behavior depends on the parameters δ and θ. Let $\Delta = \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$, $\Theta = \{0.1, 0.2, 0.3, 0.4\}$, and $\Omega = \{(\delta, \theta) \mid \delta \in \Delta, \theta \in \Theta\}$. The Tuning phase is presented in Algorithm 7.3. The main loop (lines 2-5) is executed once for each set of parameters $\omega \in \Omega$. Then, the heuristic pMVCA is executed $\lambda = 200$ times for the selected ω (lines 3 and 4) and the best $\overline{p} = 100$ solutions found on the overall procedure are kept in the pool of solutions P (line 5). Let $A \ominus B = (A \backslash B) \cup (B \backslash A)$ represent the symmetric difference between the sets A and B. In order to increase the diversification of the MSLB, a solution C that has $|C \ominus C'| \leq 1$ for some $C' \in P$ it is not allowed to enter P. Moreover, the values of the parameters Δ, Θ, λ, and $\overline{p}$ were chosen through empirical observations.

Algorithm 7.3: Tunning phase of MSLB

```
1 procedure Tuning($G = (V,E,L),\Omega,P,\lambda$)
2     foreach $\omega \in \Omega$ (in a random way) do
3         for $i = 1,2,\cdots,\lambda$ do
4             $C \leftarrow$ pMVCA($G,\omega$);
5             $P \leftarrow$ update_pool($P,C$);
```

The Roulette phase of MSLB uses the information obtained in the Tuning phase in order to execute the pMVCA heuristic privileging the parameters $\omega \in \Omega$ which had better performance. This strategy allows the MSLB to adapt itself to the characteristics of the input instance. Let p^{ω} be the number of solutions in P that were obtained by using pMVCA with the configuration ω, and let $p^{\Omega} = \sum_{\omega \in \Omega} p^{\omega}$. The Roulette executes the pMVCA twice the number of iterations of Tuning. For each iteration, it chooses randomly a parametrization $\omega \in \Omega$ in a biased way, where p^{ω}/p^{Ω} is the probability of ω to be chosen. The Roulette phase is presented in Algorithm 7.4. The main loop of the procedure (lines 2-5) is executed $2\lambda \cdot |\Omega|$ times. At each iteration, a set of parameters $\omega \in \Omega$ is selected (line 3), the pMVCA is carried out (line 4), and the pool of solutions P is updated (line 5). Note that the update of P reflects immediately on the probabilities of each ω to be chosen.

Algorithm 7.4: Roulette phase of MSLB

```
1 procedure Roulette($G = (V,E,L),\Omega,P,\lambda$)
2     for $i = 1,2,\cdots,(2\lambda \cdot |\Omega|)$ do
3         $\omega \leftarrow$ biased_randomly_pick($\Omega,P$);
4         $C \leftarrow$ pMVCA($G,\omega$);
5         $P \leftarrow$ update_pool($P,C$);
```

The procedures based on the CCut formulation are executed at the MIP-Based phase. The R&C heuristic is carried out first, and, due to its deterministic behavior, it is executed just once during the entire MSLB. Subsequently, the family of local search heuristics LB is applied in order to improve the quality of the solutions in P. The MIP-Based phase is described in Algorithm 7.5. First, the R&C heuristic is executed (lines 2-4). Afterward, the LB$_2$ is applied over each solution $C \in P$ (lines 5-7). Let maxLB$_k \geq 3$ be an input parameter that limits the search space of LB, let b be the value of the best solution in P, and let $P^* \subseteq P = \{C \in P \mid |C| = b\}$. The LB$_k$ is then applied over each solution in P^*, for each $k \in \{3,4,\cdots,\text{maxLB}_k\}$ (lines 8-11), in order to perform a deeper search over the most promising solutions. Note that the R&C

could have been executed at the Initialization phase, just as rMVCA. However, unlike rMVCA and pMVCA, R&C is a MIP-based heuristic and hence has a longer running time. In this case, moving R&C to the MIP-Based phase allows the MSLB to finish the Tuning and the Roulette phases much earlier.

Algorithm 7.5: MIP-Based phase of MSLB

```
1  procedure MIP-Based($G = (V,E,L)$,$P$,$maxLB_k$)
2      if first time calling this procedure then
3          $C \leftarrow$ R&C($G$);
4          $P \leftarrow$ update_pool($P,C$);
5      foreach $C \in P$ do
6          $C' \leftarrow \mathrm{LB}_2(G,C)$;
7          $P \leftarrow$ update_pool($P,C'$);
8      for $k = 3,4,\cdots,maxLB_k$ do
9          foreach $C \in P^*$ do
10             $C' \leftarrow \mathrm{LB}_k(G,C)$;
11             $P \leftarrow$ update_pool($P,C'$);
```

Lastly, at the end of the MIP-Based phase, the pool of solutions P is reinitialized and the method is redirected to the Tuning phase. Figure 7.3 shows an overview of the multi-start local branching metaheuristic.

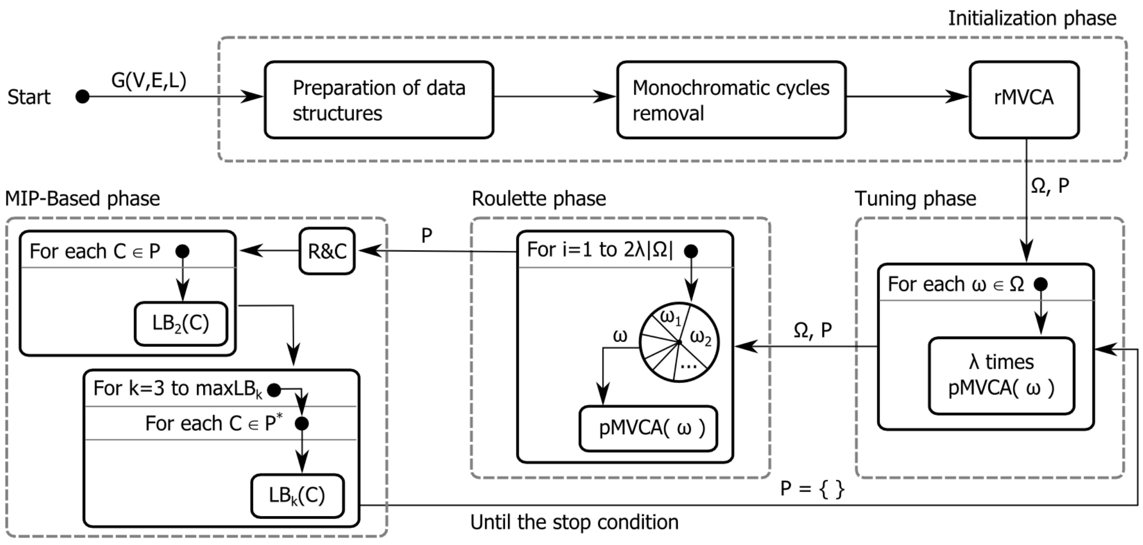


Figure 7.3: Overview of the multi-start local branching metaheuristic

7.3 Computational experiments

In this section, we perform computational experiments in order to evaluate and compare the metaheuristic proposed in this work with the state-of-the-art ones in the literature, both in terms of solution quality and computational running time. In the sequel, a statistical analysis is applied to the results. The MSLB metaheuristic, proposed in this work, is compared with the methods GRASP and VNS proposed by Consoli et al. (2009) and COMPL and INTELL proposed by Consoli et al. (2015).

7.3.1 Environment setup

The MSLB metaheuristic was implemented in C++ language and compiled by using g++ 4.8.4, with the optimization flag -O3. The CCut formulation and all of its derived procedures were implemented using the Concert library and Cplex 12.51 as the solver. The experiments were performed on a computer with Intel(R) Core(TM) i7-4790K CPU, 4.00GHz, 16 GB of RAM, and Ubuntu 14.04 as the operating system. Although the processor of this device has more than one core, the algorithm was executed by using a single core and a single thread.

In our computations we have considered the group 2 of ELGs generated by Cerulli et al. (2005), a benchmark already consolidated in the literature. Further, we also used the instances with 1000 vertices generated by Consoli et al. (2015). Thus, the group of input graphs of the experiments has instances with number of vertices $n = |V| \in \{100, 200, 500, 1000\}$, number of labels $l = |L| \in \{0.25n, 0.5n, n, 1.25n\}$, and edge densities $d \in \{0.2, 0.5, 0.8\}$. Also, each dataset consists in 10 different graphs for one n-l-d configuration. The instances of the group 1 generated by Cerulli et al. (2005) were not considered due to their very small dimensions.

7.3.2 Preliminary experiments

A first round of experiments was performed in order to determine the value of the parameter maxLB$_k$. Recall that this parameter controls the number of LB local search heuristics applied on the MIP-Based phase of MSLB. The value of maxLB$_k$ must be chosen carefully in order to provide a balance between capacity of improvement and computational cost. On the one hand, small values of maxLB$_k$ make MSLB execute fast but limit the search to a reduced portion of the solution space. On the other hand, larger values of maxLB$_k$ allow the LB$_k$ to search a wide subset of the solution space but may limit the number of solutions the method is able to address.

Four challenging datasets were chosen for this experiment: 200-200-0.2, 200-250-0.2, 500-500-0.2, and 500-625-0.2. Then, for each maxLB$_k \in \{3, 4, 5, 6, 7, 8\}$, and for each instance in the datasets, the MSLB was executed 10 times. Table 7.1 reports, for each configuration of maxLB$_k$, the total number of times the LB was able to find a solution better than the best one found in the previous phases (column #u), and the average objective function obtained for the

10 instances of the dataset (column avg.). Also, the best results for each line are highlighted in bold and the worst ones are sub-lined.

Table 7.1: Tunning of the parameter maxLB$_k$

			maxLB$_k$=3		**4**		**5**		**6**		**7**		**8**	
n	**l**	**d**	**#u**	**avg.**	**#u**	**avg.**	**#u**	**avg.**	**#u**	**avg.**	**#u**	**avg.**	**#u**	**avg.**
200	200	0.2	22	11.93	31	**11.90**	**33**	11.91	24	**11.90**	30	11.93	32	11.91
200	250	0.2	10	13.77	**16**	13.73	15	13.75	15	13.75	15	**13.72**	13	13.74
500	500	0.2	21	**15.34**	20	15.38	**24**	15.40	14	15.49	19	15.44	17	15.45
500	625	0.2	8	18.07	8	18.10	16	18.03	10	18.08	17	18.03	**18**	**18.01**
		Total	61	59.11	75	59.11	**88**	**59.09**	63	59.22	81	59.12	80	59.11

From Table 7.1 it is possible to observe that the MSLB is not very sensitive to changes in maxLB$_k$. Indeed, the difference between the best and the fourth total averages in only 0.02. Although the maxLB$_k = 5$ did not reach the best avg. for any individual dataset, this configuration always performed near the best, and, because of that, it was able to obtain the overall best results. A second round of tests was carried out with instances with $n = 1000$, from which we have noticed that the LB performs too slow for this group. It happens because the LB heuristics are based on CCut, a mathematical formulation, and it does not scale with the size of input. Based on the experiments reported here, we chose do not execute any LB for instances with $n \geq 1000$, and set maxLB$_k = 5$ for instances with $n \leq 500$.

7.3.3 Performance analysis

The next experiments aim to compare the MSLB with the methods GRASP, VNS, COMPL (CMPL in the tables), and INTELL in terms of solution quality and computational running time. The MSLB was executed using a maximum allowed CPU time (*max-time*) as stopping condition. The *max-time* configuration was the same as the used by Consoli et al. (2015): 20, 60, 300, and 1000 seconds for instances with 100, 200, 500, and 1000 vertices, respectively. For each execution, the best solution found was recorded as well as the time at which it was obtained.

The results of these experiments are reported in Tables 7.2 - 7.9. Each line of the tables represents a dataset, which is a set with 10 instances with the same *n*-*l*-*d* configuration. The first two columns of each table identify the input instances. The next column reports the optimal solution, obtained by the A^* algorithm (Chang and Leu, 1997), and the remaining columns show the computational results of the considered metaheuristics. The results reported in the columns

Opt, GRASP, VNS, CMPL, and INTELL are the ones given in Consoli et al. (2015), who performed one single execution of the methods for each instance. Conversely, we choose to perform 10 runs of the MSLB for each dataset, in order to mitigate the effects of randomness in the evaluation of the method.

The last four columns refer to the results of MSLB: the average objective function over all the ten runs (column *avg.*), the best/worst average objective function obtained in a single run (column *b.r./w.r.*), and the average objective function considering the best result found for each individual instance over all the 10 runs (column *best*). The computational times reported in the tables are the average times (in seconds) at which the best solutions were obtained. For the columns *b.r.*, *w.r.*, and *best*, the average result is reported in the case of ties. When a metaheuristic reaches the optimum, the value is sub-lined to highlight this fact. Just as in Consoli et al. (2015), the performance of an algorithm is considered better than another if either it obtained a smaller average objective function value, or an equal average objective function value but in a shorter computational running time.

Tables 7.2 and 7.3 report the results for instances with $n = 100$. All the methods perform very well for this group. Indeed, they are able to find all the optimal values in a very small amount of time. However, it is possible to see that the MSLB needs less time than the others to yield the same results. Almost the same happens for instances with 200 vertices (Tables 7.4 and 7.5). But, for instances with $l \geq 100$ and $d = 0.2$, the MSLB yielded better average objective values. Moreover, the MSLB was able to improve the best-known solution (BKS) for two datasets. These values are highlighted with a '*'.

Table 7.2: Computational results for instances with $n = 100$

							MSLB			
l	**d**	**Opt**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r.**
25	0.8	1.8	**1.8**	**1.8**	**1.8**	**1.8**	**1.8**	1.8	1.8	1.8
	0.5	2	**2**	**2**	**2**	**2**	**2**	2	2	2
	0.2	4.5	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	4.5	4.5	4.5
50	0.8	2	**2**	**2**	**2**	**2**	**2**	2	2	2
	0.5	3	**3**	**3**	**3**	**3**	**3**	3	3	3
	0.2	6.7	**6.7**	**6.7**	**6.7**	**6.7**	**6.7**	6.7	6.7	6.7
100	0.8	3	**3**	**3**	**3**	**3**	**3**	3	3	3
	0.5	4.7	**4.7**	**4.7**	**4.7**	**4.7**	**4.7**	4.7	4.7	4.7
	0.2	-	9.8	**9.7**	**9.7**	**9.7**	**9.7**	9.7	9.7	9.7
125	0.8	4	**4**	**4**	**4**	**4**	**4**	4	4	4
	0.5	5.2	**5.2**	**5.2**	**5.2**	**5.2**	**5.2**	5.2	5.2	5.2
	0.2	-	**11**	**11**	**11**	**11**	**11**	11	11	11
Total		-	57.7	**57.6**	**57.6**	**57.6**	**57.6**	57.6	57.6	57.6

Table 7.3: Computational CPU-time results for instances with $n = 100$

							MSLB			
l	**d**	**Opt**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r.**
25	0.8	0.003	0.002	**0.000**	**0.000**	**0.000**	**0.000**	0.000	0.000	0.000
	0.5	0.006	0.003	**0.000**	0.003	0.002	**0.000**	0.000	0.000	0.000
	0.2	0.027	0.003	0.003	0.002	0.002	**0.001**	0.001	0.001	0.001
50	0.8	0.005	0.002	0.005	0.005	**0.000**	**0.000**	0.000	0.000	0.000
	0.5	0.033	0.031	0.019	0.017	0.014	**0.001**	0.001	0.001	0.001
	0.2	12.100	0.044	0.027	0.067	0.016	**0.003**	0.003	0.003	0.003
100	0.8	0.225	0.093	0.223	0.237	0.077	**0.003**	0.003	0.003	0.003
	0.5	2.600	0.014	0.034	0.116	0.033	**0.002**	0.002	0.002	0.002
	0.2	-	0.939	0.406	0.272	0.233	**0.119**	0.119	0.119	0.119
125	0.8	0.515	0.003	0.008	0.008	**0.000**	**0.000**	0.000	0.000	0.000
	0.5	17.600	0.362	0.507	0.631	0.105	**0.014**	0.014	0.014	0.014
	0.2	-	0.448	0.435	0.357	0.292	**0.200**	0.200	0.200	0.200
Total		-	1.943	1.665	1.714	0.772	**0.342**	0.342	0.342	0.342

The analysis of the results for instances with $n = 500$, reported in Tables 7.6 and 7.7, can be divided in three groups. The methods obtained a similar performance for the datasets in the five first lines in terms of average objective value, but the MSLB achieved the best times. INTELL and COMPL were the best ones in the dataset 500-625-0.2. For the remaining datasets, it is possible to state that the MSLB performed better than the others both in terms of solution quality and running time. Again, the MSLB was able to improve the BKS for five datasets (values with a '*').

Table 7.4: Computational results for instances with $n = 200$

							MSLB			
l	**d**	**Opt**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r.**
50	0.8	2	**2**	**2**	**2**	**2**	**2**	2	2	2
	0.5	2.2	**2.2**	**2.2**	**2.2**	**2.2**	**2.2**	2.2	2.2	2.2
	0.2	5.2	**5.2**	**5.2**	**5.2**	**5.2**	**5.2**	5.2	5.2	5.2
100	0.8	2.6	**2.6**	**2.6**	**2.6**	**2.6**	**2.6**	2.6	2.6	2.6
	0.5	3.4	**3.4**	**3.4**	**3.4**	**3.4**	**3.4**	3.4	3.4	3.4
	0.2	NF	8.1	**7.9**	8	**7.9**	**7.9**	7.9	7.9	7.9
200	0.8	4	**4**	**4**	**4**	**4**	**4**	4	4	4
	0.5	NF	**5.4**	**5.4**	**5.4**	**5.4**	**5.4**	5.4	5.4	5.4
	0.2	NF	12.2	12	12.1	12	**11.9**	11.9*	11.9	11.9
250	0.8	4	4.1	**4**	4.1	**4**	**4**	4	4	4
	0.5	NF	**6.3**	**6.3**	**6.3**	**6.3**	**6.3**	6.3	6.3	6.3
	0.2	NF	13.9	13.9	13.9	13.9	**13.77**	13.7*	13.7	13.8
Total		-	69.4	68.9	69.2	68.9	**68.67**	68.6	68.6	68.7

Table 7.5: Computational CPU-time results for instances with $n = 200$

							MSLB			
l	**d**	**Opt**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r.**
50	0.8	0.016	0.008	0.003	**0.000**	**0.000**	**0.000**	0.000	0.000	0.000
	0.5	0.016	0.009	0.012	0.016	0.003	**0.000**	0.000	0.000	0.000
	0.2	4.300	0.020	0.229	0.205	0.190	**0.002**	0.002	0.002	0.002
100	0.8	0.087	0.031	0.120	0.446	0.022	**0.005**	0.005	0.005	0.005
	0.5	0.727	0.076	0.072	0.173	0.055	**0.004**	0.004	0.004	0.004
	0.2	NF	**0.513**	1.700	7.900	1.600	0.695	0.695	0.695	0.695
200	0.8	22.100	0.036	0.028	0.059	0.018	**0.000**	0.000	0.000	0.000
	0.5	NF	0.552	1.100	2.500	0.463	**0.028**	0.028	0.028	0.028
	0.2	NF	7.200	12.800	**2.300**	11.500	5.207	5.207	5.207	5.207
250	0.8	19.200	4.800	1.300	1.800	1.100	**0.152**	0.152	0.152	0.152
	0.5	NF	0.389	2.300	2.200	1.100	**0.027**	0.027	0.027	0.027
	0.2	NF	1.400	2.500	1.500	**1.300**	2.134	3.681	2.294	1.471
Total		-	15.035	22.164	19.100	17.351	**8.254**	9.801	8.414	7.591

The results for instances with $n = 1000$ (reported in Tables 7.8 and 7.9) highlights the difference in performance from MSLB to the other methods. The MSLB has improved the BKS for six datasets (values with a '*'), and also has proved the optimality for the dataset 1000-250-0.8. It was possible since the removal of monochromatic cycles detects any solution with value 1 and, after that, if any method finds a solution with value 2, this solution is optimal. Beyond that, even when the other methods are able to find the same average objective value for a dataset, much more time is needed. For instance, even reaching better solutions than INTELL, the total time of the MSLB was approximately 4.5 times smaller.

