FORMULAÇÕES E ALGORITMOS EXATOS PARA O PROBLEMA DA ÁRVORE GERADORA MÍNIMA COM PARES DE ARESTAS CONFLITANTES

PHILLIPPE SAMER LALLO DIAS

FORMULAÇÕES E ALGORITMOS EXATOS PARA O PROBLEMA DA ÁRVORE GERADORA MÍNIMA COM PARES DE ARESTAS CONFLITANTES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: SEBASTIÁN ALBERTO URRUTIA

Belo Horizonte

Fevereiro de 2014

PHILLIPPE SAMER LALLO DIAS

FORMULATIONS AND EXACT ALGORITHMS FOR THE MINIMUM SPANNING TREE PROBLEM WITH CONFLICTING EDGE PAIRS

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: SEBASTIÁN ALBERTO URRUTIA

Belo Horizonte

February 2014

Ficha catalográfica elaborada pela Biblioteca do ICEx – UFMG

Dias, Phillippe Samer Lallo

D541f Formulations and Exact Algorithms for the Minimum Spanning Tree Problem with Conflicting Edge Pairs / Phillippe Samer Lallo Dias. — Belo Horizonte, 2014
xxx, 54 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação.

Orientador: Sebastián Alberto Urrutia

1. Computação – Teses. 2. Otimização combinatória – Teses. 3. Programação inteira – Teses. I. Orientador. III. Título.

519.6*61(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Formulações e algoritmos exatos para o problema da árvore geradora mínima com arestas conflitantes

PHILLIPPE SAMER LALLO DIAS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. SEBASTIÁN ALBERTO URRUTIA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ABÍLIO PEREIRA DE LUCENA FILHO
Faculdade de Economia e Administração - UFRJ

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

PROF. MARTIN GOMEZ RAVETTI
Departamento de Engenharia de Produção - UFMG

PROF. THIAGO FERREIRA DE NORONHA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 14 de fevereiro de 2014.

Trabalho dedicado a meus pais, Mércio e Sandra, pelo apoio e inspiração cotidiana de qualidades cada vez mais fundamentais para perseguir ***qualquer*** *caminho de virtude, e aos irmãos Angélica e Fábio, companheiros essenciais e que dão sentido à jornada.*

Acknowledgments

I try to follow the framework quoted by John Shaw Billings [1838-1913] to summarize here the most clear responsibles for this work to exist.

First have something to say. Este trabalho é fruto de uma parceria e verdadeira amizade com o professor Sebastián, cuja confiança desde o início e sempre renovada fonte de inspiração fizeram a diferença em minha trajetória acadêmica. Obrigado pela paciência, pela ajuda em tantas oportunidades e, claro: junto com Anolan, por todas as sugestões, conversas e noites de jogos que fizeram desta uma fase inesquecível!

Second, say it. Agradeço à minha família por tudo o que permitiu chegar até aqui. Aos valores e exemplos de meus pais, além do suporte e incentivo incondicionais. Ao meu irmão Fábio, que surge com tanta energia e incentiva a trabalhar com pés no chão. E especialmente ao carinho de minha irmã Angélica, entre eventuais puxões de orelha e momentos felizes com cachorras! (:

Ah, momentos felizes! Qual seria meu estado emocional sem os amigos... Devo-lhes toda minha gratidão, com direito a alguns destaques: o sorriso da Evellyn, o exemplo do Afonso, o companheirismo do Magal, a diligência do Thiago Rafael, a sabedoria do Felipe Loredo, e as velhas amizades de Rafael, Marco Túlio e Yuri que ainda hoje edificam encontros de pura energia - e dança! Aos parceiros do LaPO, muitíssimo obrigado pelo apoio e diversão! Em especial aos sensacionais Vitor, Ramon e Marco Túlio (RR) – amigos preciosos e companhias que fazem a diferença! E claro, a todos os companheiros do DCC que tentam construir um ambiente melhor.

O DCC! Impossível agradecer à altura do impressionante trabalho da secretaria. Muito obrigado pela ajuda em incontáveis ocasiões. Um beijo especial para Linda, Sheila, Sônia e Renata!

Third, stop when you have said it. Duas surpresas finais nos planos de inspiração e expectativas. Evellyn: sua companhia é uma graça desde o início, e espero que nossa parceria cresça tanto quanto este frio na barriga... Ao fim, tive o privilégio de uma avaliação tão cuidadosa e positiva de uma banca formada por professores de reconhecida experiência e destaque acadêmico. Obrigado Martín, Thiago e Sebastián, e agradecimentos especiais ao Robson, pelo modelo de professor e profissional que representa, além da formação que permitiu desenvolver este trabalho; e ao Abílio, por revisão e sugestões tão fortes quanto a bagagem que carrega. Vocês me inspiram e espero poder desenvolver nossa parceria ainda mais.

Finally, give it an accurate title. Um título de interjeições e exclamações parece inapropriado. Só espero deixar claro que, sem vocês, nada disso seria concebível. Muito obrigado!

“ O que dá o verdadeiro sentido ao encontro é a busca,
e é preciso andar muito para se alcançar o que está perto. ”
(José Saramago)

Abstract

This work presents approaches for the exact solution of the minimum spanning tree problem under conflict constraints. Given a graph $G(V,E)$ and a set $C \subset E \times E$ of conflicting edge pairs, the problem consists of finding a conflict-free minimum spanning tree, i.e. feasible solutions may include at most one of the edges from each pair in C.

The problem is NP–hard in the general case. Although formulations and algorithms have been discussed recently in the literature, computational results indicate considerably large duality gaps and a lack of optimality certificates for the benchmark instances.

In this work, we consider polyhedral representations of conflict-free edge subsets as stable sets in an auxiliary conflict graph $\hat{G}(E,C)$. We present integer linear programming formulations including four classes of exponentially-many constraints: two of which correspond to classic polyhedral representations of spanning trees in G, and two for strengthening the intersection with relaxations of the polytope of stable sets in $\hat{G}$ (with clique and odd-cycle inequalities).

We introduce and evaluate a preprocessing method and *branch and cut* algorithms. Encouraging results consistently improve on those previously available in the literature. New feasibility and optimality certificates are provided, and stronger dual bounds are already obtained in the initial linear relaxation of the formulations, even for the hardest instances in the standard benchmark.

Keywords: Optimal trees, conflict constraints, stable set, integer programming formulations, branch and cut.

Resumo

Este trabalho apresenta abordagens para a solução exata do problema de árvores geradoras mínimas sob restrições de conflito. Dados um grafo $G(V,E)$ e um conjunto $C \subset E \times E$ de pares de arestas conflitantes, busca-se uma árvore geradora mínima de G incluindo no máximo uma das arestas de cada par em C.

O problema é NP–difícil no caso geral e, embora formulações e algoritmos tenham sido discutidos recentemente na literatura, resultados computacionais apresentavam *gaps* de dualidade consideravelmente grandes e não forneciam certificados de otimalidade para conjuntos de instâncias padrão do problema.

Neste trabalho exploramos representações poliédricas de subconjuntos de arestas livres de conflitos como conjuntos independentes em um grafo de conflitos auxiliar $\hat{G}(E,C)$. Apresentamos formulações em programação linear inteira envolvendo quatro classes de desigualdades com número exponencial de restrições: duas para representações poliédricas clássicas de árvores geradoras em G, duas fortalecendo a interseção com relaxações do politopo de conjuntos independentes em $\hat{G}$ (restringindo cliques e ciclos ímpares).

Propomos e avaliamos computacionalmente um método de pré-processamento e algoritmos *branch and cut*. As soluções obtidas superam de forma consistente os melhores resultados disponíveis na literatura anteriormente. Novos certificados de viabilidade e otimalidade são obtidos, além de limites duais mais fortes já na relaxação linear inicial das formulações, mesmo para instâncias mais difíceis do *benchmark* padrão.