Table 7.6: Computational results for instances with $n = 500$

							MSLB			
l	**d**	**Opt**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r.**
125	0.8	2	**2**	**2**	**2**	**2**	**2**	2	2	2
	0.5	2.6	**2.6**	**2.6**	**2.6**	**2.6**	**2.6**	2.6	2.6	2.6
	0.2	NF	**6.2**	**6.2**	**6.2**	**6.2**	**6.2**	6.2	6.2	6.2
250	0.8	3	**3**	**3**	**3**	**3**	**3**	3	3	3
	0.5	NF	4.2	**4.1**	4.2	**4.1**	**4.1**	4.1	4.1	4.1
	0.2	NF	9.9	9.9	9.9	9.9	**9.8**	9.8^*	9.8	9.8
500	0.8	NF	4.7	4.7	4.9	4.7	**4.4**	4.3^*	4.3	4.5
	0.5	NF	6.5	6.5	6.5	**6.4**	**6.4**	6.4	6.4	6.4
	0.2	NF	15.9	15.8	15.8	15.8	**15.38**	15.2^*	15.3	15.5
625	0.8	NF	**5.1**	**5.1**	**5.1**	**5.1**	**5.1**	5.1	5.1	5.1
	0.5	NF	7.9	7.9	7.9	7.9	**7.54**	7.4^*	7.4	7.6
	0.2	NF	18.4	18.3	**18**	**18**	18.06	17.8^*	17.9	18.1
Total		-	86.4	86.1	86.1	85.7	**84.58**	83.9	84.1	84.9

Table 7.7: Computational CPU-time results for instances with $n = 500$

							MSLB			
l	**d**	**Opt**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r.**
125	0.8	0.183	0.107	0.012	0.425	0.010	**0.008**	0.008	0.008	0.008
	0.5	0.546	0.453	1.100	0.993	0.423	**0.047**	0.047	0.047	0.047
	0.2	NF	4.100	3.800	3.800	3.500	**0.024**	0.024	0.024	0.024
250	0.8	4.900	0.190	0.042	0.551	0.034	**0.010**	0.010	0.010	0.010
	0.5	NF	**0.470**	83.700	19.200	71.300	6.679	6.679	6.679	6.679
	0.2	NF	2.500	4.900	4.700	3.700	**0.572**	0.572	0.572	0.572
500	0.8	NF	22.700	22.500	21.200	20.500	**5.825**	6.471	12.289	2.041
	0.5	NF	60.900	31.400	44.600	80.500	**1.159**	1.159	1.159	1.159
	0.2	NF	**18.900**	138.100	121.600	82.600	27.603	33.644	32.133	29.370
625	0.8	NF	4.800	10.100	20.300	4.100	**0.104**	0.104	0.104	0.104
	0.5	NF	16.600	32.200	35.200	26.700	**5.278**	6.136	5.953	3.907
	0.2	NF	50.500	156.500	137.400	97.600	**22.456**	30.088	35.343	15.322
Total		-	182.220	484.355	409.969	390.967	**69.765**	84.941	94.321	59.243

Considering the above, it is possible to classify the benchmark into easy instances and hard ones. The majority of methods is able to reach the best solution for the easy instances. For this group, the MSLB yield these solutions faster. It is mostly on account of the Tuning and Roulette phases, both relying on the pMVCA. On the other hand, for the hard datasets, the MSLB was able to find the best solutions. It was due to the capacity of intensification given by the LB family of local search heuristics.

Moreover, from the columns *b.r.* and *w.r.*, is possible to see that the gap between the best and the worst runs of MSLB is very small. Indeed, this difference is 0 for 41 of 48 datasets. For the remaining cases, this difference is ≤ 0.2 in 6 cases and 0.3 in the other one. Finally, although MSLB has been executed on a different machine from the other algorithms, both microprocessors have the same 4.0 GHz clock speed, what attenuates this difference.

Table 7.8: Computational results for instances with $n = 1000$

							MSLB			
l	**d**	**Opt**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r.**
250	0.8	NF	2.1	2.1	2.1	2.1	<u>**2**</u>	2*	2	2
	0.5	NF	3.3	3.3	3.3	3.3	**3**	3*	3	3
	0.2	NF	5.2	**5**	5.2	**5**	**5**	5	5	5
500	0.8	NF	**3**	**3**	**3**	**3**	**3**	3	3	3
	0.5	NF	4.2	4.1	4.2	**4**	**4**	4	4	4
	0.2	NF	9.8	9.6	9.6	9.5	**9**	9*	9	9
1000	0.8	NF	5.1	5.1	**5**	**5**	**5**	5	5	5
	0.5	NF	7.7	7.5	7.2	**7**	**7**	7	7	7
	0.2	NF	14.9	14.9	14.9	14.8	**14.29**	14*	14.2	14.5
1250	0.8	NF	6.2	6.2	6.1	**6**	**6**	6	6	6
	0.5	NF	8.9	8.8	8.9	8.6	**8.18**	8.1*	8.1	8.3
	0.2	NF	17.8	17.7	17.6	17.1	**17**	17*	17	17
	Total	-	88.2	87.3	87.1	85.4	**83.47**	83.1	83.3	83.8

Table 7.9: Computational CPU-time results for instances with $n = 1000$

						MSLB			
l	**d**	**GRASP**	**VNS**	**CMPL**	**INTELL**	**avg.**	**best**	**b.r.**	**w.r**
250	0.8	1.2	0.918	1.2	0.796	**0.023**	0.023	0.023	0.023
	0.5	1.2	0.563	1.5	0.459	**0.022**	0.022	0.022	0.022
	0.2	24.3	43.800	50.1	25.100	**0.176**	0.176	0.176	0.176
500	0.8	2.1	0.730	1.5	0.701	**0.034**	0.034	0.034	0.034
	0.5	102.3	237.100	162.1	92.200	**1.954**	1.954	1.954	1.954
	0.2	94.1	91.100	90.5	82.300	**0.027**	0.027	0.027	0.027
1000	0.8	36.6	28.100	29.8	26.900	**0.069**	0.069	0.069	0.069
	0.5	115.8	135.800	205.1	134.800	**0.978**	0.978	0.978	0.978
	0.2	211.9	237.100	226.1	208.900	**169.177**	238.247	217.095	71.880
1250	0.8	39.9	25.100	25.3	25.100	**0.050**	0.050	0.050	0.050
	0.5	530.2	561.400	506.4	408.300	**171.951**	186.899	198.837	128.758
	0.2	616.1	697.100	610.4	644.400	**5.320**	5.320	5.320	5.320
Total		1775.7	2058.811	1910.0	1649.955	**349.781**	433.799	424.585	209.290

7.3.4 Statistical analysis

A statistical analysis was carried out to enrich the performance analysis of the computational experiments. These tests were conducted using the software R 3.0.2. and the package PMCMR 4.1. For the first test, the null hypothesis is that there is no significant difference between the algorithms and, hence, the deviations obtained were merely random. The Friedman test is a nonparametric procedure based on ranks that detects the existence of significant differences between the results of multiple multiple test attempts (e.g. for multiple algorithms) over different groups of instances.

From the results reported in Section 7.3.3 and considering the column *avg.* for the MSLB, the algorithms received ranks from 1 to 5 (best to worst) for each dataset. The ties were broken by the average of the ranks. The average ranks obtained considering all datasets of the benchmark was: 3.990, 3.708, 4.073, 2.104, and 1.125, for the metaheuristics GRASP, VNS, COMPL, INTELL, and MSLB, respectively. Through the routine `friedman.test`, the statistic value obtained was $\chi_F^2 = 136.867$, with p-value $= 2.2\text{e}{-16}$. Indeed, the null hypothesis can be rejected with a level of significance $\alpha = 0.01$ and 99% of confidence. Thus, we are able to detect significant differences between the algorithms on a pairwise comparison through *post-hoc* tests.

The Table 7.10 presents the p-values obtained by the Nemenyi test through the routine `posthoc.friedman.nemenyi.test`. The p-values of Nemenyi test indicate that the performance of MSLB is significantly better than GRASP, VNS, and COMPL for $\alpha = 0.01$ and 99% of confidence, as well as its performance is significantly better than INTELL for $\alpha = 0.05$ and 95% of confidence.

Table 7.10: The p-values obtained by pairwise comparisons through the Nemenyi test

	GRASP	VNS	CMPL	INTELL
VNS	0.91	-	-	-
CMPL	1.00	0.79	-	-
INTELL	5.2e-08	6.6e-06	1.1e-08	-
MSLB	**3.7e-14**	**7.8e-14**	**6.2e-14**	**0.02**

The tests performed strongly suggest that the results analyzed in the section 7.3.3 were not the result of randomness. The reader is referred to Demšar (2006) and Derrac et al. (2011) for more details about Friedman and Nemenyi tests and others nonparametric statistics, in addition to a more in-depth discussion on the power of these procedures.

7.4 Concluding remarks

In the present chapter, we have proposed a revised version of the MVCA, which is the most important constructive heuristic for both MLSTP and GMLSTP. We have performed a complexity analysis and proved the time complexity of the rMVCA is $O(\alpha_n kn)$, a tighter bound for this heuristic. Further, we have proposed the MSLB, a new MIP-based metaheuristic that combines the efficiency of a parametrized version of rMVCA with the capacity of exploration of a new local search method based on local branching techniques. Lastly, we have carried out computational experiments to compare the MSLB with the state-of-the-art metaheuristics for the MLSTP. The results show MSLB achieved the best performance both in quality of solutions and processing times.

Part III

Related Problems

The last part of this thesis is focused on problems related to the MLSTP, that is to say, connectivity problems defined over edge-labeled graphs. First, we propose new mathematical formulations based on CCut for solving several problems defined over ELGs. In the sequel, we discuss in more detail the minimum labeling global cut problem. Finally, we introduce the minimum representation spanning tree problem, prove it is NP-complete, and propose heuristic and exact algorithms to solve it.

Chapter 8

Exact Methods for Connectivity Problems defined on ELGs

As discussed in the previous chapters, given an ELG $G = (V, E, L)$, the minimum labeling spanning tree problem aims to find a minimum cardinality subset of labels $L' \subseteq L$ such that the spanning subgraph induced by the set of edges with label in L' is connected. Moreover, it is possible to state that the MLSTP is the most studied problem among the ones defined on ELGs (refer to Chapter 2 for more information on the literature of the MLSTP). Other examples of interesting problems defined on ELGs are the labeled maximum matching problem (Carrabs et al., 2009), the maximum labeled clique problem (Carrabs et al., 2014), and the rainbow cycle cover problem (Silvestri et al., 2016).

In particular, many problems formulated on ELGs have the connectivity as a subjacent objective. This kind of problem has been the subject of research in recent years, as in the works of Carrabs et al. (2017) on the rainbow spanning forest problem, Ismkhan (2017) on the colorful traveling salesman problem and Cerrone et al. (2017) on the minimum labeling spanning tree problem. The two main goals of this chapter are (i) to describe connectivity problems defined on ELGs that can possibly inspire new researches (and researchers), and (ii) to propose new MIP-based mathematical models, extensions and/or adaptations of the colorful cuts formulation (refer to Section 4.1), for solving some of these problems. In the sequel, we briefly present five connectivity problems defined on ELGs.

The colorful traveling salesman problem: The colorful traveling salesman problem (CTSP) was proposed independently by Cerulli et al. (2006b) (as the minimum labelling Hamiltonian cycle problem) and by Xiong et al. (2007). Both have proved the problem is NP-Complete on complete graphs. Given an ELG, the goal of the CTSP is to find a Hamiltonian tour with the minimum number of labels. Figure 8.1 illustrates the CTSP. Fig. 8.1(a) brings an input ELG

$G = (V,E,L)$. Fig. 8.1(b) depicts a solution with 3 labels while Fig. 8.1(c) shows a solution with only 2 labels.

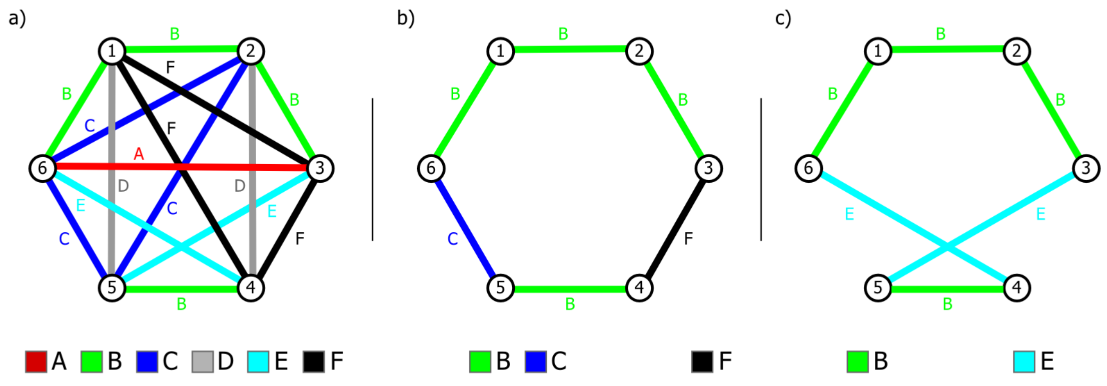


Figure 8.1: The colorful traveling salesman problem. **(a)** The input ELG $G = (V,E,L)$. **(b)** A solution for G using the labels B, C, and F. **(c)** A solution for G using the labels B and E

The k-labeled spanning forest problem: The k-labeled spanning forest problem (kLSFP), proposed by Cerulli et al. (2014), consists in: given an ELG $G = (V,E,L)$ and an integer constant $k \in \mathbb{Z}^+$, find the minimum number of connected components by using at most k labels. According to the authors, this kind of constraint is important in real life situations when there is a limit that cannot be violated, such as the number of wireless network frequencies in some area. Moreover, the authors have demonstrated the kLSFP is NP-Complete since it is a generalization of the MLSTP. Figure 8.2 brings an example of the kLSFP for $k = 4$. Fig. 8.2(a) depicts the input ELG $G = (V,E,L)$. Fig. 8.2(b) presents $G[\{D,E,G,H\}]$, an optimal solution for this instance which represents a spanning forest with four maximal connected components.

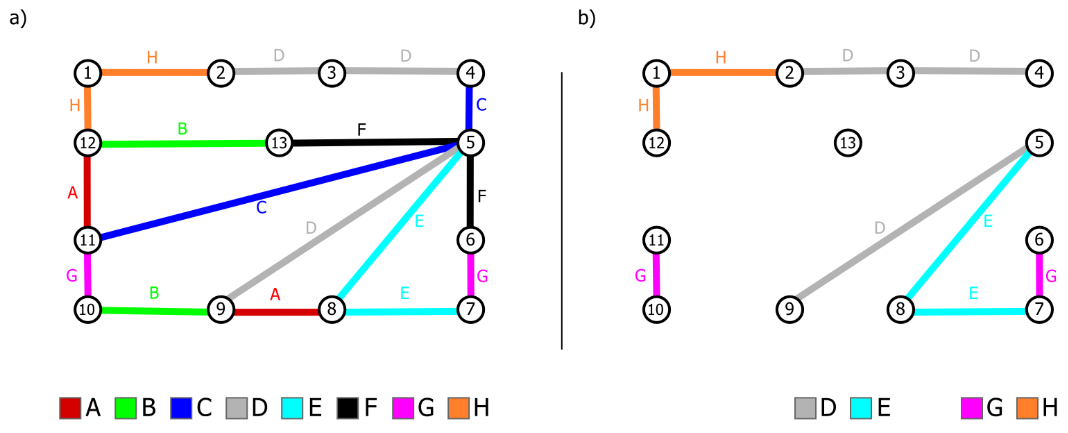


Figure 8.2: The k-labeled spanning forest problem. **(a)** The input ELG $G = (V,E,L)$. **(b)** A solution for G where $G[\{D,E,G,H\}]$ has 4 maximal connected components

The label-constrained minimum spanning tree problem: Given an edge-labeled-and-weighted graph $G = (V,E,L)$ and an integer constant $k \in \mathbb{Z}^+$, the label-constrained minimum spanning tree problem (LCMSTP), proposed by Xiong et al. (2008), aims to find a spanning tree $T = (V, E', L')$ of G such that the total edge weight is minimized and $|L'| \leq k$. The authors proved the LCMSTP is NP-hard by reduction from the MLSTP.

The cost-constrained minimum label spanning tree problem: Given an edge-labeled-and-weighted graph $G = (V, E, L)$ and an integer constant $k \in \mathbb{Z}^+$, the cost-constrained minimum label spanning tree problem (CCMLSTP), also proposed by Xiong et al. (2008), aims to find a spanning tree with the smallest number of labels and a total edge cost of no more than k.

The minimal cost/minimal label spanning tree problem: The minimal cost/minimal label spanning tree problem (MC/MLSTP), proposed by Clímaco et al. (2010), is a bicriterion multi-objective problem that combines the MLSTP and the minimum spanning tree problem. Given an edge-labeled-and-weighted graph, the first objective is to minimize the sum of the weights of the edges of the resulting spanning tree, while the second aims to minimize its number of labels. According to Clímaco et al. (2010), these two criterion are conflicting in most cases. Figure 8.3 illustrates the MC/MLSTP, highlighting the conflicting criteria. Fig. 8.3(a) brings the edge-labeled-and-weighted input graph. Fig. 8.3(b) presents a solution with 4 labels and 24 as the sum of the edge weights. Notwithstanding, Fig. 8.3(c) presents a solution with 5 labels and 21 as the sum of the edge weights.

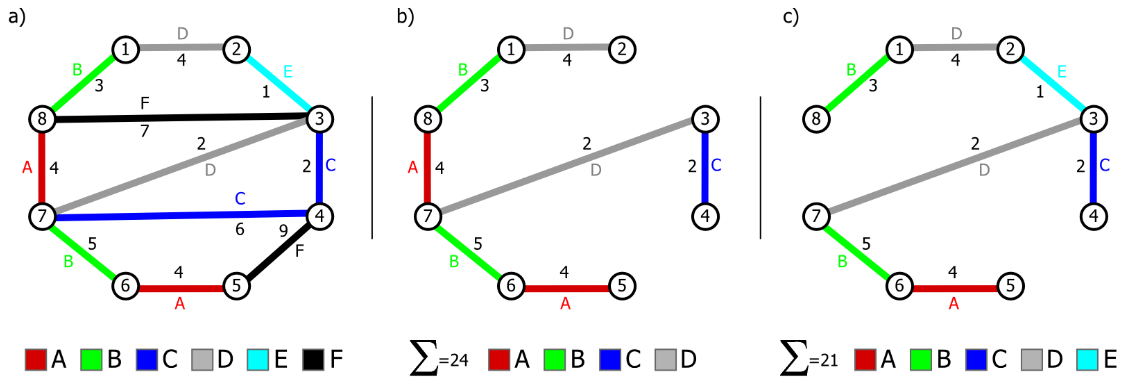


Figure 8.3: Example of the minimal cost/minimal label spanning tree problem. **(a)** The input edge-labeled-and-weighted graph. **(b)** A solution with 4 labels and edge cost 24. **(c)** A solution with 5 labels and edge cost 21

The next sections discuss in more detail some connectivity problems defined on ELGs, introduce new ones, and propose new MIP-based mathematical models, extensions and/or adaptations of the colorful cuts formulation, for solving these problems. We add a "$\star$" before the name of the problem in order to emphasize, to the best of our knowledge, when it is being first proposed in this work. Otherwise, we cite the work where the problem was introduced.

8.1 Connectivity problems with optional vertices

This section addresses a class of connectivity problems defined on ELGs in which it is not necessary to connect the entire graph, but only specific subsets of its vertices. The objective is to show how to adapt the CCut formulation to solve each of these problems.

The minimum labeling path problem (Jacob et al., 1999): Given an input ELG $G = (V, E, L)$, and a pair of vertices $s, t \in V$, the aim of the minimum labeling path problem (MLPP) is to find a path connecting s to t by using the minimum number of labels. According to Carr et al. (2000) and Broersma et al. (2005), the MLPP is NP-Complete. Note that the MLPP can be equivalently defined in the following way: given an input ELG $G = (V, E, L)$, the goal is to find a smallest cardinality subset $L' \in L$ such that s and t are connected in $G[L']$. Verifying the equivalence between these definitions is straightforward (refer to the demonstration of Definition 1.4).

The CCut formulation adapted to the MLPP is presented in the program (8.1) through (8.3). The objective function (8.1) minimizes the number of labels, and the exponential set of constraints (8.2) ensures the connectivity of the vertices s and t in the solution graph by requiring at least one active label for every colorful cut that separates these vertices.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.1}$$

$$\text{s.t.} \quad \sum_{l \in K(S)} z_l \geq 1, \qquad \forall S \subset V, s \in S, t \notin S, \tag{8.2}$$

$$z_l \in \{0, 1\}, \qquad \forall l \in L. \tag{8.3}$$

As discussed in Section 3.3, the solution of the MLPP for any pair $s, t \in V$ is a lower bound for the MLSTP. In such case, it could be interesting to find out the best of these bounds, denominates as the *labeled diameter* of an ELG. A question arises: is that possible to compute the labeled diameter at once, instead of computing the MLPP for every s, t pair of the graph?

The min-color generalized forest problem (Carr et al., 2000): Given an ELG $G = (V, E, L)$, and set of pairs of distinct vertices $\mathcal{V} = \{(s_1, t_1), (s_2, t_2), \cdots\}$, the min-color generalized forest problem (MCGFP), proposed by Carr et al. (2000), is a generalization of the MLPP that aims to find a smallest cardinality subset $L' \in L$ such that every pair $(s, t) \in \mathcal{V}$ is connected in $G[L']$. The MCGFP is NP-hard, since the case when $|\mathcal{V}| = 1$ is exactly the MLPP.

The CCut formulation adapted to the MCGFP is presented in the program (8.4) through

(8.6). The objective function (8.4) minimizes the number of labels, and the exponential set of constraints (8.5) ensures that each pair of vertices $(s,t) \in \mathcal{V}$ is connected in the solution graph by requiring at least one active label for every colorful cut that separates these vertices.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.4}$$

$$\text{s.t.} \quad \sum_{l \in K(S)} z_l \geq 1, \qquad \forall S \subset V \mid \exists\, (s_i,t_i) \in \mathcal{V} \text{ such that } s_i \in S \text{ and } t_i \notin S, \tag{8.5}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.6}$$

Figure 8.4 depicts both the MLPP and the MCGFP. Fig. 8.4(a) presents an input ELG $G = (V,E,L)$. Fig. 8.4(b) evidences that the solution of the MLPP on G for $s = 1, t = 2$ is $\{D\}$, for $s = 5, t = 4$ is $\{F\}$, and for $s = 7, t = 6$ is $\{C\}$. Fig. 8.4(c) shows that the solution for the MCGFP on G for $\mathcal{V} = \{(1,2),(5,4),(7,6)\}$ is $\{A,B\}$.

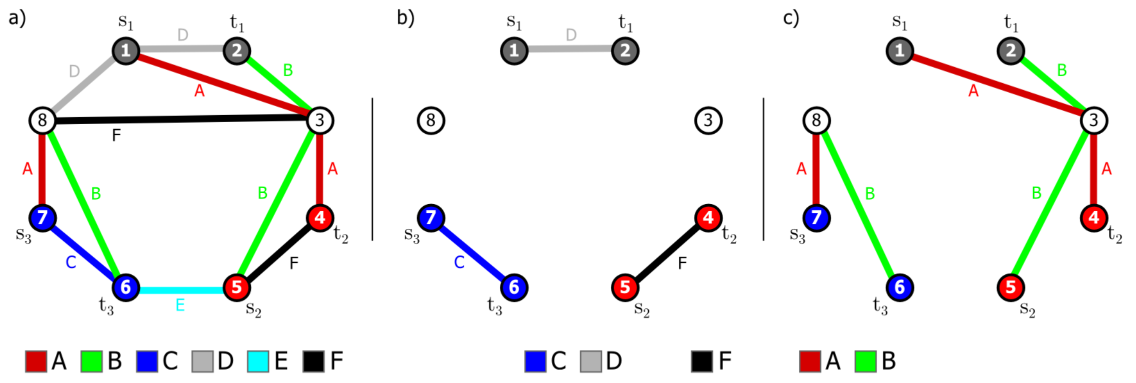


Figure 8.4: Example of the MLPP and of the MCGFP. **(a)** The input ELG $G = (V,E,L)$. **(b)** The solution for three separate instances of the MLPP on G. **(c)** The solution for the MCGFP for $\mathcal{V} = \{(1,2),(5,4),(7,6)\}$

The minimum labeling Steiner problem (Cerulli et al., 2006a): Given an undirected weighted graph $G = (V,E)$ and a subset of its vertices $Q \subseteq V$, namely terminals, the Steiner tree problem in graphs (STP) consists in finding a minimum cost subgraph of G such that the set of terminals is connected. To do so, some vertices from the set $V \backslash Q$, namely Steiner vertices, could be used. In its turn, the minimum labeling Steiner problem (MLSteiner), proposed by Cerulli et al. (2006a), extends both the MLSTP and the STP. Formally, given an ELG $G = (V,E,L)$ and a set of terminals $Q \subseteq V$, the aim of the MLSteiner is to find a minimum cardinality set of labels $L' \subseteq L$ such that all the vertices of Q belong to the same connected component in $G[L']$. The MLSteiner is NP-hard since it is exactly the MLSTP if $Q = V$.

The CCut formulation adapted to the MLSteiner problem is presented in the program (8.7) through (8.9). The objective function (8.7) minimizes the number of labels, and the exponential set of constraints (8.8) ensures that the vertices in Q are connected in the solution graph by requiring at least one active label for every colorful cut that separates them.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.7}$$

$$\text{s.t.} \quad \sum_{l \in K(S)} z_l \geq 1, \qquad \forall S \subset V, S \cap Q \neq \emptyset, S \cap Q \neq Q, \tag{8.8}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.9}$$

Figure 8.5 illustrates the MLSteiner problem. Fig. 8.5(a) shows the input ELG $G = (V,E,L)$, where $Q = \{1,2,6,7\}$ is the set of terminals, represented in black. Fig. 8.5(b) and 8.5(c) present two solutions with cost 3 and 2, respectively. Observe that both solutions use the Steiner vertex $v = 3$.

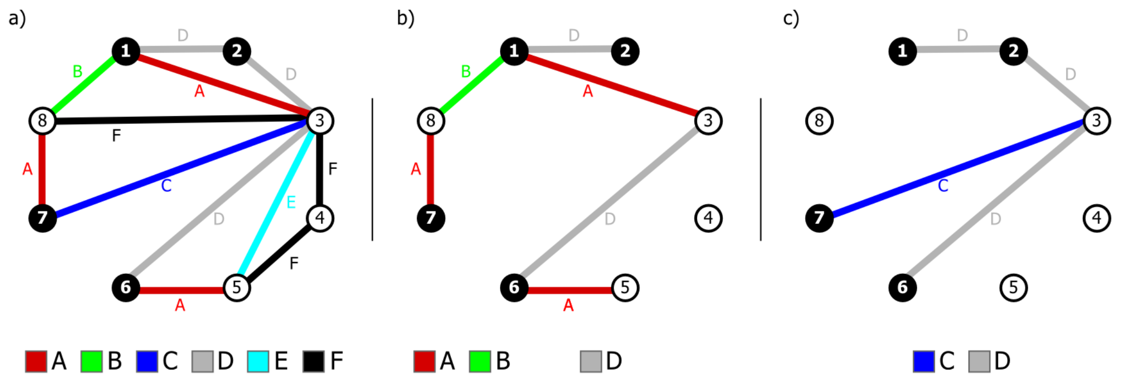


Figure 8.5: Example of the minimum labeling Steiner problem. **(a)** An input ELG, where the set of terminals are represented in black. **(b)** A solution with 3 labels using the Steiner vertex 3. **(b)** A solution with 2 labels using the Steiner vertex 3

⋆ **The prize-collecting minimum labeling tree problem:** Motivated by a real world application on transport (or computer) networks, we propose the prize-collecting minimum labeling tree problem (PC-MLT), a more general connectivity problem defined on ELGs that extends the MLSTP, the MLPP and the MLSteiner. The PC-MLT consists in finding a minimum-labeled tree such that the sum of its prizes is greater than a required value. To the best of our knowledge, this is the first study on the PC-MLT. This problem has applications when it is desirable to achieve certain objective (e.g. to provide access to some resource to a certain number of people) by using a minimum number of labels. Observe from Balas (1989) that the idea of prize-collecting optimization problems is not new.

Formally, let $G = (V, E, L)$ be an ELG, $P : V \to \mathbb{R}$ be a pricing function on the vertices of G, and $PMIN$ an input constant representing the minimum prize to be collected. The aim of the PC-MLT is to find a minimum cardinality set $L' \subset L$ such that $G[L']$ has a connected component spanning the set of vertices $Q \subseteq V$, namely the solution set, and $\sum_{v \in Q} P(v) \geq PMIN$. Observe that the MLSTP is a special case of the PC-MLT: let $P(v) = 1$, $\forall v \in V$, and $PMIN = |V|$. Therefore, since the MLSTP is NP-hard, the PC-MLT also belongs to this class of problems.

Figure 8.6 illustrates the prize-collecting minimum labeling tree problem. Fig. 8.6(a) presents an input ELG for the PC-MLT with $PMIN = 22$. The prizes are presented near each vertex of G. Fig 8.6(b) shows a solution for this instance with $L' = \{A, C, D\}$, $Q = \{2, 3, 4, 5, 8, 9, 11, 12\}$, and $\sum_{v \in Q} P(v) = 27$. The vertices of $V \backslash Q$ are in gray.

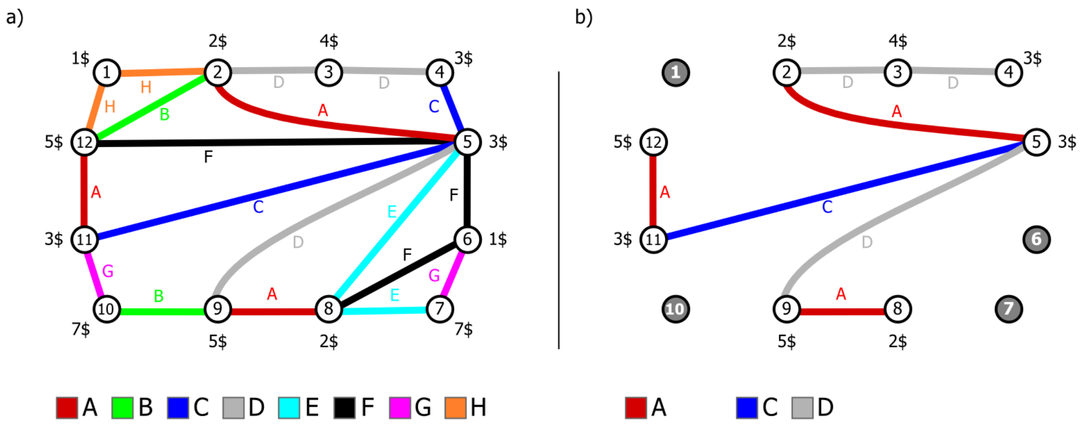


Figure 8.6: Example of the prize-collecting minimum labeling tree problem. **(a)** An input ELG with prizes associated with its vertices. **(b)** A solution for this instance with $L' = \{A, C, D\}$, $Q = \{2, 3, 4, 5, 8, 9, 11, 12\}$, and $\sum_{v \in Q} P(v) = 27$

The CCut formulation adapted to the PC-MLT is presented in the program (8.10) through (8.13). In addition to the variables z_l, the model also defines the group of binary variables $w_v \in \{0, 1\}$, for which $w_v = 1$ if and only if the vertex $v \in V$ belongs to the solution set Q. The objective function (8.10) minimizes the number of labels. The exponential set of constraints (8.11) ensures that the vertices in the solution set Q are connected in the solution graph by requiring at least one active label for every colorful cut that separates them. Remark that, in contrast to the previously presented adaptations of CCut, the set of constraints (8.11) defines $O(|V|^2)$ inequalities for each proper subset S. Lastly, the constraint (8.12) ensures that the minimum required prize is collected, and the set of constraints (8.13) defines the domain of the variables.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.10}$$

$$\text{s.t.} \quad \sum_{l \in K(S)} z_l \geq w_p + w_q - 1, \qquad \forall S \subset V, S \neq \emptyset, \forall p \in S, \forall q \in V \backslash S, \tag{8.11}$$

$$\sum_{v \in V} w_v \cdot P(v) \geq PMIN, \tag{8.12}$$

$$z_l \in \{0,1\}, w_v \in \{0,1\}, \qquad \forall v \in V, \forall l \in L. \tag{8.13}$$

8.2 Problems defined on arc-labeled digraphs

In this section, we address three connectivity problems defined on arc-labeled digraphs (ALD), that are directed graphs in which each arc has one label associated (recall Definition 1.6), and show how to adapt the CCut formulation to solve them. This kind of problem arises when there are asymmetries in the links between vertices. Consider the following definitions:

Definition 8.1. *Given an ALD $D = (V,A,L)$, and a subset of labels $L' \subseteq L$, $D[L']$ is the spanning subgraph of D induced by the set of arcs $A(L') = \{a \in A \mid l^a(a) \in L'\}$.*

Definition 8.2. *For convenience, when a problem defines a special root vertex $r \in V$, let V^+ denote the set $V \backslash \{r\}$.*

$\star$ **The minimum labeling arborescence problem:** Given an ALD $D = (V,A,L)$ and a root vertex $r \in V$, the goal of the minimum labeling arborescence problem (MLAP) is to find a set of labels $L' \subseteq L$, such that $D[L']$ contains a directed path from r to each vertex $v \in V^+$, and $|L'|$ is minimized. The MLAP is NP-complete. *Mutatis mutandis*, the proof is the same given by Chang and Leu (1997) to the MLSTP (refer to Section 2.1).

The CCut formulation adapted to the MLAP is presented in the program (8.14) through (8.16). Let $K^-(S) = \{l^a(a) \mid a \in \delta^-(S)\}$ be the set of labels represented in $\delta^-(S)$, that is the set of ingoing arcs in the cut set $[S, V \backslash S]$, for any $S \subset V$. The objective function (8.14) minimizes the number of labels in the solution. In its turn, the exponential set of constraints (8.15) ensures that there is a directed path from r to any vertex $v \in V^+$ in the solution digraph by requiring at least one active label (and as a consequence an active arc) for every $K^-(S)$ that separates the root from v.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.14}$$

$$\text{s.t.} \quad \sum_{l \in K^-(S)} z_l \geq 1, \qquad \forall S \subset V, S \neq \emptyset, r \notin S, \tag{8.15}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.16}$$

⋆ **The minimum labeling strongly connected problem:** Given an ALD $D = (V,A,L)$, the minimum labeling strongly connected problem (MLSCP) aims to find a set of labels $L' \subseteq L$, such that $|L'|$ is minimized and $D[L']$ is strongly connected. Recall a directed graph is strongly connected if there is a path in each direction between each pair of its vertices. The MLSCP is NP-complete. Again, *Mutatis mutandis*, the proof is the same given by Chang and Leu (1997) to the MLSTP (refer to Section 2.1).