Palavras-chave: Árvores ótimas, restrições de conflito, conjunto independente, formulações em programação inteira, *branch and cut*.

Resumo Estendido

Este resumo estendido visa esclarecer detalhadamente o trabalho desenvolvido. Indicamos a metodologia adotada e sumarizamos os resultados verificados. Para uma melhor apresentação em português de parte deste trabalho, sugere-se verificar o artigo correspondente nos anais do XLV Simpósio Brasileiro de Pesquisa Operacional [Samer and Urrutia, 2013].

Apresentação

Dados um grafo $G(V,E)$, um conjunto $C \subset E \times E$ de pares de arestas conflitantes, e custos $c : E \to \mathbb{R}$, o problema da árvore geradora mínima com restrições de conflito (MSTCC, do inglês *minimum spanning tree under conflict constraints*) consiste em encontrar uma árvore geradora mínima (MST, do inglês *minimum spanning tree*) livre de conflitos, i.e. uma árvore geradora de G, de mínimo custo e que inclua no máximo uma das arestas e_i ou e_j de cada par $\{e_i, e_j\} \in C$.

Este trabalho se baseia amplamente em uma definição equivalente do problema, usando o conceito de um grafo de conflitos $\hat{G}(E,C)$: denotando cada aresta no grafo original como um vértice em $\hat{G}$, representa-se cada restrição de conflito como uma aresta conectando respectivos vértices de $\hat{G}$. Assim, o problema consiste em encontrar um subconjunto de arestas de G, de mínimo custo, que corresponda simultaneamente a uma árvore geradora de G e a um conjunto independente (*stable set*) em $\hat{G}$.

O problema foi introduzido recentemente na literatura por Darmann et al. [2009, 2011], que descrevem diversos resultados sobre sua complexidade. Eles demonstram que MSTCC é fortemente NP–difícil, mesmo quando toda componente do grafo de conflitos consiste de um caminho simples de comprimento dois. Ainda mais, mostram que o problema não admite aproximação por um fator constante do ótimo, a menos que P = NP.

Novos resultados teóricos e computacionais sobre o MSTCC são descritos por Zhang et al. [2011]. Os autores discutem casos particulares que podem ser resolvidos em tempo polinomial, testes de viabilidade, heurísticas primais e dois algoritmos basea-

dos em relaxação Lagrangeana. Este trabalho é a base de comparação para avaliarmos o desempenho dos algoritmos que apresentamos nesta dissertação.

Formulações e algoritmos

Discutimos neste trabalho abordagens para a solução exata do MSTCC. Descrevemos formulações em programação linear inteira com um número exponencial de restrições, algoritmos de *branch and cut*, além de um algoritmo de pré-processamento.

A metodologia central que discutimos neste trabalho é a de explorar o grafo de conflitos $\hat{G}(E,C)$, tornando disponíveis resultados válidos para conjuntos independentes, o que inclui o uso de desigualdades válidas para o correspondente politopo $P_{stab}(\hat{G})$. Em particular, fortacelemos as formulações que propomos com desigualdades de ciclo ímpar e cliques. Sugerimos o recente tutorial de Rebennack et al. [2012] para correspondentes algoritmos de separação e uma breve introdução ao politopo $P_{stab}(\hat{G})$ e suas relaxações.

Com respeito a representações poliédricas de árvores geradoras em $G(V,E)$, exploramos neste trabalho três formulações tão justas quanto possível: uma compacta, baseada em fluxos multiproduto; duas com um número exponencial de restrições, utilizando restrições de eliminação de subciclos, e desigualdades de *cutset* direcionadas. Para uma revisão completa destas formulações e várias outras, indica-se a apresentação em [Magnanti and Wolsey, 1995, Capítulo 3].

Nosso método de solução inicia com uma fase de pré-processamento, empregando um algoritmo que propomos para a fixação de variáveis e geração de novas restrições de conflito. O algoritmo emprega condições de viabilidade tanto de árvores em G quanto de conjuntos independentes em $\hat{G}$.

Em seguida, procede-se com o arcabouço de *branch and cut* padrão, separando as classes de desigualdades com número exponencial de restrições. Uma exceção diz respeito às desigualdades clique. Conseguimos incluir *a priori* o subconjunto não dominado de desigualdades correspondendo a cliques maximais de forma muito eficiente, uma vez que o número destas nos grafos de conflitos das instâncias padrão de teste é menor que o próprio número de conflitos. Assim, obtemos uma relaxação estritamente mais forte e, simultaneamente, um modelo com menor número de restrições.

Resultados e conclusões gerais

Os experimentos computacionais que conduzimos visam avaliar a eficácia do método de pré-processamento, o impacto de incluir as desigualdades válidas para o politopo de

conjuntos independentes em $\hat{G}$ na qualidade dos limites duais, e a comparação de nossa proposta com os melhores resultados disponíveis na literatura.

Verificamos que o algoritmo de pré-processamento é extremamente eficaz no caso de instâncias cujo grafo de conflitos é denso, permitindo resolver o problema imediatamente. De fato, 19 dentre as 27 instâncias desta classe são convertidas em problemas clássicos de árvores geradoras (sem conflitos). No caso de grafos de conflitos esparsos, entretanto, nenhum avanço significativo foi constatado. O restante dos resultados se refere principalmente a este subconjunto de instâncias.

O uso de desigualdades de ciclo ímpar e clique permite fortalecer a relaxação linear das formulações (acima de 18%, em média), bem como os limites finais obtidos com *branch and cut* (acima de 12%, em média).

Finalmente, o desempenho do algoritmo proposto melhora os resultados anteriormente disponíveis de forma consistente. De fato, o próprio limite dual da relaxação linear inicial das formulações que discutimos é superior em relação aos fornecidos após horas de computação com esquemas de relaxação Lagrangeana de Zhang et al. [2011]. Também foi possível obter vários certificados de otimalidade, e cinco novos certificados de viabilidade, sendo três limites primais inéditos e duas instâncias comprovadamente inviáveis.

List of Figures

List of Tables

List of Acronyms

Acronym	*Expansion*
B&C	Branch and Cut
DCUT	Directed Cutset Inequalities
ILP	Integer Linear Programming
LP	Linear Programming
MIP (MILP)	Mixed Integer (Linear) Programming
MST	Minimum Spanning Tree
MSTCC	Minimum Spanning Tree under Conflict Constraints
OCI	Odd-Cycle Inequalities
SEC	Subtour Elimination Constraints

Contents

Chapter 1

Introduction

Disjunctions model alternative relationships, being found in different contexts that involve a selection among different options. For instance, we verify this notion in the set union operation, the logical connective $\vee$, and any proposition which imposes satisfying one contition *or* the other. To some extent, relationships of this nature are contrasted with conjunctions, where the selection must include all of the available alternatives. Analogously, examples include the set intersection operation and the logical $\wedge$ connective.

Disjunctive relations arise in many contexts of integer programming (IP). Disjunctive and multiple-choice constraints model the satisfaction of at least one among two or more inequalities in a system [Wolsey, 1998, section 1.5]. Disjunctive cuts are implemented in several mixed-integer programming solvers. Branching on disjunctions rather than on variables in a branch and cut framework provides an interesting alternative to reduce the corresponding enumeration tree, and different rules may be used to select branching disjunctions [Karamanov and Cornuéjols, 2011]. It is also important to mention that the Disjunctive Programming framework pioneered by Egon Balas in the 1970s is still relevant and has relationships with other IP techniques [Balas, 2010].