The CCut formulation adapted to the MLSCP is presented in the program (8.17) through (8.19). The objective function (8.17) minimizes the number of labels in the solution, and the exponential set of constraints (8.18) ensures that there is a path in each direction between each pair of vertices $s,t \in V$ in the solution digraph by requiring at least one active label (and as a consequence an active arc) for every $K^-(S)$ that separates t from s. Recall that $K^-(S) = \{l^a(a) \mid a \in \delta^-(S)\}$.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.17}$$

$$\text{s.t.} \quad \sum_{l \in K^-(S)} z_l \geq 1, \qquad \forall S \subset V, S \neq \emptyset, \tag{8.18}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.19}$$

Figure 8.7 depicts both the MLAP and the MLSCP. Fig. 8.7(a) presents an input ALD $D = (V,A,L)$. Fig. 8.7(b) shows a solution for the MLAP considering the root node $r = 8$. Fig. 8.7(c) illustrates a solution for the MLSCP.

The maximum flow minimum labeling problem (Cerulli and Granata, 2009): The maximum flow minimum labeling problem (MFMLP) is a variant of the classical maximum flow problem defined on ALDs. Given and ALD $D = (V,A,L)$, a capacity function $c : A \to \mathbb{R}$, and two vertices $s,t \in V$, the MFMLP aims to find a subset $L' \subseteq L$ with minimum cardinality, such that the flow from the source s to the sink t on $D[L']$ is maximum, i.e. it has the same value of maximum s-t flow on D. Granata et al. (2013) proved the MFMLP is NP-hard by reduction from the minimum labeled spanning tree problem.

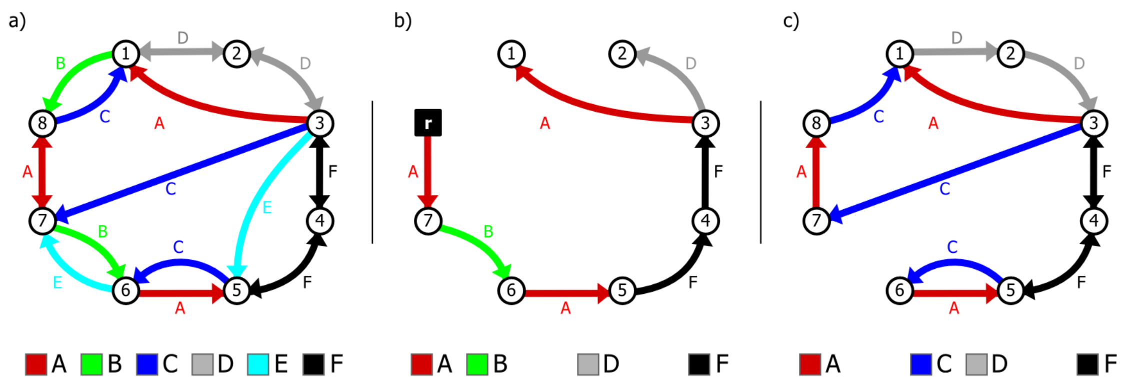


Figure 8.7: Example of both the MLAP and the MLSCP. **(a)** An input ALD $D = (V,A,L)$. **(b)** $L' = \{A,B,D,F\}$, a solution for the MLAP on D, considering the root node $r = 8$. **(c)** $L' = \{A,C,D,F\}$, a solution for the MLSCP on D

Figure 8.8 illustrates the maximum flow minimum labeling problem. Fig. 8.8(a) presents an input capacited ALD $D = (V,A,L)$. Fig. 8.8(b) shows a solution for the MFMLP with $L' = \{A,B,D,E,F\}$ and Fig. 8.8(c) shows another one with $L' = \{A,B,D,F\}$.

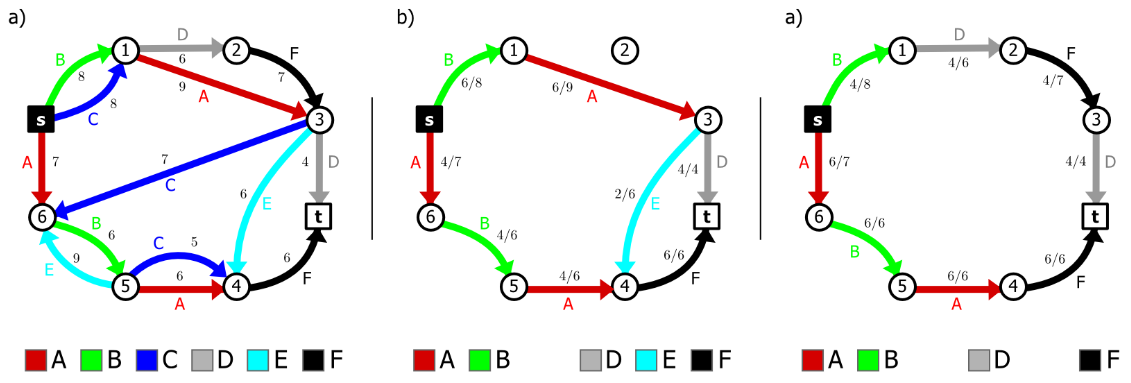


Figure 8.8: maximum flow minimum labeling problem. **(a)** An input ALD with capacities associated with its vertices and a maximum s-t flow $F^* = 10$. Solutions for this input instance with **(b)** $L' = \{A,B,D,E,F\}$ and **(c)** $L' = \{A,B,D,F\}$

The CCut formulation adapted to the MFMLP is presented in the program (8.20) through (8.22), in which F^* is the maximum flow in D, $K^-(S) = \{l^a(a) \mid a \in \delta^-(S)\}$ is the set of labels represented in $\delta^-(S)$, and $\delta^-(S)$ is the set of ingoing arcs in the cut set $[S, V\backslash S]$, for any $S \subset V$. The objective function (8.20) minimizes the number of labels in the solution, and the exponential set of constraints (8.21) ensures that the sum of the capacities of the arcs in any s-t cut is not lesser than F^*. In this case, from the min-cut max-flow theorem, it follows that the maximum flow from s to t is at least F^*.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.20}$$

$$\text{s.t.} \quad \sum_{l \in K^-(S)} \Big(\sum_{\substack{a \in \delta^-(S) \\ l^a(a)=l}} c(a) \Big) \cdot z_l \geq F^*, \qquad \forall S \subset V, s \notin S, t \in S, \tag{8.21}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.22}$$

Considering that sometimes the capacities are too big in comparison to F^*, the formulation can be improved by replacing the inequalities (8.21) by the inequalities (8.23).

$$\sum_{l \in K^-(S)} min\Big(\sum_{\substack{a \in \delta^-(S) \\ l^a(a)=l}} c(a),\ F^* \Big) \cdot z_l \geq F^*, \qquad \forall S \subset V, s \notin S, t \in S. \tag{8.23}$$

Moreover, consider the concept of *necessary label* given by Granata et al. (2013):

Definition 8.3. *A label $l \in L$ is necessary with respect to the MFMLP if the maximum s-t flow in the graph $D[L \backslash \{l\}]$ is strictly less than the maximum s-t flow in D.*

It is easy to see that the label $F \in L$ is necessary in the instance presented in Figure 8.8(a). Let $\mathcal{L} \subseteq L$ be the set of necessary labels for an input ALD. The formulation can be further improved by replacing the inequalities (8.23) by the inequalities (8.24).

$$\sum_{l \in K^-(S) \backslash \mathcal{L}} min\Big(\sum_{\substack{a \in \delta^-(S) \\ l^a(a)=l}} c(a),\ F^* - \sum_{\substack{a \in \delta^-(S) \\ l^a(a)\in\mathcal{L}}} c(a) \Big) \cdot z_l \geq F^* - \sum_{\substack{a \in \delta^-(S) \\ l^a(a)\in\mathcal{L}}} c(a), \quad \forall S \subset V, s \notin S, t \in S. \tag{8.24}$$

Lastly, it could be interesting to consider an extension of the MFMLP in which the desired maximum s-t flow in the solution digraph should be greater than a given value $F^\diamond \leq F^*$. We call this extension the minimum labeling given flow problem (MLGFP).

8.3 Problems with enhanced connectivity

In this section, we address three problems defined on ELGs that look for solution graphs with more connectivity requirements than the MLSTP. This kind of problem arises when the solution graph is wanted to be still connected even if k links, k labels, or a node fails. We show how to adapt the CCut formulation to solve these problems.

$\star$ **The minimum labeling edge-k-connected graph problem:** Given an ELG $G = (V, E, L)$, the aim of the minimum labeling edge-k-connected graph problem (MLEkCGP) is to find a minimum cardinality set of labels $L' \subseteq L$ such that $G[L']$ is edge-k-connected, i.e. it remains connected even if any subset of $k-1$ edges are removed from it. The MLEkCGP is NP-Complete. Indeed, when $k = 1$ this problem is exactly the MLSTP.

The CCut formulation adapted to the MLEkCGP problem is presented in the program (8.25) through (8.27). Let $E(L')$, $L' \subseteq L$, be the set of edges with label $l(e) \in L'$. The objective function (8.25) minimizes the number of labels, and the exponential set of constraints (8.26) ensures the solution graph is edge-k-connected by requiring at least k active edges for every cut in the graph.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.25}$$

$$\text{s.t.} \quad \sum_{l \in K(S)} min\big(k,\ |E(\{l\}) \cap \delta(S)|\big) \cdot z_l \geq k, \qquad \forall S \subset V, S \neq \emptyset, \tag{8.26}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.27}$$

$\star$ **The minimum labeling label-k-connected graph problem:** Given an ELG $G = (V, E, L)$, the aim of the minimum labeling label-k-connected graph problem (MLLkCGP) is to find a minimum cardinality set of labels $L' \subseteq L$ such that $G[L']$ is label-k-connected, i.e. it remains connected even if $k-1$ labels are removed from it. The MLLkCGP is NP-Complete. Indeed, when $k = 1$ this problem is exactly the MLSTP.

The CCut formulation adapted to the MLLkCGP problem is presented in the program (8.28) through (8.30). The objective function (8.28) minimizes the number of labels, and the exponential set of constraints (8.29) ensures the solution graph is label-k-connected by requiring at least k active labels for every cut in the graph.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.28}$$

$$\text{s.t.} \quad \sum_{l \in K(S)} z_l \geq k, \qquad \forall S \subset V, S \neq \emptyset, \tag{8.29}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.30}$$

The minimum labeling vertex-biconnected graph problem (Perez and Consoli, 2015):

Given an ELG $G = (V, E, L)$, the aim of the minimum labeling vertex-biconnected graph problem (MLVBGP) is to find a minimum cardinality set of labels $L' \subseteq L$, such that $G[L']$ is vertex-biconnected, i.e. it is connected even if one vertex is removed from it.

The MLVBGP is NP-Complete. The proof is given by extending the transformation given in Section 2.1. Let $G(U, \mathcal{S}) = (V, E, L)$, $|U| \geq 2$, be the graph built from a set covering instance as given in Section 2.1. Let $G' \leftarrow G /\!\!/ \{k^*\}$, and let v be the vertex originated from v^* in the contraction. Add a new vertex w to G' and, for every edge $e = (v, j) \in E$, add to G' a new edge $y = (w, j)$, such that $l(e) = l(y)$. It is straightforward to verify that G' has a vertex-biconnected spanning subgraph with k labels if and only if a minimum set covering with k sets does exist.

Figure 8.9 illustrates the extension of the graph $G(U, \mathcal{S}) = (V, E, L)$. Fig. 8.9(a) brings the ELG G built from a set covering instance presented in Section 2.1, Fig. 2.1(b). Fig. 8.9(b) depicts the graph $G' = G /\!\!/ \{k^*\}$, and Fig. 8.9(c) shows the graph G' after adding the *clone* vertex w.

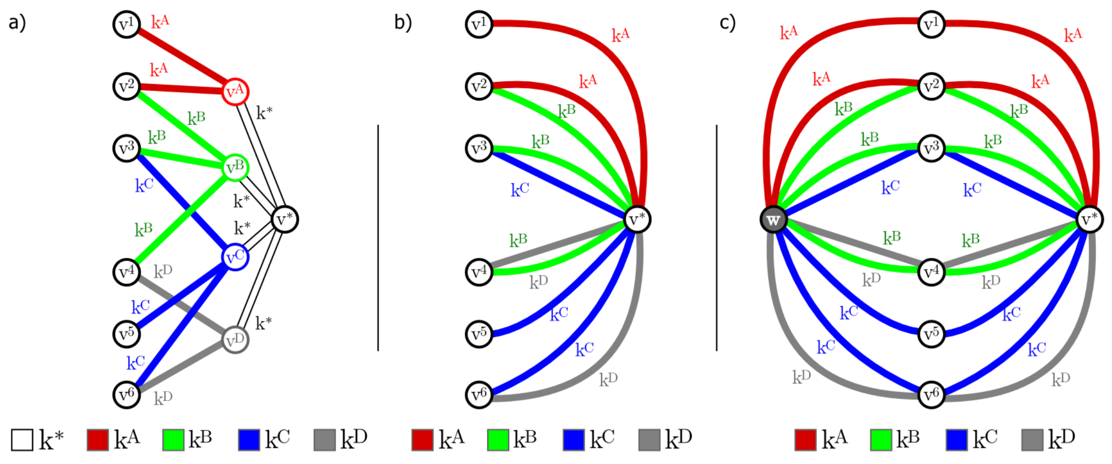


Figure 8.9: The MLVBGP is NP-complete. **(a)** An ELG $G(U, \mathcal{S}) = (V, E, L)$ built from an instance of the set covering problem. **(b)** The graph $G' = G /\!\!/ \{k^*\}$. **(c)** The graph G' after adding the *clone* vertex w, depicted in gray

Let $G[[S]]$ be the subgraph induced by the set of vertices $S \subseteq V$ and, for convenience, let $K(S, G)$ be the colorful cut derived from the set of vertices S on the graph $G = (V, E, L)$, where $S \subset V$, $S \neq \emptyset$. Figure 8.10 illustrates these concepts. Fig. 8.10(a) presents the input ELG $G = (V, E, L)$, and the Figures 8.10(b) to (f) show the graphs $G[[V \backslash \{v\}]]$, for each $v \in V$. Moreover, considering $S = \{1\}$, the colorful cut $K(S, G[[V \backslash \{2\}]])$ is $\{C, D\}$ (Fig. 8.10(c)), while $K(S, G[[V \backslash \{3\}]])$ is $\{A, C\}$ (Fig. 8.10(d)).

The CCut formulation adapted to the MLVBGP problem is presented in the program (8.31) through (8.33). It is derived directly from the definition of the problem, using CCut to

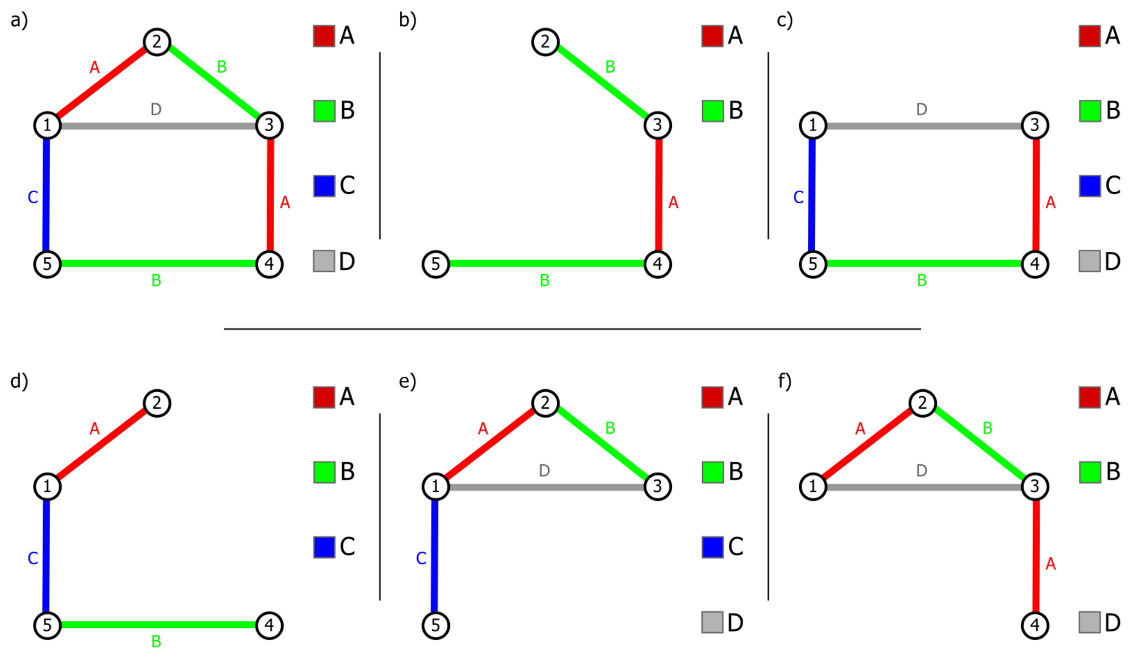


Figure 8.10: Illustration of the concepts of subgraph induced by a set of vertices and $K(S,G)$, the colorful cut derived from the set of vertices S on the graph G. **(a)** An input ELG $G = (V,E,L)$. **(b)** to **(f)** The graphs $G[[V\backslash\{v\}]]$, for each $v \in V$

ensure that the graph $G[[V\backslash\{v\}]]$ is connected $\forall v \in V$, considering the subgraph induced by the chosen colors. The objective function (8.31) minimizes the number of labels, and the exponential set of constraints (8.32) ensures the solution graph is vertex biconnected by requiring that every cut in the solution graph has an active label even if a vertex is removed.

$$\text{Minimize} \sum_{l \in L} z_l \tag{8.31}$$

$$\text{s.t.} \quad \sum_{l \in K(S^v, G^v)} z_l \geq 1, \quad \begin{cases} \forall v \in V \mid G^v = G[[V\backslash\{v\}]] = (V^v, E^v, L^v), \\ \forall S^v \subset V^v, S^v \neq \emptyset, \end{cases} \tag{8.32}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{8.33}$$

Indeed, this formulation can be further improved by replacing the set of inequalities (8.32) by the set (8.34). First, let $N(S) = \{v \in V\backslash S \mid e = (v,x) \in \delta(S) \ or \ e = (x,v) \in \delta(S)\}$, for any $S \subseteq V$, be the set of neighbor vertices of S. Considering the graph $G = (V,E,L)$ presented in Fig. 8.10(a) and $S = \{5\}$, we have that $N(S) = \{1,4\}$, $K(S,G[[V\backslash\{1\}]]) = \{B\}$, $K(S,G[[V\backslash\{2\}]]) = K(S,G[[V\backslash\{3\}]]) = \{B,C\}$, and $K(S,G[[V\backslash\{4\}]]) = \{C\}$. Observe that if $v \notin N(S)$, and $v \notin S$, then $K(S,G[[V\backslash\{v\}]]) = K(S,G)$. Further, note that if $v \in N(S)$, then $K(S,G[[V\backslash\{v\}]]) \subseteq K(S,G)$, and the colorful cut inequality associated with $K(S,G)$ is dominated (or redundant) by the one associated with $K(S, G[[V\backslash\{v\}]])$. Thence, to ensure the solution graph is biconnected, it is only necessary to satisfy the colorful cuts $K(S, G[[V\backslash\{v\}]])$, for all $S \subset V$, $S \neq \emptyset$, $|S| \leq |V|-2$, and for all $v \in N(S)$. This is the set of constraints (8.34).

$$\sum_{l \in K(S,G^v)} z_l \geq 1, \qquad \begin{cases} \forall S \subset V, S \neq \emptyset, |S| \leq |V|-2, \\ \forall v \in N(S), G^v = G[[V\backslash\{v\}]] = (V^v, E^v, L^v). \end{cases} \tag{8.34}$$

Lastly, the Figure 8.11 illustrates the three enhanced connectivity problems discussed in this section. The graph in Fig. 8.11(a) is edge-2-connected, satisfying the MLEkCGP for $k = 2$. However, the removal of either the label D or the vertex 3 disconnects it. The graph in Fig. 8.11(b) is label-2-connected, satisfying the MLLkCGP for $k = 2$. Again, the removal of the vertex 3 disconnects it. The graph in Fig. 8.11(c) is vertex biconnected, satisfying the MLVBGP.

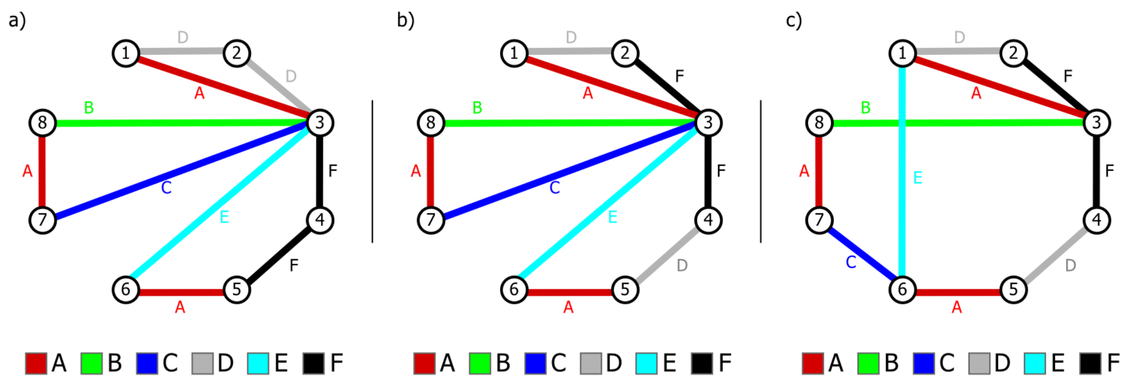


Figure 8.11: Illustration of enhanced connectivity problems defined on ELGs. **(a)** An ELG that is edge-2-connected, satisfying the MLEkCGP for $k = 2$. **(b)** An ELG that is label-2-connected, satisfying the MLLkCGP for $k = 2$. **(c)** An ELG that is vertex biconnected, satisfying the MLVBGP

Chapter 9

The Minimum Labeling Global Cut Problem

This chapter addresses the minimum labeling global cut problem (MLGCP). Given an ELG, the problem consists in finding a minimum cardinality set of labels whose removal disconnects the input graph. This problem is closely related to both MLSTP and GMLSTP, since it arises as the separation problem for the colorful cuts inequalities (refer to Section 4.1). Figure 9.1 illustrates the MLGCP. Fig. 9.1(a) brings the input ELG. Fig. 9.1(b) highlights a cut with the labels E and F, while the Fig. 9.1(c) shows the input graph is disconnected without these labels.

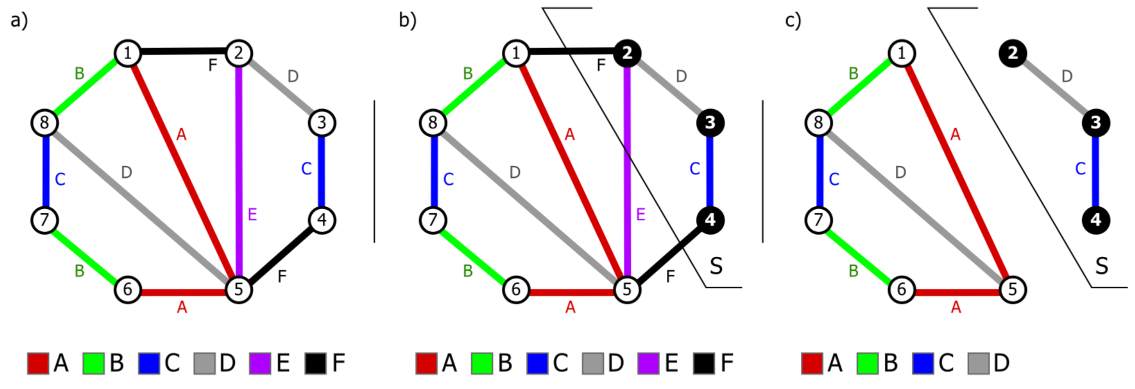


Figure 9.1: Illustration of the MLGCP. **(a)** The input ELG $G = (V,E,L)$. **(b)** The colorful cut $K(\{2,3,4\}) = \{E,F\}$. **(c)** The graph $G[\{A,B,C,D\}]$, which is disconnected

Formally, given an ELG $G = (V,E,L)$, the minimum labeling global cut problem aims to find a subset of labels $L' \subseteq L$, such that $G[L \backslash L']$ is not connected and $|L'|$ is minimized. The computational complexity of the MLGCP still remains as a theoretical open question. Despite of that, the problem of finding an *s*-*t* cut with the minimum number of colors — namely, the minimum labeling s-t cut problem (MLstCP) — is NP-hard (Jha et al., 2002; Coudert et al., 2007).

Coudert et al. (2007) and Coudert et al. (2014) address both the MLGCP and the MLstCP with the goal of assessing the capacity of a network to remain connected when links are at shared risk. For instance, considering an wireless network, an attacker can shut down all of the links at a certain frequency by creating a strong jam signal. Another example is when two cables share the same duct in part of their way.

Several approximation algorithms have been proposed for the MLstCP (Zhang et al., 2011; Tang and Zhang, 2012; Zhang, 2014). The best results were obtained by Zhang (2014), who proved the problem is $l_{\max}-$approximated and $f_{\max}-$approximated, where $l_{\max}$ is the size of the largest $s-t$ path and $f_{\max}$ is an upper bound on the number of edges of a label. Lastly, Zhang (2014) has demonstrated that the MLGCP can be solved polynomially if the input ELG is planar, has a small value of $f_{\max}$, or has a bounded *treewidth*.

In the following sections, we propose three new mathematical formulations for the MLGCP and branch-and-cut algorithms to solve them. The computational experiments showed that the proposed methods are able to solve small and average sized instances in a reasonable amount of time.

9.1 Partition based formulations

The first model we propose, denominated PART, aims to partition the set of vertices of the input ELG into two sets S and $\overline{S} = V \backslash S$. For any set $S \subset V$, $S \neq \emptyset$, the removal of all edges that have one endpoint in S and the other in $\overline{S}$ disconnects the graph. Let z_l, $\forall l \in L$, be a variable that is set to 1 if and only if the label l is part of the solution cut. Let x_e, $\forall e \in E$, be a variable that is set to 1 if and only if the edge e is part of the solution cut. And let w_v, $\forall v \in V$ be a variable that is set to 1 if the vertex v belongs to the set S and to 0 otherwise. The model PART is described in the program (9.1) to (9.8).

$$\text{PART} = \text{Minimize} \sum_{l \in L} z_l \tag{9.1}$$

$$\text{s.t.} \quad 1 \leq \sum_{v \in V} w_v \leq |V| - 1, \tag{9.2}$$

$$z_{l(e)} \geq x_e, \qquad \forall e \in E, \tag{9.3}$$

$$x_e \geq w_i - w_j, \qquad \forall e = (i,j) \in E, \tag{9.4}$$

$$x_e \geq w_j - w_i, \qquad \forall e = (i,j) \in E, \tag{9.5}$$

$$z_l \geq 0, \qquad \forall l \in L, \tag{9.6}$$

$$x_e \geq 0, \qquad \forall e \in E, \tag{9.7}$$

$$w_v \in \{0,1\}, \qquad \forall v \in V. \tag{9.8}$$

The objective function (9.1) minimizes the number of labels that are necessary to disconnect the input ELG. The constraint (9.2) avoid connected solutions. The set of inequalities (9.3) binds the variables of labels with the edge ones. The constraints (9.4) and (9.5) activate the edges of the cut, while the constraints (9.6) to (9.8) define the domain of the decision variables.

Observe that combining the constraints (9.3), (9.4) and (9.5), we have $z_e \geq x_e \geq |w_i - w_j|$, which is equivalent to $z_e \geq |w_i - w_j|$. Given that, it is possible to eliminate the edge variables from the formulation. Moreover, we can eliminate the symmetry of the solutions by arbitrarily choosing a vertex to be in $\overline{S}$. The reformulated model, denominated PART$_2$, is presented in the program (9.9) to (9.15).

$$\text{PART}_2 = \text{Minimize} \sum_{l \in L} z_l \tag{9.9}$$

$$\text{s.t.} \quad \sum_{v \in V} w_v \geq 1, \tag{9.10}$$

$$w_1 = 0, \tag{9.11}$$

$$z_{l(e)} \geq w_i - w_j, \qquad \forall e = (i,j) \in E, \tag{9.12}$$

$$z_{l(e)} \geq w_j - w_i, \qquad \forall e = (i,j) \in E, \tag{9.13}$$

$$z_l \geq 0, \qquad \forall l \in L, \tag{9.14}$$

$$w_v \in \{0,1\}, \qquad \forall v \in V. \tag{9.15}$$

9.2 A clustering based formulation

The second formulation that we propose for the MLGCP is based on the cluster editing problem (CEP). The CEP consist in: given an undirected graph $G = (V,E)$, transform G into a graph that consists of a disjoint union of cliques by inserting and/or deleting a minimum number of edges. To solve the MLGCP in the way the CEP is solved, we consider the addition of an edge with cost 0, while the cost of the removal of $e \in E$ is modeled as the cost of removing the label $l(e) \in L$ from G.

Consider $G = (V,E,L)$ the input ELG for the MLGCP. Let x_{ij}, $\forall i, j \in V$, $i \neq j$, be a binary decision variable that is set to 1 if the edge $e = (i, j)$ is part of the solution, and to 0 otherwise. Further, let z_l, $\forall l \in L$, be a continuous variable that is set to 1 if and only if the label l is part of the solution cut. The formulation P3E is presented in the program (9.16)-(9.23). The objective function (9.16) minimizes the number of labels in the solution cut. The constraints (9.17) bind the edge with the label variables. The set of inequalities (9.18), (9.19), and (9.20) is the classic P$_3$ (paths with three vertices) elimination constraints, as discussed in the sequel. The inequality (9.21) ensures the solution is not empty. Finally, the expressions (9.22) to (9.23) define the domain of the decision variables.

$$\text{P3E} = \text{Minimize} \sum_{l \in L} z_l \tag{9.16}$$

$$\text{s.t.} \quad z_{l(e)} \geq x_{ij}, \qquad \forall e = (i,j) \in E, \tag{9.17}$$

$$+x_{ij} + x_{jk} - x_{ki} \geq 0, \qquad \forall i,j,k \in V, i \neq j, j \neq k, k \neq i, \tag{9.18}$$

$$+x_{ij} - x_{jk} + x_{ki} \geq 0, \qquad \forall i,j,k \in V, i \neq j, j \neq k, k \neq i, \tag{9.19}$$

$$-x_{ij} + x_{jk} + x_{ki} \geq 0, \qquad \forall i,j,k \in V, i \neq j, j \neq k, k \neq i, \tag{9.20}$$

$$\sum_{e=(i,j) \in E} x_{ij} \geq 1, \tag{9.21}$$

$$z_l \geq 0, \qquad \forall l \in L, \tag{9.22}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i,j \in V, i \neq j. \tag{9.23}$$

According to Grötschel and Wakabayashi (1990), G is a disjoint union of cliques if and only if G does not have any induced path with three vertices, for short P$_3$, as its subgraph.

Figure 9.2 illustrates the configurations allowed and forbidden to G become a disjoint union of cliques for any $u, v, w \in V$. Fig. 9.2(a) presents the allowed configurations, while the Fig. 9.2(b) presents the forbidden ones, namely, the P_3 subgraphs.

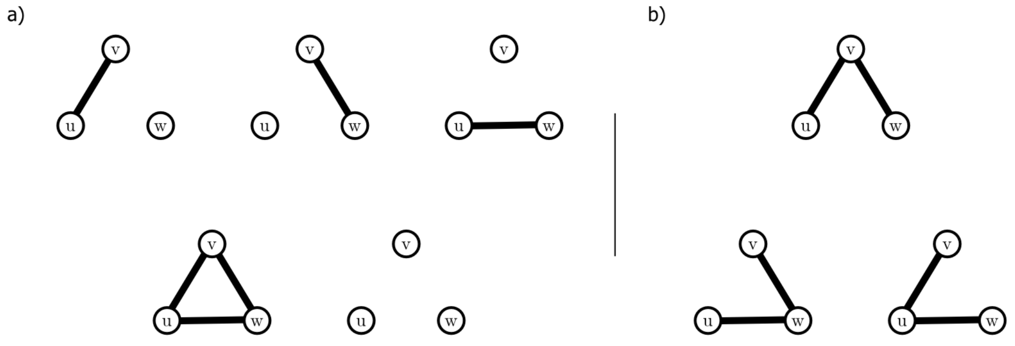


Figure 9.2: Allowed and forbidden configurations according to the P_3 elimination inequalities. **(a)** The allowed and **(b)** the forbidden configurations

Given the size of the set of P_3 elimination inequalities, we propose to solve the P3E model by branch-and-cut. The proposed algorithm starts without any inequality for elimination of P_3 and, for every integer solution found, it checks by enumeration if any constraint of this set has been violated.

9.3 A tree elimination based formulation

The third formulation we propose for the MLGCP is based on the fact that a spanning tree is a minimal connected graph with respect to its number of edges, and, as a consequence, if a graph does not have any spanning tree it is disconnected. Let $\mathcal{T}$ be the set of all spanning trees of G, such that $L^t(T)$, $T \in \mathcal{T}$, represents the set of labels of the edges of T; and let z_l, $\forall l \in L$, be a variable that is set to 1 if and only if the label l is part of the solution cut; the program (9.24) to (9.26) presents the tree elimination formulation (TEF).

$$\text{TEF} = \text{Minimize} \sum_{l \in L} z_l \tag{9.24}$$

$$\text{s.t.} \quad \sum_{l \in L(T)} z_l \geq 1, \qquad \forall T \in \mathcal{T}, \tag{9.25}$$

$$z_l \in \{0,1\}, \qquad \forall l \in L. \tag{9.26}$$

The objective function (9.24) minimizes the number of labels of the solution. The exponential set of tree elimination constraints (9.25) ensures that the solution graph is not connected by prohibiting any spanning tree of G. The constraints (9.26) define the domain of the decision variables z.

Figure 9.3 illustrates the tree elimination constraints. Fig. 9.3(a) brings an input ELG $G = (V,E,L)$, while Fig. 9.3(b) shows a spanning tree T of G that uses the set of labels $\{A,C,D,E\}$. Since T has to be eliminated for G to be disconnected, the inequality

$$z_A + z_C + z_D + z_E \geq 1$$

ensures that at least one label of T is part of the solution.

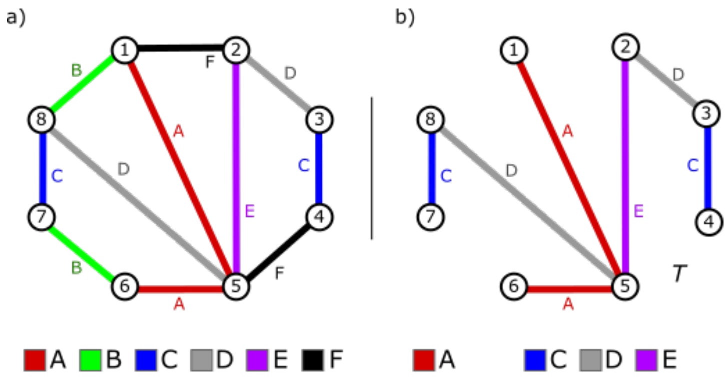


Figure 9.3: Illustration of spanning tree on an ELG. **(a)** An input ELG $G = (V,E,L)$. **(b)** A tree of G with the set of labels $\{A,C,D,E\}$

Since the number of spanning trees on a graph with n nodes is $O(n^{n-2})$, it is not practical to add all the set of restrictions (9.25) at once to the model. Instead, we propose a branch-and-cut algorithm. It starts the model without any tree elimination constraint and, for any integer solution found, if it is possible to build a spanning tree with the remaining labels, the associated tree elimination inequality (9.25) is added to the model. We used the MVCA heuristic (refer to Section 2.2) to find spanning trees which use the minimum number of labels.

Furthermore, we propose a separation heuristic for the tree eliminations inequalities (9.25). Let z_l^* be the value of the variable z_l in the linear relaxation of the TEF model, and let H be an empty ELG with the same set of vertices of the input graph. Select the label l with lower z_l^* and add all of it edges to H. Repeat this procedure until H is connected. If the sum of the values z_l^* of the selected labels is lesser than 1, then this cut is violated.

9.4 Computational experiments

In this section we describe the computational experiments performed in order to assess the performance of the mathematical formulations proposed for the MLGCP. All the methods were implemented in C++ language and compiled by using g++ 4.6.3, with the optimization flag -O3. All formulations and their derived procedures were implemented using the Concert library and Cplex 12.4 as the solver. The experiments were performed on a computer with Intel(R) Core(TM) i7, 64 bits, CPU, 2.93GHz, 8 GB of RAM, and Ubuntu 14.04 as the operating system. Although the processor of this device has more than one core, the algorithms were executed by using a single core and a single thread within a time limit of one hour.

We have considered the group 2 of ELGs generated by Cerulli et al. (2005) for the first set of computations. This group of input graphs has instances with number of vertices $n = |V| \in \{50, 100, 200\}$, number of labels $l = |L| \in \{0.25n, 0.5n, n, 1.25n\}$, and edge densities $d \in \{0.2, 0.5, 0.8\}$. Also, each dataset consists in 10 different graphs for one *n-l-d* configuration.