In the context of combinatorial optimization, disjunctive relations further constraint the set of feasible solutions to a problem with respect to the interaction between selected components. Usually expressed by means of *or* conditions, a natural example is the scheduling problem with incompatible jobs, which requires processing given pairs of tasks in different machines. In fact, to the best of our knowledge, a theoretical approach to the complexity of particular cases of that problem is one of the earliest references on this kind of combinatorial optimization problem [Bodlaender and Jansen, 1993].

Nevertheless, disjunctively constrained versions of classic problems in graph theory were only recently discussed in the literature. For instance, shortest paths, spanning trees and matchings under disjunctive constraints were introduced by Darmann et al.

[2009, 2011], while constrained maximum flows are investigated by Pferschy and Schauer [2011a,b]. We present a more rigorous review of related work in Section 2.2.

In such problems, the selection of some alternative pairs of edges is constrained in any feasible solution. Specifically, these are restricted according to the type of disjunctive constraints, being allowed to include:

- at most one of the edges in each pair, if under *conflict constraints*;
- at least one of the edges in each pair, if under *forcing constraints*.

The literature on disjunctively constrained problems on graphs regards mainly complexity and approximability results, but a particular interest in the minimum spanning tree problem under conflict constraints has led to the development of algorithms and benchmark instances [Zhang et al., 2011]. Its feasibility version is also discussed in the context of the quadratic bottleneck spanning tree problem [Punnen and Zhang, 2011].

In this thesis, we discuss approaches for the exact solution of the minimum spanning tree under conflict constraints (MSTCC) problem. Given an undirected graph $G(V,E)$, a set $C \subset E \times E$ of conflicting edge pairs, and weights $c : E \to \mathbb{R}$, the problem consists of finding a minimum weight spanning tree in G which does not include both edges from a pair in C. A more rigorous definition of the problem is presented in Section 2.1, and we also discuss interesting properties and related results.

To the best of our knowledge, no practical application of the problem was known as this thesis was written. Nevertheless, it is not hard to conceive conflict constraints in real-world settings where the standard MST problem arises, e.g. communication networks with different link technologies (which might be mutually exclusive in some cases), or utilities distribution networks. In fact, the latter is a standard application of the quadratic MST problem [Assad and Xu, 1992], which generalizes MSTCC, as we indicate in Section 2.1. Furthermore, although that was also the case for the matching problem under conflict constraints (introduced together with MSTCC by Darmann et al. [2011]), it was rendered an application on the thesis by Engels [2011], studying models for a freight car distribution problem arising in the logistics office of the german railway system (*Deutsche Bahn Schenker Rail*).

1.1 Main contributions

The main contribution of the work presented in this thesis is an algorithmic approach for the exact solution of MSTCC problem instances.

We introduce integer linear programming formulations building on an equivalent definition of the MSTCC problem, which introduces an auxiliary, conflict graph to represent conflict constraints (see Section 2.1). The methodology of exploiting such conflict graph to enhance formulations and design algorithms for the problem might be considered a contribution itself. The complete formulations and background information on the polyhedral representations we use are described in Chapter 3.

The solution approaches we propose include a general preprocessing method, which could be used in conjunction with any solution technique for the problem, and branch and cut algorithms. These are described in Chapter 4, along with the separation procedures we use. In particular, inequalities corresponding both to the spanning tree and the stable set polytopes are separated (the latter allows tightening the representation of conflict-free edge selections).

Experiments evaluating our implementation design and comparing our approach with the best results previously available in the literature are described in Chapter 5. Encouraging results include stronger dual bounds already for root LP relaxations, for all instances in a benchmark set. We are also able to provide several optimality certificates and new results for five instances whose feasibility was previously unknown.

With respect to scientific divulgation, the present work was communicated as follows:

***XVI IPCO*:** poster session presentation at the 16th Conference on Integer Programming and Combinatorial Optimization (March 18 – 20, 2013, Valparaíso, Chile).

***XLV SBPO*:** full paper presentation at the brazilian symposium on operations research (September 16 – 19, 2013, Natal, Brazil).

***Optimization Letters*:** paper accepted for publication on the journal by Springer (DOI: 10.1007/s11590-014-0750-x). The manuscript is also available at arXiv:1307.1424.

1.2 Notation and terminology

We describe next the basic terminology and notation relevant to this thesis. These are actually quite standard, following basically from that of Bondy and Murty [2007] for graph theory, and Grötschel et al. [1988] and Bertsimas and Weismantel [2005] for integer programming and polyhedral combinatorics definitions.

For simplicity, we may refer to an *algorithm* as "our implementation" of the algorithm, when the context suggests so.

Graph theory

All graphs in this work are finite and simple, i.e. they have neither loops nor parallel links joining any pair of vertices. Unless stated otherwise, they are also undirected, and we use n to denote $|V|$, as well as $|E| = m$. Given a graph G, we denote its set of vertices by $V(G)$, and its set of edges by $E(G)$. Therefore, we use $G(V,E)$ and $(V(G),E(G))$ interchangeably. The *degree* $d(v)$ of a vertex v in G is the number of edges incident to v. We also denote the set of edges incident to vertex i by $\delta(i)$.

We call H a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. H is thus a *proper subgraph* if the inclusion is strict in both cases. A *spanning* subgraph of G is a subgraph H such that $V(H) = V(G)$. Given a non-empty subset $S \subseteq V$, let $E(S) \subseteq E$ be the set of edges with both endpoints in S. We designate $G[S] = (S, E(S))$ as the subgraph *induced* by vertices in S. Analogously, the subgraph induced by a subset $T \subseteq E$ of edges is $G[T] = (V(T), T)$, where $V(T) \subseteq V$ is the set of ends (terminal nodes) of edges in T.

A *walk* in G is a finite, non-empty sequence alternating vertices and edges $W = v_0 e_1 v_1 e_2 v_2 \dots e_k v_k$, such that the ends of e_i are v_{i-1} and v_i, for $1 \leq i \leq k$. If the edges and the internal vertices in a walk W are distinct, W is called a *path*.

Two vertices u, v of G are connected if there is a (u,v)-path in G. The *components* of G are the subgraphs $G[V_1]$, $G[V_2]$, ..., $G[V_k]$, where $V_1, V_2, \dots, V_k$ consists of the partition of $V(G)$ such that any two vertices u, v are connected iff both u and v belong to the same set V_i. If G has only one component, we say that it is *connected*. Otherwise, G is *disconnected*. A given edge e is a *cut-edge* if the number of components of $G(V, E\backslash\{e\})$ is greater than that of G.

A *cycle* is a walk where the origin and terminus coincide, and where the edges and internal vertices are distinct. A *chord* in a cycle is an edge joining non-consecutive vertices. A chordless cycle is called a *hole*.

A connected acyclic graph is a *tree*. A *spanning tree* of G is a spanning subgraph of G that is a tree. Given a particular vertex r, an *arborescence* is a directed, rooted tree in which all edges point away from r; equivalently, for any vertex $v \neq r$, there is exactly one directed path from r to v.

A *stable set* or *set packing* $S \subseteq V$ is a set of vertices, no two of which are adjacent in G. A *matching* $M \subseteq E$ is a set of edges without common vertices. A *cut* $(U, V\backslash U)$ is a partition of the vertices of G into two disjoint subsets that are joined by at least one edge. We also define the *cutset* of the cut as $\delta(U) = \{\{i,j\} \in E : i \in U, j \notin U\}$, i.e. the set of edges whose end points are in different subsets of the partition.

A proper *vertex coloring* of a graph is a labelling of its vertices such that no two vertices sharing the same edge have the same label. A coloring using at most k colors is

called a *k-coloring*. The smallest number of colors needed to color a graph is called its *chromatic number*. Any complete subgraph in $G(V,E)$ is a *clique*, i.e. a subset of V such that every two vertices in the subset are connected by an edge. A graph is *perfect* if the chromatic number of every induced subgraph equals the size of the largest clique of that subgraph.