Tables 9.1 to 9.3 present the results of the first group of experiments performed. Each line represents a dataset with 10 different graphs for one *n-l-d* configuration. The first group of columns identifies the dataset. The column *UB* brings the average value of the best integer solutions found (considering all the methods). The remaining columns are divided into three groups: $PART_2$, *P3E* and *TEF*, one for each formulation. The column *O* indicates the number of optimal solutions found by the method. The column *t(s)* indicates the average running time in seconds. The column *gap* represents (as a percentage) the average of the differences between the UBs and LBs achieved by the method. Finally, the column *gapr* reports (as a percentage) the average of the differences between the UBs and the linear relaxations[1].

We can observe from Table 9.1 that the models $PART_2$ and *TEF* are able to solve to optimality all the instances of this group. Although the model *P3E* has failed in only one instance, its computational times are too high in comparison with the other models. *TEF* has obtained the best linear relaxation for all datasets and the best computational times for instances with small values of d and l.

The results for the method *P3E* are omitted from the next tables since it was not able to find any integer solution within one hour. Table 9.2 shows that *TEF* still has the best linear relaxation for all instances. For instances with $l = 25$, the *TEF* has the best computational times. While $PART_2$ was able to solve all of the instances on this group, *TEF* has failed for 32 graphs. Then, it is possible to notice a degradation on the performance of the *TEF* as the values of l and

[1]The values for the linear relaxations were obtained by setting Cplex to *NodeLimit(1)*.

d grows.

Table 9.1: Results of the computational experiments for instances with $|V| = 50$

Instance			***P3E***				***PART$_2$***				***TEF***			
$\|L\|$	***d***	***UB***	***O***	***t (s)***	***gap***	***gapr***	***O***	***t (s)***	***gap***	***gapr***	***O***	***t (s)***	***gap***	***gapr***
	ld	2,5	10	843,4	0	67,4	10	0,05	0	56,0	10	0,006	0	6
12	md	7,4	10	947,0	0	90,4	10	0,08	0	77,8	10	0,001	0	22,0
	hd	9,8	10	596,7	0	69,9	10	0,05	0	35,3	10	0,001	0	24,2
	ld	2,7	10	1015,9	0	74,6	10	0,183	0	38,1	10	0,009	0	8,6
25	md	9,9	10	997,5	0	93,4	10	0,31	0	81,7	10	0,04	0	38,9
	hd	15,5	10	1511,2	0	96,1	10	0,31	0	87,9	10	0,05	0	31,7
	ld	2,8	9	1399,9	8,9	73,6	10	0,17	0	37,5	10	0,04	0	13,7
50	md	11,6	10	805,2	0	94,8	10	0,82	0	85,1	10	0,85	0	40,3
	hd	21,3	10	1280,9	0	97,2	10	1,48	0	90,8	10	3,1	0	45,9
	ld	2,8	10	1124,9	0	73,8	10	0,13	0	38,0	10	0,05	0	11,1
62	md	12,1	10	715,0	0	95,1	10	1,1	0	85,8	10	4,4	0	45,5
	hd	22,7	10	809,4	0	97,7	10	1,8	0	91,0	10	12,7	0	58,6

Table 9.2: Results of the computational experiments for instances with $|V| = 100$

Instance			***PART$_2$***				***TEF***			
$\|L\|$	***d***	***UB***	***O***	***t (s)***	***gap***	***gapr***	***O***	***t (s)***	***gap***	***gapr***
	ld	6,2	10	0,9	0	83,2	10	0,06	0	37,9
25	md	16,5	10	0,7	0	91,1	10	0,09	0	35,6
	hd	21	10	0,5	0	86,3	10	0,1	0	39,2
	ld	6,8	10	2,5	0	83,7	10	0,4	0	37,7
50	md	22,2	10	5,7	0	93,5	10	6,5	0	55,1
	hd	33,1	10	4,8	0	95,8	10	6,5	0	48,3
	ld	7,2	10	2,1	0	84,3	10	10,9	0	41,5
100	md	26,5	10	9,3	0	95,3	5	2012,7	8,4	63,9
	hd	45,2	10	22,4	0	96,4	2	3071,5	10,3	61,8
	ld	7,2	10	3,1	0	84,1	10	75,8	0	38,0
125	md	27,1	10	36,1	0	95,0	1	3409,0	17,8	55,7
	hd	48,6	10	43,3	0	96,7	0	3600,0	15,0	90,1

Table 9.3 shows that *TEF* still has the best linear relaxation for all of the instances. However, it failed to solve three groups of instances (marked with a $*$). It was not able to yield feasible integer solutions within one hour of execution. Even with a worse linear relaxation, $PART_2$ has solved 101 instances out of 120.

Table 9.3: Results of the computational experiments for instances with $|V| = 200$

Instance			***$PART_2$***				***TEF***			
$\|L\|$	***d***	***UB***	***O***	***t (s)***	***gap***	***gapr***	***O***	***t (s)***	***gap***	***gapr***
	ld	13,2	10	34,4	0	93,7	10	5,7	0	57,4
50	md	32,7	10	9,7	0	97,1	10	9,4	0	48,5
	hd	43,3	10	6,9	0	97,5	10	2,5	0	43,4
	ld	15	10	188,3	0	94,7	9	1219,8	3,1	55,0
100	md	45,4	10	699,5	0	98,3	0	3600	15,5	62,9
	hd	68,8	10	238,9	0	98,7	0	3600	9,4	62,8
	ld	15,9	10	614,4	0	94,0	0	3600	45,4	61,6
200	md	54,1	6	2066,0	11,1	98,2	0	3600	32,3	56,9
	hd	93,8	7	2051,9	14,3	98,9	0	3600	$*$	$*$
	ld	16,1	10	691,6	0	93,9	0	3600	42,7	56,4
250	md	56,5	3	2990,2	35,6	98,1	0	3600	$*$	$*$
	hd	99,4	5	2550,3	26,7	96,9	0	3600	$*$	$*$

We have observed from the results of the first set of experiments that the optimal solution for most of the instances considered is trivial in the sense that it was obtained by disconnecting exactly one vertex from the remaining of the graph. One question arises: how far are the optimal solutions from the trivial ones? Consider the *labeled hypercube* (LH) family of graphs built as follows: Let $LH^0 = (\{v\}, \emptyset, \emptyset)$ be the LH graph of dimension 0. To build the LH^n you take two graphs LH^{n-1} and connect each vertex to its correspondent by using an edge with a new label L^n. Following this construction, we have that removing one label from any LH^n graph, $n > 0$, disconnects it, while that all the trivial solutions have cardinality n. Follows that the difference between the value of the optimal and the trivial solutions can be arbitrarily big. Figure 9.4 presents the first four LH graphs.

Therefore, we generated a new group of ELGs and carried out a second set of computational experiments. The graphs of this group were generated in such a way that a non-trivial solution (most likely the optimal one) is known in advance. Their dimensions range from n-l-d = 100-20-0.2 to 200-400-0.8.

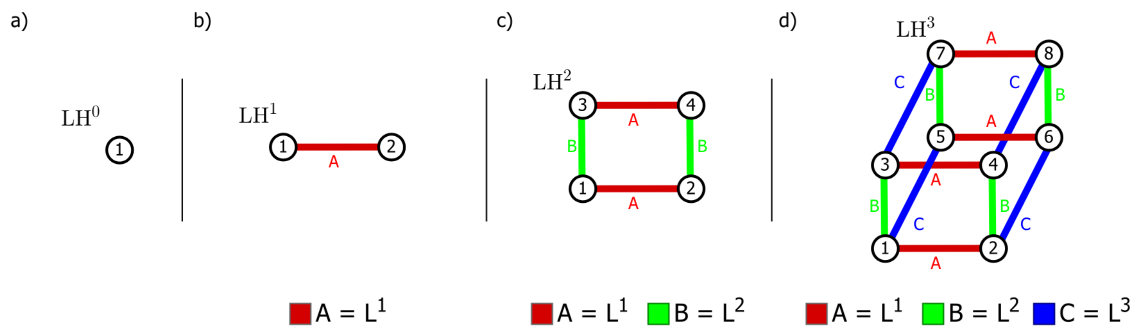


Figure 9.4: Illustration of the construction of the the labeled hypercube graphs. The graphs **(a)** LH^0, **(b)** LH^1, **(c)** LH^2, and **(d)** LH^3

Tables 9.4 and 9.5 present the results of this second set of experiments. Each line of these tables refer to a single input graph. The first column brings the dimension of the instance, where D stands for 100 times the density of the graph. The columns T and P represent, respectively, the costs of the best trivial solution and of the previously known non-trivial one. The meaning of the following columns are similar to the ones in Tables 9.1 to 9.3. Values in bold mean the method found a solution as good as P.

From Table 9.4 we have that PART$_2$ was able to solve all instances with 100 vertices within one hour. In contrast, TEF did not obtain the optimal solution in 9 of 30 instances, not even being able to yield any feasible solution in 3 cases. Moreover, it is possible to notice a degradation on the linear relaxations obtained by TEF in relation to the ones of the first experiment. From Table 9.5 we have that PART$_2$ was able to solve 25 of 30 of the instances with 200 vertices. As expected, the running times of this model grow as the number of labels of the input graph increases. The TEF only achieved the optimal solution for 5 instances, while it did not find any feasible solutions for another 5 instances. For the instances 200.100.80, 200.133.80, and 200.133.50 the TEF obtained the optimal solution in much less time than PART$_2$.

Lastly, the computational experiments performed showed that the proposed methods are able to solve small to medium instances to optimality within one hour. The method *P3E* presented a bad performance due to its large number of variables and inequalities. In its turn, even having a poor linear relaxation, the *PART*$_2$ has achieved the best performance. This performance is due to the linear number of inequalities of the model (with respect to the edges of the graph) and to the *branch-and-bound* on the variables w, which have a linear dimension n. The model *TEF* achieved the best linear relaxations, but presented serious convergence problems.

Table 9.4: Results of the second experiment for instances with $|V| = 100$

Instance			PART_2				TEF			
n.l.D	**T**	**P**	**UB**	**t(s)**	**gap**	**gapr**	**UB**	**t(s)**	**gap**	**gapr**
100.50.20	15	13	**13**	3,0	0,0%	77,3%	**13**	5,0	0,0%	76,9%
100.67.20	16	15	**15**	2,4	0,0%	85,7%	**15**	80,1	0,0%	86,7%
100.83.20	16	14	**14**	4,3	0,0%	78,9%	**14**	25,1	0,0%	78,6%
100.100.20	16	14	**14**	2,4	0,0%	84,6%	**14**	198,6	0,0%	85,7%
100.117.20	16	14	**14**	2,8	0,0%	84,6%	**14**	351,4	0,0%	78,6%
100.133.20	15	14	**14**	13,4	0,0%	81,1%	**14**	590,5	0,0%	82,1%
100.150.20	16	15	**15**	11,2	0,0%	85,1%	**15**	361,2	0,0%	80,0%
100.164.20	14	13	**13**	7,7	0,0%	81,5%	14	3600,0	9,8%	85,7%
100.183.20	14	13	**13**	6,6	0,0%	79,9%	**13**	1234,3	0,0%	84,6%
100.200.20	16	14	**14**	28,0	0,0%	81,1%	17	3600,0	27,1%	88,2%
100.50.50	24	23	**23**	6,1	0,0%	88,1%	**23**	0,2	0,0%	78,3%
100.67.50	34	30	**30**	7,1	0,0%	90,1%	**30**	27,6	0,0%	83,3%
100.83.50	33	32	**32**	10,1	0,0%	90,9%	**32**	201,9	0,0%	81,3%
100.100.50	36	35	**35**	10,0	0,0%	91,3%	**35**	2033,8	0,0%	85,7%
100.117.50	38	35	**35**	6,3	0,0%	91,4%	$*$	3600,0	$*$	$*$
100.133.50	40	37	**37**	18,1	0,0%	92,2%	$*$	3600,0	$*$	$*$
100.150.50	40	38	**38**	7,5	0,0%	91,8%	41	3600,0	41,5%	90,2%
100.164.50	40	37	**37**	163,9	0,0%	91,6%	$*$	3600,0	$*$	$*$
100.183.50	41	35	**35**	58,8	0,0%	91,0%	41	3600,0	46,3%	91,9%
100.200.50	41	39	**39**	14,6	0,0%	92,2%	42	3600,0	51,2%	92,9%
100.50.80	35	31	**31**	2,9	0,0%	90,1%	**31**	0,1	0,0%	87,1%
100.67.80	42	36	**36**	8,9	0,0%	91,1%	**36**	1,0	0,0%	66,7%
100.83.80	41	38	**38**	7,3	0,0%	91,7%	**38**	2,9	0,0%	85,5%
100.100.80	44	37	**37**	11,8	0,0%	91,7%	**37**	10,1	0,0%	83,8%
100.117.80	46	42	**42**	13,2	0,0%	92,3%	**42**	146,5	0,0%	84,5%
100.133.80	48	47	**47**	84,8	0,0%	93,0%	**47**	917,7	0,0%	87,9%
100.150.80	42	38	**38**	17,5	0,0%	90,4%	**38**	120,8	0,0%	88,2%
100.164.80	39	36	**36**	9,2	0,0%	90,6%	**36**	183,0	0,0%	86,1%
100.183.80	42	40	**40**	19,7	0,0%	91,4%	**40**	1028,3	0,0%	90,0%
100.200.80	43	42	**42**	15,8	0,0%	91,6%	43	3600,0	14,0%	89,9%

Table 9.5: Results of the second experiment for instances with $|V| = 200$

Instance			**PART**.2				**TEF**			
n.l.D	**T**	**P**	**UB**	**t(s)**	**gap**	**gapr**	**UB**	**t(s)**	**gap**	**gapr**
200.100.20	33	29	**29**	147,73	0,0%	91,3%	31	3600,00	27,5%	89,2%
200.133.20	33	29	**29**	127,55	0,0%	90,3%	*	3600,00	*	*
200.167.20	31	25	**25**	125,71	0,0%	89,7%	*	3600,00	*	*
200.200.20	32	30	**30**	297,66	0,0%	91,5%	33	3600,00	56,6%	92,4%
200.233.20	32	27	**27**	181,01	0,0%	91,4%	34	3600,00	63,2%	93,1%
200.267.20	32	30	**30**	127,42	0,0%	90,6%	33	3600,00	62,1%	90,9%
200.300.20	34	33	**33**	170,04	0,0%	92,0%	35	3600,00	66,7%	92,9%
200.333.20	33	31	**31**	387,29	0,0%	92,4%	34	3600,00	72,5%	92,6%
200.367.20	32	29	**29**	117,65	0,0%	92,6%	35	3600,00	73,3%	91,4%
200.400.20	32	26	**26**	544,64	0,0%	89,9%	36	3600,00	75,0%	94,4%
200.100.50	59	51	**51**	99,09	0,0%	94,8%	**51**	751,05	0,0%	90,2%
200.133.50	44	43	**43**	811,47	0,0%	93,5%	**43**	23,68	0,0%	93,0%
200.167.50	63	52	**52**	952,59	0,0%	94,4%	71	3600,00	54,9%	93,7%
200.200.50	77	71	**71**	99,07	0,0%	95,9%	78	3600,00	61,5%	92,3%
200.233.50	68	64	**64**	108,29	0,0%	95,2%	83	3600,00	69,5%	94,6%
200.267.50	66	63	**63**	1852,27	0,0%	95,2%	*	3600,00	*	*
200.300.50	78	73	**73**	1640,15	0,0%	95,9%	81	3600,00	70,2%	94,1%
200.333.50	75	68	**68**	1416,81	0,0%	95,5%	84	3600,00	77,3%	95,8%
200.367.50	83	75	**75**	3600,00	37,3%	95,8%	84	3600,00	72,9%	95,2%
200.400.50	75	68	**68**	3027,69	0,0%	95,5%	*	3600,00	*	*
200.100.80	82	69	**69**	211,98	0,0%	95,8%	**69**	3,55	0,0%	82,6%
200.133.80	81	75	**75**	84,57	0,0%	96,1%	**75**	10,23	0,0%	86,7%
200.166.80	96	86	**86**	81,65	0,0%	96,6%	88	3600,00	6,0%	90,9%
200.200.80	99	85	**85**	70,87	0,0%	96,4%	86	3600,00	16,4%	93,0%
200.233.80	88	83	**83**	105,88	0,0%	96,4%	**83**	3600,00	16,3%	91,6%
200.267.80	95	89	**89**	95,03	0,0%	96,5%	100	3600,00	52,0%	93,0%
200.300.80	102	95	**95**	3600,00	19,7%	96,8%	135	3600,00	73,3%	94,8%
200.333.80	103	93	**93**	3600,00	62,8%	96,6%	*	3600,00	*	*
200.367.80	108	102	**102**	3600,00	60,2%	96,8%	136	3600,00	74,1%	95,6%
200.400.80	98	92	**92**	3600,00	59,1%	96,4%	144	3600,00	79,0%	95,8%

Chapter 10

The Minimum Representation Spanning Tree Problem

In this chapter we introduce the minimum representation spanning tree problem (MRSTP), a new connectivity problem defined on ELGs. In contrast to the MLSTP, which searches for the overall homogeneity of the network, the focus of the MRSTP is the homogeneity of each vertex in the graph. One application of the MRSTP occurs when it is necessary to make investments on the nodes of a network, for instance for buying software licenses, or hardware to enable the use of new protocols. The problem also has applications in areas such as transport projects, network design, telecommunication systems, and distribution of energy. Follows the formal definition of the MRSTP.

Definition 10.1. *Given an ELG $G = (V,E,L)$, the representation set $r(v)$ of a vertex $v \in V$ is the set of labels that have at least one edge incident to v. Formally, $r(v) = K(\{v\})$, where K stands for the colorful cut defined by $\{v\}$ (see Definition 4.2).*

Definition 10.2. *Given an ELG $G = (V,E,L)$, the minimum representation spanning tree problem (MRSTP) aims to find a spanning tree $T = (V',E',L')$ of G such that the function $R(T) = \sum_{v \in V'} |r(v)|$ is minimized.*

Remark that, since T is a tree, $r(v) \geq 1$, $\forall v \in V'$. Thence, it is convenient to use the *representation function* $Rt(G) = \sum_{v \in V} (|r(v)| - 1)$ as the objective function for the MRSTP.

Figure 10.1(a) illustrates the MRSTP. Fig. 10.1(a) introduces the input ELG $G = (V,E,L)$. Fig. 10.1(b) shows a solution with $|L| = 3$ and $Rt = 4$, while Fig. 10.1(c) presents another solution with $|L| = 4$ e $Rt = 3$. Observe that the best solution for the MLSTP is not necessarily the best one for the MRSTP.