Given a set of conflicting edge pairs $C \subset E(G) \times E(G)$, a *conflict-free* subset $T \subseteq E(G)$ is such that at most one of the edges e_i, e_j is in T, for each $\{e_i, e_j\} \in C$. Therefore, a *conflict-free spanning tree* (arborescence) is a subgraph induced by a conflict-free set $T \subseteq E(G)$ which is a spanning tree (arborescence). Conflict-free matchings, shortest paths, etc. are defined similarly.

Polyhedral combinatorics

Let $T \subseteq E$ be the edge set of a spanning tree of $G(V,E)$. Define the incidence vector $\mathbf{x} = (x_1, x_2, \ldots, x_{|E|})$ of T as $x_e = 1$ if edge e is included in T, and $x_e = 0$ otherwise.

The spanning tree polytope of G, denoted by $P_{st}(G)$, is the convex hull of the set F_{st} of incidence vectors of spanning trees in G. That is: $P_{st}(G) = conv(F_{st})$.

Other polytopes are analogously defined. Whenever possible, we use P_{name} to denote $P_{name}(G)$, the polyhedron of interest considering a particular graph G.

A graph G is *t-perfect* if the polytope of stable sets in G is determined by the cycle-constraint relaxation (see Section 3.2).

Chapter 2

The minimum spanning tree under conflict constraints problem

This chapter formally presents the problem that we study in this thesis, starting with its precise definition in Section 2.1. The problem was introduced recently by Darmann et al. [2009, 2011], and we review the related literature in Section 2.2, also indicating recent work on different problems with similar constraints.

We remark that the problem has been discussed in the literature using different designations, including the MST problem under disjunctive constraints, and MST under conflict constraints (MSTCC). In the remaining of this text, we adopt the latter denomination.

2.1 Problem definition

Given a graph $G(V,E)$ and a set of conflicting edge pairs $C \subset E \times E$, we define in Section 1.2 a *conflict-free* spanning tree of G as a set of edges $T \subseteq E$ inducing a spanning tree of G, such that at most one of the edges e_i, e_j is in T, for each $\{e_i, e_j\} \in C$. Using the template of Korte and Vygen [2012], we state the minimum spanning tree under conflict constraints (MSTCC) problem formally as follows:

MINIMUM SPANNING TREE UNDER CONFLICT CONSTRAINTS PROBLEM

Instance: an undirected graph $G(V,E)$, a set of conflicting edge pairs $C \subset E \times E$, and weights $c : E \to \mathbb{R}$.

Task: find a conflict-free spanning tree of G of minimum weight, or decide that none exists.

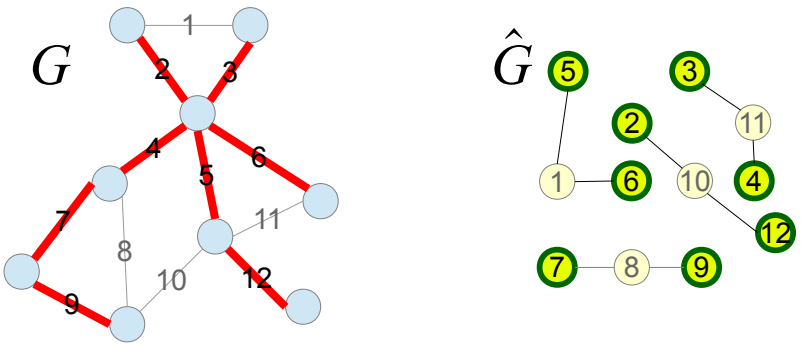

Figure 2.1. Feasible solution to a MSTCC instance: a subset of E inducing both a spanning tree in $G(V,E)$ and a stable set in $\hat{G}(E,C)$. The conflicting edge pairs are part of the input instance.

This thesis regards approaches for the solution of the MSTCC problem, and exploits largely an equivalent definition of it, using the concept of a conflict graph $\hat{G}(E,C)$. By denoting each edge in the original graph as a node in $\hat{G}$, we represent each conflict constraint by an edge connecting the corresponding nodes in $\hat{G}$. The problem is thus to find a subset of E of minimum cost, corresponding both to a spanning tree of G and to a stable set in $\hat{G}$.

Figure 2.1 illustrates a feasible solution to an instance of MSTCC, in terms of the original and the conflict graphs. We remark that the equivalent definition using this auxiliary graph was introduced with the problem [Darmann et al., 2011].

The conflict graph also allows a precise combinatorial definition of the MSTCC problem. Let $P_{st}(G)$ denote the polytope of spanning trees in $G(V,E)$ and $P_{stab}(\hat{G})$ denote the polytope of stable sets in $\hat{G}(E,C)$, as defined in Section 1.2. Without loss of generalization, assume the task is to find the optimal weight of a conflict-free spanning tree in G, defined as infinity if none exists, for consistency.

The IP formulations we present on Chapter 3 consist of particularizations of the following, general optimization problem:

$$\min \left\{ \sum_{e \in E} c_e x_e : \mathbf{x} \in P_{st}(G) \cap P_{stab}(\hat{G}) \subseteq \mathbb{B}^{|E|} \right\}$$

Finally, we note that MSTCC can also be formulated as a quadratic minimum spanning tree problem [Assad and Xu, 1992]. For instance, Zhang et al. [2011] describe a standard *big-M* modeling of conflicting edge pairs.

2.2 Literature review

Problems on optimal trees with different side constraints have been extensively studied. These include constraints on vertex degree, diameter, capacity, cardinality, among others. The chapter by Magnanti and Wolsey [1995] indicates different examples.

The MSTCC problem is relatively recent, being introduced by Darmann et al. [2009, 2011], together with other variations of problems on paths, trees and matchings under disjunctive constraints. In particular, they regard both conflict and *forcing* constraints: the latter requires solutions to include *at least* one of the edges from each pair in C. We note that feasible solutions in this case induce a vertex cover in the corresponding conflict graph.

The authors establish several results on the complexity and approximation hardness of such problems. Interestingly, they observe particular graph classes for the conflict graph $\hat{G}$ to prove that a sharp line separates between easy and hard instances: the particular case in which $\hat{G}$ consists of isolated edges can be solved in polynomial time; however, if $\hat{G}$ consists of simple paths of length two or more, the problem is strongly NP-hard. Note that no assumption is made regarding the structure of the original graph, over which a conflict-free spanning tree is sought. The result also proves that MSTCC cannot be approximated by a constant factor of the optimal value, unless P = NP.

Both theoretical and computational results on MSTCC are further described by Zhang et al. [2011]. Motivated by its application in developing algorithms for the quadratic bottleneck spanning tree problem [Punnen and Zhang, 2011], the work describes different particular cases which are polynomially solvable, and feasibility tests relating to the stable set and vertex cover problems. They present heuristics and two exact algorithms based on Lagrangean relaxation schemes. One formulation is integral, as all disjunctive constraints are relaxed and the standard MST is solved as subproblem. The other approach relaxes only part of the conflicts and solves the NP-hard maximum edge clique partitioning [Dessmark et al., 2007] as subproblem, yielding stronger dual bounds, though at higher computational costs.

Zhang et al. [2011] also introduce two sets of benchmark instances and discuss computational results for their algorithms. Nevertheless, considerably large duality gaps are associated with these algorithms, which do not provide optimality certificates. This work is the baseline of comparison for the approach we developed in this thesis, and an experimental evaluation of the algorithms we propose is described in Chapter 5 to indicate their relative performance.

Finally, we remark that conflict graphs have already been used for many years in integer programming to represent logical relations among variables [Atamtürk et al., 2000].