We prove the MRSTP is NP-Complete by using the same steps that Chang and Leu

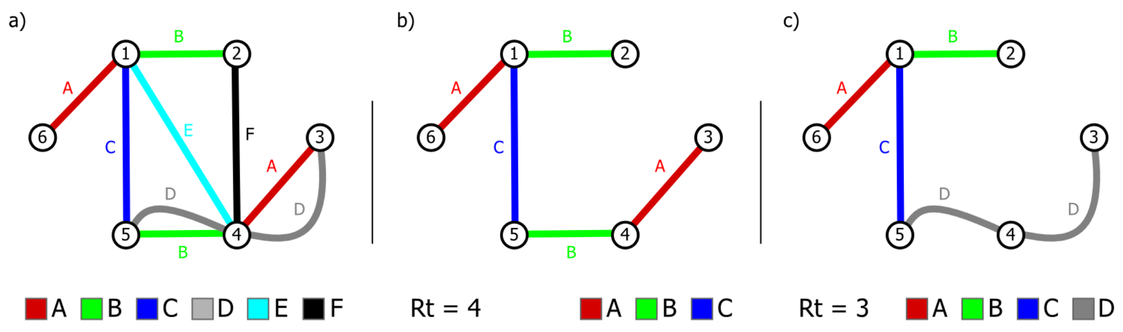


Figure 10.1: Illustration of the minimum representation spanning tree problem. **(a)** The input ELG $G = (V,E,L)$. **(b)** A solution for the MRSTP with $|L| = 3$ and $Rt = 4$. **(c)** A solution for the MRSTP with $|L| = 4$ and $Rt = 3$

(1997) used to prove the MLSTP is NP-Complete. First we define a decision version of the MRSTP, the bounded minimum representation spanning tree problem (B-MRSTP). In the sequel, we show the B-MRSTP is NP, and prove how to map the decision version of the set covering problem (SCP) to the B-MRSTP.

Definition 10.3. *The bounded minimum representation spanning tree problem: Given an ELG $G = (V,E,L)$ and a constant $k \in \mathbb{Z}^+$, is there a spanning tree T of G such that $Rt(T) \leq k$?*

Lemma 10.1. *The B-MRSTP is NP.*

Proof. *It is easy to see that B-MRSTP $\in$ NP since an algorithm can check in polynomial time if the solution edges connect all the vertices and $Rt(T) \leq k$.* □

Lemma 10.2. *The B-MRSTP is NP-hard.*

Proof. *This Lemma is proved by transforming the the decision version of the set covering problem (B-SCP), that is NP-complete (Karp, 1972), to the B-MRSTP. Given U the universe set, $\mathcal{S}$ a set of subsets of U, and a constant $k \in \mathbb{Z}^+$, the question associated with the B-SCP is if there exists a set $\mathcal{C} \subseteq \mathcal{S}$ such that $|\mathcal{C}| \leq k$ and $\bigcup_{S \in \mathcal{C}}(S) = U$.*

First, let $G(U,\mathcal{S}) = (V,E,L)$ be an ELG built as follows: consider the sets $V = \{p,q\}$, $L = \{W\}$, and $E = \{e = (p,q)\}$ such that $l(e) = W$, initially composed only by auxiliary elements. Then, add a vertex $v(u)$ to V for each $u \in U$, add a label S to L for each set $S \in \mathcal{S}$, and for each $u \in U$ and $S \in \mathcal{S}$ such that $u \in S$ add the edge $e = (p,v(u))$ with the label $l(e) = S$ to E.

Figure 10.2 illustrates the construction of the graph $G(U,\mathcal{S})$. Fig. 10.2(a) presents the sets U and $\mathcal{S}$ of an example instance of the SCP. Fig. 10.2(b) shows the ELG $G(U,\mathcal{S}) = (V,E,L)$ built from this instance. Fig. 10.2(c) presents a solution of the MRSTP for the graph $G(U,\mathcal{S})$.

Observe that the construction of the graph $G(U,\mathcal{S})$ can be accomplished in a polynomial a)

$U = \{1, 2, 3, 4, 5, 6\}$

$\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

$S_1 = \{5\}$ $S_2 = \{1, 2, 3\}$

$S_3 = \{1, 2\}$ $S_4 = \{4, 6\}$

$S_5 = \{3\}$ $S_6 = \{4, 5, 6\}$

$U \setminus \mathcal{S}$	S_1	S_2	S_3	S_4	S_5	S_6
1		🟩	🟦			
2		🟩	🟦			
3		🟩			▦	
4				▨		⬛
5	🟥					⬛
6				▨		⬛

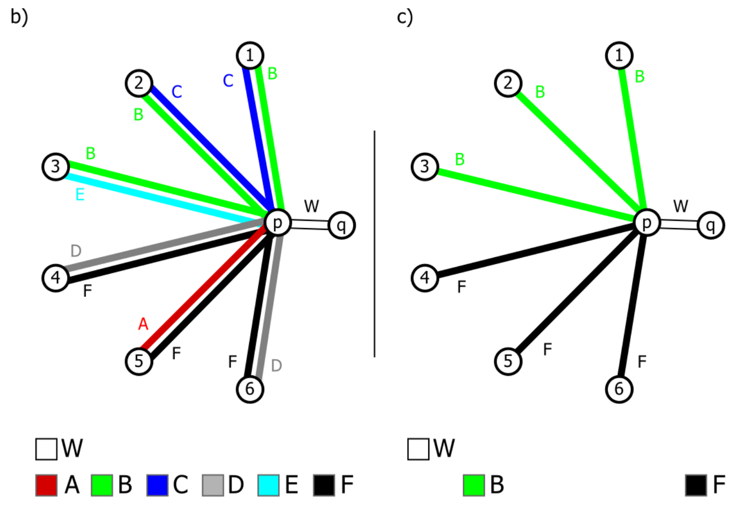


Figure 10.2: Transformation from the SCP to the MRSTP. **(a)** An input instance $(U, \mathcal{S})$ for the set covering problem. **(b)** the instance of the MRSTP originated from $(U, \mathcal{S})$. **(c)** The solution of the MRSTP for this instance

number of operations. Further, G has a spanning tree T with $Rt(T) \leq k$ if and only if the SCP for $(U, \mathcal{S})$ has a solution $\mathcal{C}$ with $|\mathcal{C}| \leq k$. □

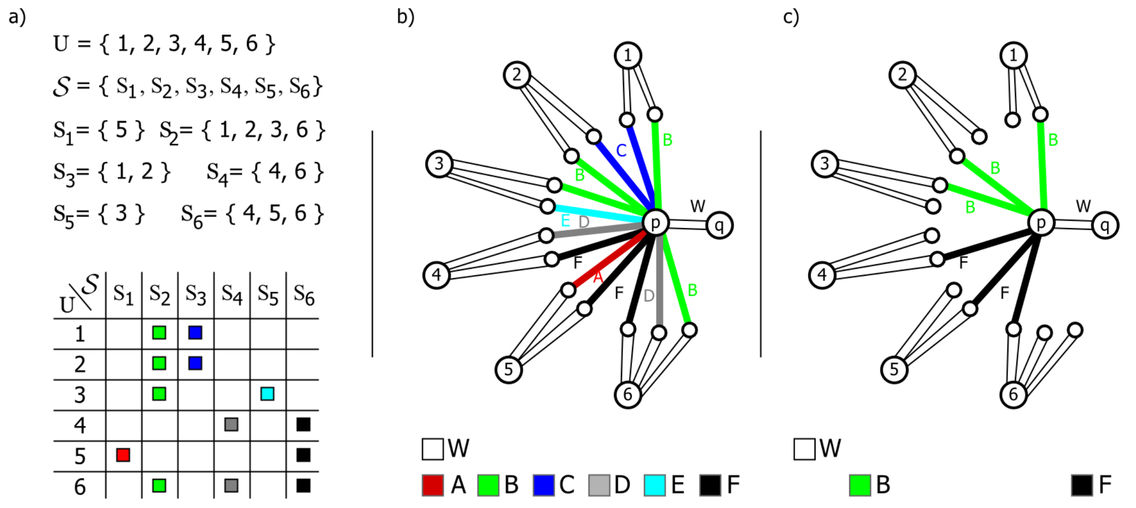
Theorem 10.1. *B-MRSTP is NP-complete.*

Proof. *This theorem is a direct consequence of Lemmas 10.1 and 10.2.* □

Observe that the construction proposed in the proof of Lemma 10.2 only proves the MRSTP is NP-hard for multigraph instances of the problem. However, we can change the graph $G(U, \mathcal{S}) = (V, E, L)$ so that it becomes simple. For each edge $e = (p, v(u)) \in E \setminus \{e' = (p, q)\}$, remove e from E, add to V a new vertex $v(e)$, and add to E the edges $k' = (p, v(e))$, $k'' = (v(e), v(u))$, such that $l(k') = l(e)$ and $l(k'') = W$. It is easy to see that G has a spanning tree T with $Rt(T) \leq k + |U|$ if and only if the SCP has a solution $\mathcal{C}$ with $|\mathcal{C}| \leq k$. Further, remark that every graph $G(U, \mathcal{S})$ is planar by construction, and then the MRSTP is NP-hard even for planar input graphs. Figure 10.3 illustrates this new transformation.

10.1 A MIP-based exact method

In this section we propose a MIP formulation for the MRSTP. Let x_e be a binary variable that is set to 1 if the edge $e \in E$ is in the final solution and to 0 otherwise, and let z_l^v be a binary variable that is set to 1 if the label $l \in L$ is represented in the vertex $v \in V$ and to 0 otherwise. The formulation is presented in the program (10.1) to (10.6).



$U = \{1, 2, 3, 4, 5, 6\}$

$\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

$S_1 = \{5\}$ $S_2 = \{1, 2, 3, 6\}$

$S_3 = \{1, 2\}$ $S_4 = \{4, 6\}$

$S_5 = \{3\}$ $S_6 = \{4, 5, 6\}$

$U \backslash \mathcal{S}$	S_1	S_2	S_3	S_4	S_5	S_6
1		🟩	🟦			
2		🟩	🟦			
3		🟩			▣	
4				▨		■
5	🟥					■
6		🟩		▨		■

Figure 10.3: Transformation from the SCP to the MRSTP in a simple ELG. **(a)** An input instance $(U, \mathcal{S})$ for the set covering problem. **(b)** the simple ELG originated from $(U, \mathcal{S})$. **(c)** The solution of the MRSTP for this instance

The objective function (10.1) models the representation function Rt. The set of inequalities (10.2) ensures the connectivity of the solution. The constraints (10.3) and (10.4) bind the edges to the label variables. The constraint (10.5) strengthens the formulation by reinforcing the solution is a tree. Finally, the domain of the variables is defined in (10.6).

$$\text{Minimize} \left(\sum_{v \in V} \sum_{l \in L} z_l^v \right) - |V| \tag{10.1}$$

$$\text{s.t.} \quad \sum_{x_e \in \delta(S)} x_e \geq 1, \qquad \forall S \subset V, S \neq \emptyset, \tag{10.2}$$

$$z_{l(e)}^v \geq x_e, \qquad \forall e = (v, w) \in E, \tag{10.3}$$

$$z_{l(e)}^w \geq x_e, \qquad \forall e = (v, w) \in E, \tag{10.4}$$

$$\sum_{x_e \in E} x_e = |V| - 1, \tag{10.5}$$

$$z_l^v, x_e \in \{0, 1\}, \qquad \forall l \in L, v \in V, e \in E. \tag{10.6}$$

10.2 Constructive heuristics

Moreover, we propose the *KBased* and the *PBased* algorithms, two new greedy constructive heuristics for the MRSTP inspired by Kruskal's and Prim's algorithms (Cormen et al., 2009), respectively. Let $G = (V,E,L)$ be the input ELG, and $\mathcal{K}$ be the set of the maximal monochromatic connected components (MMCC) of G. The KBased algorithm starts with the solution $E' = \emptyset$ and, while $G(V,E',L)$ is not connected, at each iteration, it adds to E', avoiding to create any cycles, the edges of the MMCC $k \in \mathcal{K}$ that minimizes the number of connected components of $G(V,E',L)$.

Algorithm 10.1 describes the KBased heuristic. The necessary initializations are carried out in the lines 2 and 3. Each iteration of the main loop (lines 5 to 9) looks for the MMCC that minimizes the number of connected components, denoted by $W(G)$, of the solution (line 6) and adds its non-cycle edges to the final graph (lines 7 to 9). Notice that, given a solution tree, its Rt function can be computed in $O(|V|^2)$.

Algorithm 10.1: The KBased heuristic for the MRSTP

```
1  procedure KBased($G = (V,E,L)$)
2      Let $E' \leftarrow \emptyset$ be the set of edges of the solution;
3      Let $\mathcal{K}$ be the set of MMCCs of $G$;
4      Let $E(K)$ be the set of edges of the MMCC $K \in \mathcal{K}$ ;
5      while $|E'| < |V| - 1$ do
6          $k \leftarrow argmin_{K \in \mathcal{K}}(W(G = (V, E' \cup E(K), L)))$;
7          foreach $e \in E(k)$ do
8              if $G = (V, E' \cup \{e\}, L)$ has no cycles then
9                  $E' \leftarrow E' \cup \{e'\}$ ;
10     return $G = (V,E',L)$;
```

In its turn, the PBased heuristic is a variant of the KBased algorithm. It starts with a maximal connected component $S = \{r\}$, where $r \in V$ is an arbitrarily chosen root node, and executes the same main loop as the KBased algorithm. The difference is that the PBased heuristic only considers to enter the solution the MMCCs that make S to grow.

10.3 Computational experiments

In this section we describe the computational experiments performed to evaluate the performance of both the proposed heuristics and the mathematical formulation. All the methods were implemented in C++ language and compiled by using g++ 4.8.4, with the optimization flag -O3. The mathematical formulation and its derived procedures were implemented using the Concert library and Cplex 12.51 as the solver. The experiments were performed on a computer with Intel(R) Core(TM) i7, 64 bits, CPU, 4.00GHz, 16 GB of RAM, and Ubuntu 14.04 as the operating system. Although the processor of this device has more than one core, the algorithms were executed by using a single core and a single thread within a time limit of two hours.

For this set of experiments, we have considered the 120 edge-labeled graphs of the group 1, generated by Cerulli et al. (2005). This group of ELGs has instances with number of vertices $n = |V| \in \{20, 30, 40, 50\}$, number of labels $l = |L| = n$, and edge densities $d \in \{\text{ld} = 0.2, \text{md} = 0.5, \text{hd} = 0.8\}$. Also, each dataset consists in 10 different graphs for a *n-l-d* configuration.

Moreover, given the exponential size of the group of inequalities (10.2), it is not practical to solve the model with all of these constraints. In such a case, given a solution for the linear relaxation of the model, we separate the inequalities (10.2) by using a simple DFS procedure: compute the connected components of the solution graph by considering only the edges $e \in E$ that have $x_e > 0$, and add an inequality (10.2) for each maximal connected component found (if the graph is not connected). This separation procedure is also executed whenever an integer solution is found, to ensure its feasibility.

The results of the experiments are reported in Table 10.1. Each line represents a dataset with 10 different graphs for one *n-l-d* configuration, and the first group of columns identifies this dataset. The remaining columns are divided into three groups: *KBased*, *PBased* and *Exact*, related to the results obtained by the proposed heuristics and by the mathematical formulation, respectively. The column *rt* reports the average of the best integer solutions found by the methods. The columns *t(ms)* and *t(s)* indicates the average running time in milliseconds and seconds, respectively. The column *O* reports the number of optimal solutions found by the exact method. The column *gap* represents (as a percentage) the average value of the differences between the UBs and LBs achieved by the exact method. Finally, the column *gapr* reports (as a percentage) the average of the differences between the UBs and the linear relaxations[1].

Regarding the heuristics, we can observe that the KBased has obtained the best results, having lost to the PBased only in two datasets: 20-20-*hd* and 40-40-*md*. As expected from constructive greedy heuristics, both KBased and PBased has presented very small running times, ranging from 0.001 to 9.4 milliseconds, with a slight advantage for the KBased. Their objective function results, however, are not too close to the ones obtained by the exact method. Despite of that, these heuristics arise as interesting alternatives for larger instances (e.g. 50-50-*hd*), when the exact method is not able to find good integer solutions. Another way to improve these *rt* results is to use these heuristics to provide initial solutions for metaheuristic methods.

From the results reported, we have that the exact method performed very well for graphs with $n \leq 30$, having solved 57 out of 60 instances. However, the performance drops very quickly for graphs with $n \geq 40$, when the method solved only 34 out of 60 instances. Also, it is possible

[1]The values for the linear relaxations were obtained by setting Cplex to *NodeLimit(1)*.

Table 10.1: Computational experiments for instances with $|V| \in \{20, 30, 40, 50\}$

Instance		**KBased**		**PBased**		**Exact**				
n	**d**	**rt**	**t(ms)**	**rt**	**t(ms)**	**rt**	**t(s)**	**O**	**gap**	**gapr**
	ld	12.4	0.0	12.7	0.0	**12.3**	0.0	**10**	0.00%	1.25%
20	md	6.3	0.1	6.8	0.1	**5.7**	0.2	**10**	0.00%	3.04%
	hd	4.8	0.0	4.5	0.0	**2.8**	5.9	**10**	0.00%	6.16%
	ld	16	1.1	16.4	1.0	**15.3**	0.3	**10**	0.00%	1.12%
30	md	9.3	1.6	9.5	1.6	**6.7**	10.3	**10**	0.00%	2.98%
	hd	5.7	1.1	6.4	1.3	**3.8**	2845.7	7	1.97%	8.78%
	ld	19.7	3.6	20.8	4.0	**18.9**	20.4	**10**	0.00%	2.08%
40	md	12.7	4.5	12.6	4.4	**8.6**	1855.2	8	0.89%	4.21%
	hd	8.7	3.1	8.8	3.1	**5.9**	5585.1	3	7.79%	11.21%
	ld	23.7	7.9	24.1	8.1	**21.7**	728.7	9	0.17%	1.84%
50	md	14.3	8.9	14.9	9.4	**10**	5062.1	4	1.93%	4.75%
	hd	**12.6**	6.1	13.9	7.2	15.1	7200.0	0	19.97%	21.27%

to observe from its running times that the complexity of the problem grows with the number of edges on the input graph, to the point of the exact method not being able to solve any instance on the dataset 50-50-*hd*. Further, for the instance 7 of this group, the exact method was not able to find any feasible integer solution[2].