The standard motivation is to generate clique (or more general upper-bounding) inequalities, whether in the context of particular applications, such as in airline crew-scheduling [Hoffman and Padberg, 1993], or in polyhedral investigations of general programs with a given structure. Examples of the latter case include the study of inequality systems with only two binary variables per row by Johnson and Padberg [1982], or set packing relaxations of a series of problems on acyclic digraphs, linear orderings, cuts and multicuts, by Borndörfer and Weismantel [2000].

In a different context, conflict graphs were also studied recently to leverage SAT conflict analysis techniques to enhance computational performance of MIP solvers [Achterberg, 2007]. The author presents a succesful method to generate cutting planes from pruned nodes in the enumeration tree of a branch and cut framework.

Different problems under conflict constraints

We also note that recent papers on different combinatorial optimization problems under disjunctive constraints can be found.

The knapsack problem (KP). Yamada et al. [2002] introduce a generalization of KP with conflict constraints to model item incompatibilities. The complexity of particular cases of the problem is discussed by Pferschy and Schauer [2009].

Both exact and heuristic algorithms for the problem were proposed. Hifi and Michrafy [2007] describe a specialized branch and bound method using different variable fixation tests. An algorithm based on local branching [Fischetti and Lodi, 2003] was introduced later by Akeb et al. [2011], capable of solving larger problem instances. A two-dimensional version of the problem was also recently studied by de Queiroz and Miyazawa [2012].

Bin-packing and scheduling. Bodlaender and Jansen [1993] present the problem of scheduling with incompatible tasks. They establish complexity results on its feasibility version, while the optimization problem was presented next [Bodlaender et al., 1994], and it was proved that it generalizes the vertex colouring problem. Later, Jansen and Öhring [1997] designed approximation algorithms observing that it is also a generalization of the bin-packing problem.

A thorough literature describing both exact and heuristic algorithms for such problems is available, as Sadykov and Vanderbeck [2012] overview. Interestingly, the authors present a branch and price algorithm for the bin packing problem with conflicts, solving conflict-constrained knapsack subproblems with dynamic programming.

Other problems on graphs. As described in Chapter 1, different disjunctively constrained versions of classic problems on graph theory were introduced recently. We highlight the work of Darmann et al. [2011] on paths, trees and matchings with conflicts in the beginning of this section.

The particular case of shortest paths under conflict constraints is equivalent to the problem with forbidden vertices, which was previously studied by Kann [1993], establishing its innaproximability even in the unweighted case.

The solution for minimum cost perfect matchings under conflict constraints is approached in the work of Öncan et al. [2013], where polynomially solvable cases, a Lagrangean relaxation scheme and heuristics are presented.

Finally, the complexity of maximum flow problems under conflict constraints has been studied by Pferschy and Schauer [2011a,b]. The authors also relate these with the more classic transportation problem on networks subject to excluding constraints [Goossens and Spieksma, 2009].

Chapter 3

Integer linear programming formulations

This chapter presents integer linear programming (ILP) formulations for modeling the minimum spanning tree under conflict constraints (MSTCC) problem. Some of the formulations include exponentially-sized classes of inequalities, and corresponding separation procedures for using them in a branch and cut framework are described in Chapter 4. Section 3.1 outlines the methodology adopted in this work. Since it builds on representations of the stable set polytope, Section 3.2 briefly reviews the relaxations considered in this work. Finally, the complete formulations we introduce are described in Section 3.3.

In what follows, suppose we are given an input graph $G(V, E)$, with $|V| = n$, $|E| = m$, weights c_e associated with each edge, and a set $C \subset E \times E$ of conflicting edge pairs. Out of C, a conflict graph $\hat{G}(E, C)$ is defined, where edges of the original graph are associated with its nodes, while conflicts are associated with its edges.

3.1 Overview

The methodology in this work builds on the observation that solutions to the MSTCC problem consist of subsets of E corresponding simultaneously to a spanning tree of $G(V, E)$ and a stable set in $\hat{G}(E, C)$. Although the conflict graph $\hat{G}$ has been associated with the MSTCC problem, as introduced by Darmann et al. [2011, 2009], the existing solution approaches do not explore previous results on solving stable set problems.

This is precisely one of the key features of the exact solution algorithm to be introduced here. The ILP formulations we present are conceived exploiting different representations of two polytopes: the one of spanning trees in G (denoted by $P_{st}(G)$ in the

remainder of this chapter), and that of stable sets in $\hat{G}$ ($P_{stab}(\hat{G})$ in what follows). Note that the intersection of these spans the feasibility conditions of MSTCC.

This approach enables leveraging existing results regarding both polyhedral regions, such as valid inequalities for $P_{stab}(\hat{G})$, which we discuss and include in the proposed formulations. A fundamental point is that, on the one hand, there are several representations of $P_{st}(G)$ which are as tight as possible, i.e. they define the convex hull of incidence vectors of spanning trees in G, and, additionally, the optimization problem associated with it (MST) is well-solved (see Magnanti and Wolsey [1995]).

On the other hand, that is not the case for $P_{stab}(\hat{G})$, as the stable set problem is NP-hard. Although it has been studied for decades, and several properties and valid inequalities have been established for it [Padberg, 1973, 1979], no ideal description of P_{stab} is to be expected, unless P = NP.

We thus start with a review of $P_{stab}(\hat{G})$ relaxations that we consider in this work to denote the polyhedral region corresponding to conflict-free incidence vectors. Finally, in Section 3.3 these are combined with three different descriptions of $P_{st}(G)$ to provide MSTCC formulations. We note in advance that one of them (which uses an extended multicommodity flow formulation) is not capable of solving benchmark problem instances, in spite of different efforts to strengthen it. Nevertheless, it actually provided insight in designing the other formulations, and a brief discussion on computational results is included.

3.2 Stable set polytope relaxations

Given a stable set $\varphi \subseteq E$ in a graph $\hat{G}(E, C)$, its incidence vector $\chi^{\varphi} \in \mathbb{B}^{|E|}$ is defined as:

$$\chi_i^{\varphi} = \begin{cases} 1, & \text{if } e_i \in \varphi \\ 0, & \text{otherwise} \end{cases}$$

The $P_{stab}(\hat{G})$ polytope consists of the convex hull of incidence vectors of all stable sets in $\hat{G}$. That is:

$$P_{stab}(\hat{G}) = conv\left\{\chi^{\varphi} : \varphi \subseteq E \text{ a stable set}\right\}$$

Note that it is indeed a *polytope*, as it is embedded in the $|E|$-dimentional unit cube. To describe this region by a system of linear inequalities, we associate a variable $x \in \mathbb{B}^{|E|}$ to denote incidence vectors. Then, the combinatorial definition of a stable set yields the

simplest relaxation of $P_{stab}(\hat{G})$:

$$x_{e_i} + x_{e_j} \leq 1, \qquad \forall \{e_i, e_j\} \in C \tag{3.1}$$

$$0 \leq x_e \leq 1, \qquad e \in E \tag{3.2}$$

All integer solutions to this system correspond to incidence vectors of stable sets in $\hat{G}$: edge inequalities (3.1) constraint the selection of at most one of the vertices joined by each edge, while (3.2) is the continuous relaxation of x. Together, these are refered to as trivial inequalities, and the standard terminology for this region is simply *stable set polytope relaxation*, or $P_{rstab}(\hat{G}) = \{x \in \mathbb{R}^{|E|} : \text{ x satisfies (3.1) e (3.2) }\}$. An important result is that it yields the convex hull of stable sets in $\hat{G}$ if and only if the conflict graph is bipartite [Grötschel et al., 1988, Section 9.1].