Lastly, it is possible to state that the MRSTP is harder to solve to optimality in comparison to the MLSTP. The mathematical formulation CCut was able to solve the MLSTP for all of the instances with 100 vertices of the group 2 of Cerulli et al. (2005), while the exact method proposed for the MRSTP has failed to solve an entire set of instances with 50 vertices. In this sense, we believe that further researches can propose improved exact, heuristic and meta-heuristic methods for this problem, as well as to study more deeply its characteristics in order to introduce reduction rules or preprocessing procedures.

[2]We have used the *rt* value obtained by the KBased heuristic for this instance to compute the averages reported on the columns *rt*, *gap*, and *gapr* of the exact method in the line 50-50-*hd*.

Chapter 11

Concluding Remarks and Future Work

In this thesis we have addressed several connectivity problems defined over edge-labeled graphs, in special the minimum labeling spanning tree problem and its generalized version. We have carried out an extensive literature review on these problems, describing the state-of-the-art contributions in each field. By observing the recent publications, one can verify that this theme is indeed an area of interest for researches.

The contributions of this work are both theoretical and practical. On the theoretical side, we have introduced new useful concepts, definitions, properties and theorems regarding edge-labeled graphs, as well as a polyhedral study on the GMLSTP. We can summarize the main theoretical contributions of this work as follows:

- We have introduced the concept of *label contraction* and some interesting properties relating contracted graphs with graphs induced by a set of labels;
- We have provided some results for the CCut polytope, in particular concerning its dimension and its facet compositions. New valid inequalities were introduced, and the conditions in which they define facets have been given;
- We have performed polyhedral comparisons against the polytope associated with state-of-the-art formulations and their variations. Our results have showed that CCut formulation theoretically performs better with respect to its polytope than all currently available mathematical formulations for the GMLSTP and MLSTP;
- We have proposed a new monochromatic cycles removal procedure (MCR) that can be carried out in $O(\alpha(m,n) \cdot m)$, where α stands for the inverse of the Ackerman's function.
- We have proposed the heuristic rMVCA, a revised version of MVCA, and have proved

its worst case time complexity is $O(\alpha_n kn)$, where $n = |V|$, $k = |L|$, and $\alpha_n = \alpha(n,n)$. It is a tighter bound on the running time complexity of the MVCA.

On the practical side, we have proposed new heuristics — such as the metaheuristic-based algorithm MSLB, and the constructive heuristic pMVCA — and exact methods — such as new mathematical formulations and branch-and-cut algorithms — for solving the GMLSTP. We can summarize the main practical contributions on this work as follows:

- We have proposed the MSLB, a new MIP-based metaheuristic that combines the efficiency of a parametrized version of rMVCA with the capacity of exploration of a new local search method based on local branching techniques;
- We have carried out computational experiments to compare the MSLB with the state-of-the-art metaheuristics for the MLSTP. The results have showed that the MSLB achieved the best performance both in quality of solutions and processing times.
- We have presented a new mathematical formulation based on the concept of *colorful cuts* —namely CCut— for solving the GMLSTP. It is the first model to use only decision variables for labels.
- We have proposed branch-and-cut algorithms for CCut and compared them with the state-of-the-art exact methods for both the MLSTP and the GMLSTP. The computational experiments have showed that the proposed methods outperform the previous ones.
- We have improved the CCut formulation by proposing and separating a new set of valid inequalities, namely the partitioning cuts inequalities;
- We have introduced CCutBB and CCutHB, two new branching strategies for solving the CCut formulation. The computational experiments performed show that, from the best of our knowledge, the new approaches were able to achieve the best results regarding exact methods for the MLSTP so far.

The last part of this thesis has addressed problems related to the MLSTP, to be specific, connectivity problems defined over edge-labeled graphs (or digraphs). We have introduced new variants of the MLSTP, such as the minimum representation spanning tree problem and the prize collecting MLSTP. We have also proposed new mathematical formulations based on CCut for solving these problems.

Further, we have dedicated special attention to the minimum labeling global cut problem, since its weighted version arises as the separation problem for the colorful cuts inequalities of the CCut formulation. We have proposed the first MIP-based formulations to solve this problem. The computational experiments performed showed that the proposed methods are able to solve small to medium instances to optimality in a reasonable amount of time.

In future research projects, we intend to study some heuristic pre-processing techniques to apply on the input ELG without losing its optimality. Moreover, we further plan to adapt the MSLB in order to solve other problems defined on ELG, such as the minimum labeling path problem and the minimum global cut problem, aiming to evaluate this method in a more general way. Moreover, we believe that the search for more facet-defining families of inequalities should be continued, as well as the polyhedral studies for the related problems discussed in this work.

References

Balas, E. “The prize collecting traveling salesman problem”. In: *Networks* 19.6 (1989), pp. 621–636 (cit. on p. 121).

Ballou, R. H. *Gerenciamento da Cadeia de Suprimentos-: Logística Empresarial*. Bookman, 2006 (cit. on p. 9).

Broersma, H. and X. Li. “Spanning trees with many or few colors in edge-colored graphs”. In: *Discussiones Mathematicae Graph Theory* 17.2 (1997), pp. 259–269 (cit. on p. 12).

Broersma, H., X. Li, G. Woeginger, and S. Zhang. “Paths and cycles in colored graphs”. In: *Australasian Journal on Combinatorics* 31 (2005), pp. 299–311 (cit. on p. 119).

Brüggemann, T., J. Monnot, and G. J. Woeginger. “Local search for the minimum label spanning tree problem with bounded color classes”. In: *Operations Research Letters* 31.3 (2003), pp. 195–201 (cit. on pp. 3, 12).

Bui, T. N. and C. M. Zrncic. “An ant-based algorithm for finding degree-constrained minimum spanning tree”. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM. 2006, pp. 11–18 (cit. on p. 2).

Captivo, M., J. a. C. N. Clímaco, and M. M. B. Pascoal. “A Mixed Integer Linear Formulation for the Minimum Label Spanning Tree Problem”. In: *Comput. Oper. Res.* 36.11 (2009), pp. 3082–3085 (cit. on pp. 13, 21).

Carr, R. D., S. Doddi, G. Konjevod, and M. V. Marathe. “On the red-blue set cover problem”. In: *Symposium on Discrete Algorithms: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. Vol. 9. 2000, pp. 345–353 (cit. on pp. 3, 119).

Carrabs, F., R. Cerulli, and P. Dell'Olmo. "A mathematical programming approach for the maximum labeled clique problem". In: *Procedia-Social and Behavioral Sciences* 108 (2014), pp. 69–78 (cit. on p. 116).

Carrabs, F., R. Cerulli, and M. Gentili. "The labeled maximum matching problem". In: *Computers & Operations Research* 36.6 (2009), pp. 1859–1871 (cit. on pp. 3, 116).

Carrabs, F., C. Cerrone, R. Cerulli, and S. Silvestri. "The rainbow spanning forest problem". In: *Soft Computing* (2017), pp. 1–12 (cit. on p. 116).

Cerrone, C., R. Cerulli, and B. Golden. "Carousel greedy: a generalized greedy algorithm with applications in optimization". In: *Computers & Operations Research* 85 (2017), pp. 97–112 (cit. on pp. 12, 15, 16, 99, 116).

Cerrone, C., R. Cerulli, and M. Gaudioso. "Omega one multi ethnic genetic approach". In: *Optimization Letters* 10.2 (2016), pp. 309–324 (cit. on p. 13).

Cerulli, R and D Granata. "Varianti colorate del problema del massimo flusso". In: *Technical report* (2009) (cit. on p. 124).

Cerulli, R., A. Fink, M. Gentili, and S. Voß. "Extensions of the minimum labelling spanning tree problem". In: *Journal of Telecommunications and Information Technology* 4 (2006), pp. 39–45 (cit. on pp. 3, 120).

Cerulli, R., P. Dell'Olmo, M. Gentili, and A. Raiconi. "Heuristic approaches for the minimum labelling hamiltonian cycle problem". In: *Electronic notes in discrete mathematics* 25 (2006), pp. 131–138 (cit. on p. 116).

Cerulli, R., A. Fink, M. Gentili, and S. Voß. "Metaheuristics Comparison for the Minimum Labelling Spanning Tree Problem". In: *The Next Wave in Computing, Optimization, and Decision Technologies*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Vol. 29. Operations Research/Computer Science Interfaces Series. Springer US, 2005, pp. 93–106 (cit. on pp. 13, 15, 41, 89, 90, 92, 105, 137, 148, 149).

Cerulli, R., A. Fink, M. Gentili, and A. Raiconi. "The k-labeled spanning forest problem". In: *Procedia - Social and Behavioral Sciences* 108 (2014), pp. 153–163 (cit. on p. 117).

Chang, R.-S. and S.-J. Leu. “The Minimum Labeling Spanning Trees”. In: *Inf. Process. Lett.* 63.5 (1997), pp. 277–282 (cit. on pp. 2, 3, 12–15, 106, 123, 124, 143).

Chen, Y., N. Cornick, A. O. Hall, R. Shajpal, J. Silberholz, I. Yahav, and B. L. Golden. “Comparison of heuristics for solving the gmlst problem”. In: *Telecommunications Modeling, Policy, and Technology*. Springer, 2008, pp. 191–217 (cit. on pp. 4, 13).

Chopra, S. and P. Meindl. *Supply chain management. Strategy, planning & operation*. Springer, 2007 (cit. on p. 9).

Chwatal, A. M. and G. R. Raidl. “Solving the Minimum Label Spanning Tree Problem by Ant Colony Optimization.” In: *GEM*. Ed. by H. R. Arabnia, R. R. Hashemi, and A. M. G. Solo. CSREA Press, 2010, pp. 91–97 (cit. on p. 13).

— “Solving the Minimum Label Spanning Tree Problem by Mathematical Programming Techniques.” In: *Adv. Operations Research* 2011 (2011) (cit. on pp. 13, 14, 20–23, 40, 43).

Chwatal, A. M., G. R. Raidl, and K. Oberlechner. “Solving an Extended Minimum Label Spanning Tree Problem to Compress Fingerprint Templates”. In: *Journal of Mathematical Modelling and Algorithms* 8.3 (2009). previous technical report version at `https://www.ac.tuwien.ac.at/files/pub/chwatal-08a.pdf`, pp. 293–334 (cit. on p. 2).

Clímaco, J. C. N., M. E. V. Captivo, and M. M. B. Pascoal. “On the bicriterion - minimal cost/minimal label - spanning tree problem.” In: *European Journal of Operational Research* 204.2 (2010), pp. 199–205 (cit. on p. 118).

Consoli, S, J. Moreno, N Mladenović, and K Darby-Dowman. “Constructive heuristics for the minimum labelling spanning tree problem: a preliminary comparison”. In: (2006) (cit. on p. 16).

Consoli, S., N. Mladenović, and J. Moreno Pérez. “Solving the Minimum Labelling Spanning Tree Problem by Intelligent Optimization”. In: *Appl. Soft Comput.* 28.C (2015), pp. 440–452 (cit. on pp. 13, 15, 18, 19, 105–107).

Consoli, S, K Darby-Dowman, N Mladenović, and J. Moreno-Perez. “Solving the minimum labelling spanning tree problem using hybrid local search”. In: (2007) (cit. on p. 13).

Consoli, S., K. Darby-Dowman, N. Mladenovic, and J. A. Moreno-Pérez. "Greedy Randomized Adaptive Search and Variable Neighbourhood Search for the minimum labelling spanning tree problem". In: *European Journal of Operational Research* 196.2 (2009), pp. 440 –449 (cit. on pp. 2, 6, 13, 15, 18, 105).

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009 (cit. on pp. 2, 30, 97, 146).

Coudert, D., S. Perennes, H. Rivano, and M.-E. Voge. "Combinatorial optimization in networks with Shared Risk Link Groups". In: *Hal - Inria, Open Archive*. 2014 (cit. on p. 132).

Coudert, D., P. Datta, S. Pérennes, H. Rivano, and M.-E. Voge. "Shared risk resource group complexity and approximability issues". In: *Parallel Processing Letters* 17.02 (2007), pp. 169–184 (cit. on pp. 3, 131, 132).

Dantzig, G. B., A. Orden, P. Wolfe, et al. "The generalized simplex method for minimizing a linear form under linear inequality restraints". In: *Pacific Journal of Mathematics* 5.2 (1955), pp. 183–195 (cit. on p. 83).

Demšar, J. "Statistical comparisons of classifiers over multiple data sets". In: *Journal of Machine learning research* 7.Jan (2006), pp. 1–30 (cit. on p. 114).

Derrac, J., S. García, D. Molina, and F. Herrera. "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms". In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 3–18 (cit. on p. 114).

Duin, C., S. Voß, et al. "The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs". In: *Networks* 34.3 (1999), pp. 181–191 (cit. on p. 12).

Fischetti, M. and A. Lodi. "Local branching". In: *Mathematical programming* 98.1-3 (2003), pp. 23–47 (cit. on p. 100).

Granata, D., R. Cerulli, M. G. Scutellà, and A. Raiconi. "Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs". In: *Handbook of Combinatorial Optimization*. Springer, 2013, pp. 1913–1948 (cit. on pp. 3, 12, 124, 126).

Grötschel, M. and Y. Wakabayashi. “Facets of the clique partitioning polytope”. In: *Mathematical Programming* 47.1 (1990), pp. 367–387 (cit. on p. 134).

Ismkhan, H. “Effective three-phase evolutionary algorithm to handle the large-scale colorful traveling salesman problem”. In: *Expert Systems with Applications* 67 (2017), pp. 148–162 (cit. on p. 116).

Jacob, R, G Koniedov, S Krumke, M Marathe, R Ravi, and H Wirth. “The minimum label path problem”. In: *Unpublished manuscript, Los Alamos National Laboratory* (1999) (cit. on p. 119).

Jha, S., O. Sheyner, and J. Wing. “Two formal analyses of attack graphs”. In: *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*. IEEE. 2002, pp. 49–63 (cit. on p. 131).

Jothi, R. and B. Raghavachari. “Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design”. In: *ACM Transactions on Algorithms (TALG)* 1.2 (2005), pp. 265–282 (cit. on p. 2).

Karp, R. M. *Reducibility among combinatorial problems*. Springer, 1972 (cit. on pp. 2, 14, 144).

Krumke, S. O. and H.-C. Wirth. “On the Minimum Label Spanning Tree Problem”. In: *Inf. Process. Lett.* 66.2 (1998), pp. 81–85 (cit. on pp. 15, 29, 30, 34).

Lam, W. H., N. M. Holyoak, and H. Lo. “How park-and-ride schemes can be successful in Eastern Asia”. In: *Journal of urban planning and development* 127.2 (2001), pp. 63–78 (cit. on p. 7).

Myung, Y.-S., C.-H. Lee, and D.-W. Tcha. “On the generalized minimum spanning tree problem”. In: *Networks* 26.4 (1995), pp. 231–241 (cit. on p. 2).

Nummela, J. and B. A. Julstrom. “An effective genetic algorithm for the minimum-label spanning tree problem”. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM. 2006, pp. 553–558 (cit. on p. 13).

Perez, J. and S. Consoli. “On the Minimum Labelling Spanning bi-Connected Subgraph problem”. In: *Annals of MIC 2015: The XI Metaheuristics International Conference, arXiv preprint arXiv:1505.01742* (2015) (cit. on p. 127).

Resende, M. G. “Greedy randomized adaptive search procedures Greedy Randomized Adaptive Search Procedures: GPASP Greedy randomized adaptive search procedures GPASP”. In: *Encyclopedia of optimization* (2009), pp. 1460–1469 (cit. on p. 99).

Ruiz, R. and T. Stützle. “A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem”. In: *European Journal of Operational Research* 177.3 (2007), pp. 2033–2049 (cit. on p. 12).

Silva, T. G., E. V. Queiroga, L. S. Ochi, and L. A. F. Cabral. “Novos Ótimos para o Poblema da Árvore Geradora com Rotulação Mínima”. In: *Anais do XLVII Simpósio Brasileiro de Pesquisa Operacional*. Porto de Galinhas, 2015, pp. 4214–4225 (cit. on p. 89).

Silvestri, S., G. Laporte, and R. Cerulli. “The rainbow cycle cover problem”. In: *Networks* 68.4 (2016), pp. 260–270 (cit. on p. 116).

Tanenbaum, A. S. *Redes de computadores*. Pearson, 2003 (cit. on p. 6).

Tang, L. and P. Zhang. “Approximating minimum label st cut via linear programming”. In: *LATIN 2012: Theoretical Informatics*. Springer, 2012, pp. 655–666 (cit. on p. 132).

Van-Nes, R. “Design of Multimodal Transport Networks: A Hierachical Approach”. PhD thesis. Delft University, 2002 (cit. on pp. 2, 7).

Wan, Y., G. Chert, and Y. Xu. “A Note on the Minimum Label Spanning Tree”. In: *Inf. Process. Lett.* 84.2 (2002), pp. 99–101 (cit. on p. 15).

Wolsey, L. A. *Integer programming*. Wiley, 1998 (cit. on pp. 40, 48, 49).

Wolsey, L. A. and G. L. Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014 (cit. on pp. 40, 53).

Xiong, Y., B. L. Golden, and E. A. Wasil. “A one-parameter genetic algorithm for the minimum labeling spanning tree problem.” In: *IEEE Trans. Evolutionary Computation* 9.1 (2005), pp. 55–60 (cit. on pp. 4, 13).

Xiong, Y., B. Golden, and E. Wasil. “Improved Heuristics for the Minimum Label Spanning Tree Problem”. In: *Evolutionary Computation, IEEE Transactions on* 10.6 (2006), pp. 700–703 (cit. on p. 13).

Xiong, Y., B. Golden, and E. Wasil. “The colorful traveling salesman problem”. In: *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*. Springer, 2007, pp. 115–123 (cit. on p. 116).

Xiong, Y., B. Golden, E. Wasil, and S. Chen. “The label-constrained minimum spanning tree problem”. In: *Telecommunications Modeling, Policy, and Technology*. Springer, 2008, pp. 39–58 (cit. on p. 118).

Xiong, Y., B. Golden, and E. Wasil. “Worst-case behavior of the MVCA heuristic for the minimum labeling spanning tree problem”. In: *Operations Research Letters* 33.1 (2005), pp. 77–80 (cit. on p. 15).

Zhang, P. “Efficient algorithms for the label cut problems”. In: *Theory and Applications of Models of Computation*. Springer, 2014, pp. 259–270 (cit. on p. 132).

Zhang, P., J.-Y. Cai, L.-Q. Tang, and W.-B. Zhao. “Approximation and hardness results for label cut and related problems”. In: *Journal of Combinatorial Optimization* 21.2 (2011), pp. 192–208 (cit. on p. 132).