We strengthen this representation with the intersection of two tighter polyhedra, as we describe next. The first is known as the cycle-constrained stable set polytope $P_{cstab} \subset P_{rstab} \subset \mathbb{R}^{|E|}_+$, given by (3.1), (3.2) and odd-cycle inequalities:

$$\sum_{i \in U} x_{e_i} \leq \frac{|U|-1}{2}, \quad \forall U \subset E \text{ inducing an odd-cycle in } \hat{G} \tag{3.3}$$

These are valid for P_{stab} since the cardinality of any stable set in a subset U of vertices of $\hat{G}$ inducing an odd-cycle (with or without chords) is at most $\lfloor \frac{|U|}{2} \rfloor = \frac{|U|-1}{2}$. Still, P_{cstab} yields the convex hull of stable sets in $\hat{G}$ if and only if the conflict graph is *t-perfect* [Grötschel et al., 1988, Section 9.1]. Although this is a quite restrictive condition, the separation of odd-cycle inequalities remarkably improves the quality of dual bounds for MSTCC benchmark instances, as we describe in Section 5.3.

Finally, an additional relaxation consists of the clique-constrained stable set polytope $P_{qstab} \subset P_{rstab} \subset \mathbb{R}^{|E|}_+$, described by (3.2) and clique inequalities:

$$\sum_{i \in Q} x_{e_i} \leq 1, \qquad \forall Q \subset E \text{ inducing a clique in } \hat{G} \tag{3.4}$$

The unit upper bound is clearly valid for P_{stab}, as no stable set in $\hat{G}$ could include more than one vertex from any complete subgraph. The class of graphs for which P_{stab} and P_{qstab} coincide is precisely that of perfect graphs [Grötschel et al., 1988, Section 9.2]. Note that edge inequalities (3.1) correspond to 2-cliques, and any $x \in P_{qstab}$ also satisfy them. Also, in the case of triangles in $\hat{G}$, (3.3) and (3.4) coincide.

We remark that, when they induce odd-holes (chordless odd-cycles), inequalities (3.3) might define facets of P_{stab}, and the sequential lifting procedure of Padberg [1973] could be used for that purpose. Padberg also proved that inequalities (3.4) are facet-

defining for P_{stab} if and only if the clique induced by Q is maximal. We build on this last result to include exactly those non-dominated inequalities in the model (see Section 4.3). Unfortunately, while clique inequalities traditionally had a strong impact in solving stable set problems, the impact of these in MSTCC benchmark instances was less expressive, as we evaluate in Section 5.3.

In conclusion, let P_{st} denote any representation of the polytope of spanning trees in $G(V,E)$. The MSTCC formulations we introduce in the next section consist of a particularization of $\min\{\sum_{e\in E} c_e x_e : \mathbf{x} \in P_{st} \cap P_{cstab} \cap P_{qstab} \subseteq \mathbb{B}^{|E|}\}$.

3.3 ILP formulations for MSTCC

The spanning tree polytope P_{st} has a major relevance in polyhedral combinatorics, and is arguably one of the most studied. Related results, such as those by Edmonds [1971], were very important to the development of this research field.

Several formulations of this polytope are known. Some of these are *compact* (in the sense of including a polynomial number of variables and constraints), e.g. those based on network flows, the one using Miller-Tucker-Zemlin constraints [Miller et al., 1960; Desrochers and Laporte, 1991], or the more recent one by Urrutia and Lucena [2014]. As for exponentially-sized formulations, those using subtour elimination constraints (SEC) and cutset inequalities are the most usual. Moreover, there are interesting relations of equivalence and inclusion among these, and the reader is referred to the chapter by Magnanti and Wolsey [1995] for a thorough revision on the subject.

The main results of this work correspond to formulations with SEC and a directed version of cutset inequalities, both providing the tightest possible representations of the P_{st} polytope. Actually, a different setting was considered in early exploratory experiments, using a single-commodity flow formulation, which is not tight. Unfortunatelly, as in the case for the classic MST problem, rather weaker LP relaxation bounds were verified.

Finally, we stress that, although simply appending edge inequalities (3.1) to each of the following representations of P_{st} would yield correct formulations for MSTCC, the present approach strengthens it with tighter relaxations of the stable set polytope. As in the previous discussion, define the incidence vector $\mathbf{x} = (x_1, x_2, \ldots, x_{|E|})$ of a given solution so that $x_e = 1$ if edge e is included in the solution, and $x_e = 0$ otherwise.

3.3.1 Extended multicommodity flow formulation

This section describes a first attempt to formulate MSTCC as an ILP with a polynomial number of variables and constraints. Although this formulation is not further investigated

in the remainder of the text, it is included and brief computational results are outlined here since, to some extent, it motivates the development of a cutting plane approach.

We consider a network flow f to impose connectivity and node spanning requirements on the solution. It is thus necessary to set an arbitrary root node r as flow source; we use $r = 1$. Although the model is still defined on the undirected graph $G(V,E)$, flow variables $f_{i,j}$ and $f_{j,i}$ are introduced considering flow in both directions of each edge $\{i,j\} \in E$. Moreover, there is a different commodity associated with each vertex $k \neq r$. One flow unit of each commodity k emanates from the root and must reach appropriate vertex k. Let f_{ij}^k denote the amount of commodity k flowing from i to j.

Flow directions are defined in consistency with the usual cutset $\delta(U)$ associated with a subset U of V: $\delta(U) = \{\{i,j\} \in E : i \in U, j \notin U\}$, such that $\delta(\{i\})$ denotes the set of edges incident to vertex i. Then, let $\delta^+(\{i\})$ indicate flow directions leaving vertex i, i.e. variables $f_{i,j}^k$ for all j,k. Analogously, $\delta^-(\{i\})$ indicates flow directions reaching vertex i.

The extended multicommodity flow representation of the spanning tree polytope consists of $P_{emflow} \subset \mathbb{R}_+^{|E|} \times \mathbb{R}_+^{2|E|\times(|V|-1)}$, given by (3.5) – (3.11) below:

$$\sum_{e\in\delta^-(\{r\})} f_e^k - \sum_{e\in\delta^+(\{r\})} f_e^k = -1, \quad \forall k \tag{3.5}$$

$$\sum_{e\in\delta^-(\{k\})} f_e^k - \sum_{e\in\delta^+(\{k\})} f_e^k = 1, \quad \forall k \tag{3.6}$$

$$\sum_{e\in\delta^-(\{v\})} f_e^k - \sum_{e\in\delta^+(\{v\})} f_e^k = 0, \quad \forall k, v \notin \{r,k\} \tag{3.7}$$

$$\sum_{e\in E} x_e = |V| - 1 \tag{3.8}$$

$$f_{ij}^k + f_{ji}^{k'} \leq x_e, \quad \forall k,\ k',\ e = \{i,j\} \in E \tag{3.9}$$

$$f_{ij}^k \geq 0 \text{ and } f_{ji}^k \geq 0, \quad \forall k,\ \{i,j\} \in E \tag{3.10}$$

$$0 \leq x_e \leq 1, \quad e \in E \tag{3.11}$$

Equations (3.5) to (3.7) are flow balance constraints on every commodity k. This connectivity requirement, together with the selection of $n-1$ edges in equation (3.8), ensures the solution is a spanning tree. Inequalities (3.10) restrict flow values to be non-negative, and (3.11) is the continuous relaxation of binary variables for selecting edges.

Although the formulation includes variables $f_{i,j}^k$ and $f_{j,i}^k$, bidirectional flow inequalities (3.9) are shown to correctly bind flow of different commodities in a given edge to the same direction, as Magnanti and Wolsey [1995] show. Moreover, they prove that this formulation is tight (providing the convex hull P_{st} of incidence vectors of spanning trees of G in the LP relaxation) only if this class of $O(mn^2)$ inequalities is included.

Now, by projecting P_{emflow} on $\mathbf{x}$ variables and taking the intersection with a relaxation of P_{stab}, a formulation for MSTCC is provided. To keep the number of constraints polynomial in the input size, one would start with the simplest relaxation with edge inequalities (3.1), though alternatives were considered (see the end of this section). That is:

$$\min \left\{ \sum_{e \in E} c_e x_e : \mathbf{x} \in proj_x(P_{emflow}) \cap P_{rstab} \subseteq \mathbb{B}^{|E|} \right\} \tag{3.12}$$

Preliminary experiments were performed with both this and the analogous single commodity formulation (which is not tight, including fractional vertices), described by Zhang et al. [2011]. On the one hand, none of these is able to solve MSTCC benchmark instances in less than dozens of hours of computation with CPLEX 12.5.1, in a platform with an Intel Core i7 980 (3.33GHz) CPU and 24GB of RAM. E.g., one of the smallest instances ($|V| = 50$, $|E| = 200$, $|C| = 398$) takes over 13 hours to be solved and, just by replacing the conflict graph so that $|C| = 597$, it takes over 55 hours. In fact, only two *type 1* instances (problems which are not trivial) could actually be solved within two days of execution.

On the other hand, the results with an alternative set of problem instances yield limited intuition. A set of 56 instances was constructed, where graphs $G(V, E)$ are smaller, with $|V| \in \{10, 25, 50\}$, and conflict graphs $\hat{G}(E, C)$ are yet sparser, with $|C| \leq \frac{15}{100} \times m$ (while the benchmark ones have $|C| \leq \frac{15}{100} \times \frac{m(m-1)}{2}$, i.e. up to 15% of the maximum density of $\hat{G}$). In this case, the single commodity formulation yields much worse dual bounds than its multicommodity counterpart; note that the same conclusion holds for the classic MST problem [Magnanti and Wolsey, 1995]. While the former requires a large number of nodes in the enumeration tree to verify an optimal solution, the latter can solve several of these smaller instances in the root LP relaxation.

We considered that, using a decomposition strategy with tight formulations equivalent to P_{emflow}, would enable solving large problem instances by generating the model dynamically, as opposed to loading the complete formulation *a priori*.

Finally, different strategies for better approximating P_{stab} in (3.12) were evaluated, none of which had any impact worthy of notice. This included making triangle inequalities (either 3-cliques in (3.4) or 3-cycles in (3.3)) explicit, their lift to maximal cliques, and the complete enumeration of maximal clique inequalities.

3.3.2 Undirected formulation with subtour elimination constraints

Let $P_{sec} \subset \mathbb{R}_+^{|E|}$ denote the representation of the spanning tree polytope with subtour elimination constraints (SEC), given by:

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad \forall S \subset V, S \neq \emptyset \tag{3.13}$$

$$\sum_{e \in E} x_e = |V| - 1 \tag{3.14}$$

$$0 \leq x_e \leq 1, \qquad e \in E \tag{3.15}$$

While SEC (3.13) enforce a cycle-free condition, since any connected subgraph on S inducing a cycle has at least $|S|$ edges, a feasible solution is guaranteed to be a spanning tree of G by picking $|V| - 1$ edges (3.14). Constraints (3.15) correspond to the continuous relaxation of binary variables x_e.

An important result by Edmonds [1971] is that the above formulation is tight, as all its vertices are integer-valued. Since there are exponentially-many inequalities (3.13), they could only be included *a priori* in a model for very small-sized instances. An usual approach in practice is thus to generate them dynamically, as they are violated by solutions to LP relaxations of the formulation. Section 4.4 describes the standard separation procedure implemented for this purpose in the corresponding algorithm.

This representation is used to formulate MSTCC as the following ILP, to which we refer as the *undirected formulation*:

$$\min \left\{ \sum_{e \in E} c_e x_e : \mathbf{x} \in P_{sec} \cap P_{cstab} \cap P_{qstab} \subseteq \mathbb{B}^{|E|} \right\} \tag{3.16}$$

3.3.3 Directed formulation with cutset inequalities

Alternatively, a formulation with directed cutset inequalities (DCUT) is also investigated. In this case, we consider the directed graph $G'(V, A)$, with $A = \{(i, j) \cup (j, i) : \{i, j\} \in E\}$, and both arcs have the same cost $c_{i,j}$ as the corresponding edge in G. The problem is thus to find a conflict-free minimum spanning arborescence of G'. This requires setting a particular vertex r as the arborescence root. Magnanti and Wolsey [1995] show that this formulation is symmetric with respect to the selection of the root vertex, i.e. the LP relaxation bound for the classic minimum spanning arborescence problem is independent of the choice for r. We use $r = 1$ in this work; further research could investigate the impact of alternative choices.

The directed formulation $P_{dcut} \subset \mathbb{R}_+^{2|E|}$ consists of:

$$\sum_{a \in \delta^+(U)} y_a \geq 1, \qquad \forall U \subset V, r \in U \tag{3.17}$$

$$\sum_{a \in \delta^-(v)} y_a = 1, \qquad \forall v \in V \backslash \{r\} \tag{3.18}$$

$$\sum_{a \in \delta^-(r)} y_a = 0, \tag{3.19}$$

$$\sum_{a \in A} y_a = |V| - 1, \tag{3.20}$$

$$0 \leq y_a \leq 1, \qquad a \in A \tag{3.21}$$

Cutset inequalities (3.17) guarantee the selection of a connected subgraph, establishing that every component (induced by a proper subset of vertices U containing the root) has at least one arc leaving it. As in the case of SEC (3.13), this class of inequalities is exponentially sized, and the corresponding separation procedure is described in Section 4.4.

Constraints (3.20) and (3.21) are analogous to the undirected case. Equalities (3.18) impose that exactly one arc must reach every non-root node, while (3.19) excludes from the solution all those arcs directed into the root node. Both constraints are actually implied by (3.17) and (3.20), but including them explicitly makes the solution of the LP relaxation by the cutting-plane approach more efficient. As less iterations of the separation procedure for DCUT are performed, the relaxation is completed much faster and the resulting model is smaller.

In consistency with the discussion in Section 3.2, the representation of conflict-free arborescences considers the projection $\pi : \mathbb{R}_+^{2|E|} \mapsto \mathbb{R}_+^{|E|}$, mapping $\mathbf{y}$ back into the space of undirected variables by $\pi(\mathbf{y})_{\{i,j\}} = y_{(i,j)} + y_{(j,i)}$. Note that $y_{(i,j)} + y_{(j,i)} \leq 1$ is also implied by (3.17) and (3.20). Moreover, Magnanti and Wolsey [1995] show that both formulations for P_{st} are equivalent, i.e. P_{sec} and $proj_x(P_{dcut})$ describe precisely the same polyhedral region, as tightly as possible.

Finally, we may formulate MSTCC by:

$$\min \left\{ \sum_{a \in A} c_a y_a : \mathbf{y} \in P_{dcut} \subseteq \mathbb{B}^{2|E|}, \pi(\mathbf{y}) \in (P_{cstab} \cap P_{qstab}) \subseteq \mathbb{B}^{|E|} \right\}, \tag{3.22}$$

3.4 Final remarks

This chapter discussed several results concerning polyhedral representations of interest for the development of exact algorithms for the MSTCC problem, including descriptions

of the spanning tree polytope, as well as different relaxations of the stable set polytope. The intersection of these provides ILP formulations for MSTCC, which in turn are suitable for a cutting plane approach.

While this work was rather successful in using these formulations to improve on the previous literature results, as described in Section 5.4, a more thorough problem-specific analysis of the MSTCC polytope would also be interesting. For instance, one could investigate the form of valid inequalities, whether also valid for the stable set relaxation or not. It would be interesting to know under which conditions are odd-hole and maximal clique inequalities also facet-defining for the new polytope.

While such issues are interesting in their own right, the algorithmic approach may require enhanced formulations and to leverage more established techniques from the stable set literature. In this sense, a natural extension would be to consider the edge projection technique to separate rank inequalities, a large class which generalize both odd-cycle and clique inequalities [Rossi and Smriglio, 2001; Rebennack et al., 2012].

We also note that Rebennack et al. [2012] describe different classes of inequalities for P_{stab}, and offers a tutorial introduction on branch and cut approaches for the stable set problem. The current work built on their description of the separation procedure for odd-cycle inequalities.

Finally, a more rigorous description of the issues discussed in Section 3.2 is available in the book by [Grötschel et al., 1988, Chapter 9], along with more in-depth results on the stable set polytope.

Chapter 4

Exact algorithms for MSTCC

This chapter introduces an approach for the exact solution of the MSTCC problem. This approach includes two main phases. First, a preprocessing method, described in Section 4.2, attempts to reduce the instance size and probes problem variables for possible fixations. That is followed by a branch and cut phase, where a model with exponentially-many constraints corresponding to one of the formulations presented in Sections 3.3.2 or 3.3.3 is solved, i.e. either the undirected formulation (with subtour elimination constraints) or the directed one (with cutset inequalities).

The remaining algorithms are presented next. The general methods for each formulation are outlined in Section 4.3. The procedures for solving separation problems with respect to each class of inequalities are at the core of each algorithm; these are described in Section 4.4. Finally, we give particular implementation details and design choices in Section 4.5, followed by general remarks.

4.1 Overview

As an enhancement to the solution process, MSTCC problem-specific feasibility conditions were considered to devise the preprocessing algorithm described in the next section. It is worth remarking that, although designed in the context of our branch and cut approach, the algorithm may be integrated to any solution technique for the problem.

A branch and cut framework is the general method we specialize to solve the discussed ILP formulations with the dynamic generation of cutting planes. The reader is referred to the book by Applegate et al. [2007] for a recent and thorough presentation of the computational aspects of the method. The corresponding separation algorithms used in this work are exact. Moreover, an important feature is that, even though the clique sepa-

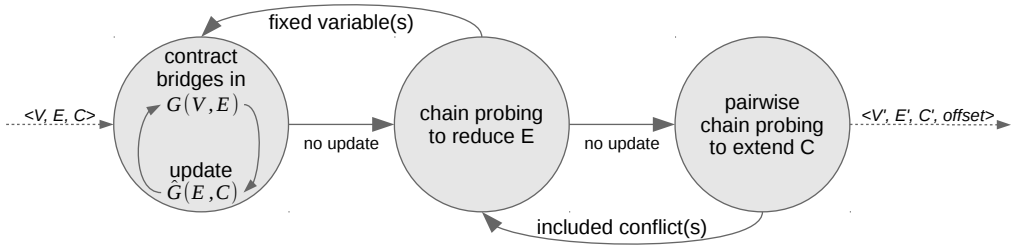

Figure 4.1. Three steps of the preprocessing algorithm, given input graphs $G(V,E)$ and $\hat{G}(E,C)$. Both a contraction in the first step and a removal in the second implies fixing edge variables and updating the corresponding conflicting pairs in C. The output includes the objective function offset due to contracted edges.

ration problem is itself NP-hard, we were able to identify and include in our formulations the subset of non-dominated inequalities in this class *a priori* (see Section 4.3).

Finally, this chapter was written regarding reproducibility matters. The exposure detail is such that a reader (with access to the corresponding literature references) could implement the described algorithms, execute similar experimental evaluations and derive the same general conclusions [Johnson, 2002].

4.2 Preprocessing algorithm

The preprocessing phase is the first one in the solution approaches developed in this thesis, and it is an important factor for their overall sucess (as described in Section 5.2). Also, it is an essential part of the problem-specific analysis of the present contribution. Figure 4.1 depicts the overall method, which is described next.

Let a MSTCC input instance consist of the original graph $G(V,E)$ and the conflict one $\hat{G}(E,C)$. The general algorithm is a three-phase iterative process, where each phase is executed as long as the problem instance is updated. To every step fixing an edge in G corresponds an update in the conflicting pairs in $\hat{G}$.

The first phase checks for cut-edges (bridges) in G, using depth-first search. As long as the original graph is connected, any cut-edge e_1 is contracted, its cost c_{e_1} is added as an offset to the optimal value of the reduced problem (if any), and conflicting pairs are removed both from G and $\hat{G}$, i.e. we can fix variables corresponding to e_k to zero, for all $e_k \in E$ such that $\{e_1, e_k\} \in C$. If at any point we verify that G is not connected, the original problem is infeasible; on the other hand, if the resulting graph is a conflict-free tree, it is also the unique feasible solution to the problem. Figure 4.2 illustrates the execution of this phase; note that the resulting problem instance has a unique solution.

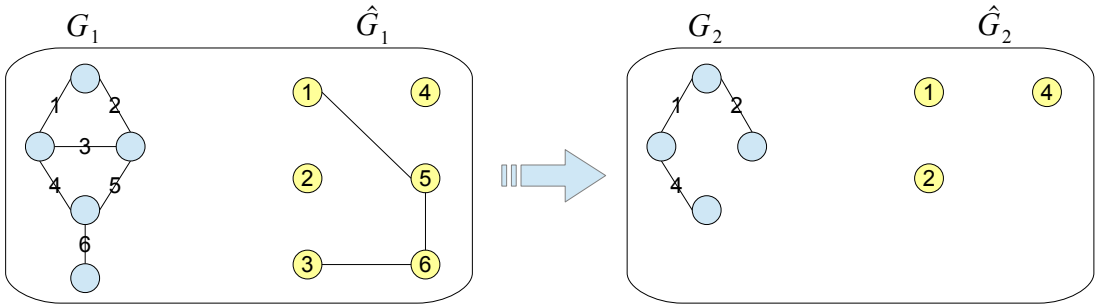

Figure 4.2. Example of the first preprocessing phase, given input (original) graph G_1 and conflict graph $\hat{G}_1$. Since cut-edge 6 is necessary for the graph to be connected, it must be selected in any spanning tree. After contracting it, we remove the conflicting ones (edges 3 and 5) from G_1, as well as the corresponding nodes in the conflict graph $\hat{G}_1$. The algorithm continues checking for other cut-edges, and recognizes that the resulting graph G_2 is a conflict-free tree and, therefore, the unique feasible solution.

The remaining phases use the *probing* technique (i.e. evaluating the consequences of possibly setting a binary variable to one of its bounds), based on implications from feasibility conditions, as Atamtürk et al. [2000] denote. Nevertheless, in the context of that work it means analyzing the structure of a general IP to derive infeasibility implications, while our next steps analyze the combinatorial structure at hand: any solution is required to induce a connected subgraph of G, and conflicting edge pairs might render that infeasible after tentatively fixing variables in chain.

In the second phase, we check the connectivity of subgraphs of G including a given edge e (with degree in the conflict graph $\delta_{\hat{G}}(e) > 0$). If the chain removing conflicting pairs and fixing any cut-edges possibly implied by the selection leads to a disconnected graph, we may remove e from E and the corresponding conflicts from C. In this case, we return to the first phase as G might include new cut-edges. Figure 4.3 includes a sample execution of this phase. Note that no decision could be draw from probing edge 1 if it does not imply any cut-edge.

Finally, if no edge could be fixed in the previous step, a third phase performs a similar evaluation on the connectivity of G, now probing pairs of variables. The chain starts fixing in the solution edges e_1 and e_2 (neither already in conflict with each other nor both conflict-free in $\hat{G}$), and proceeds by removing conflicting ones and including any cut-edges implied by the selection. Now, if G would become disconnected, the new conflict pair (e_1, e_2) is included in C, and we may return to the second phase to check if it is possible to remove any edge. Moreover, if that is the case, we return to the first phase. See Figure 4.4 for a sample iteration of this phase.

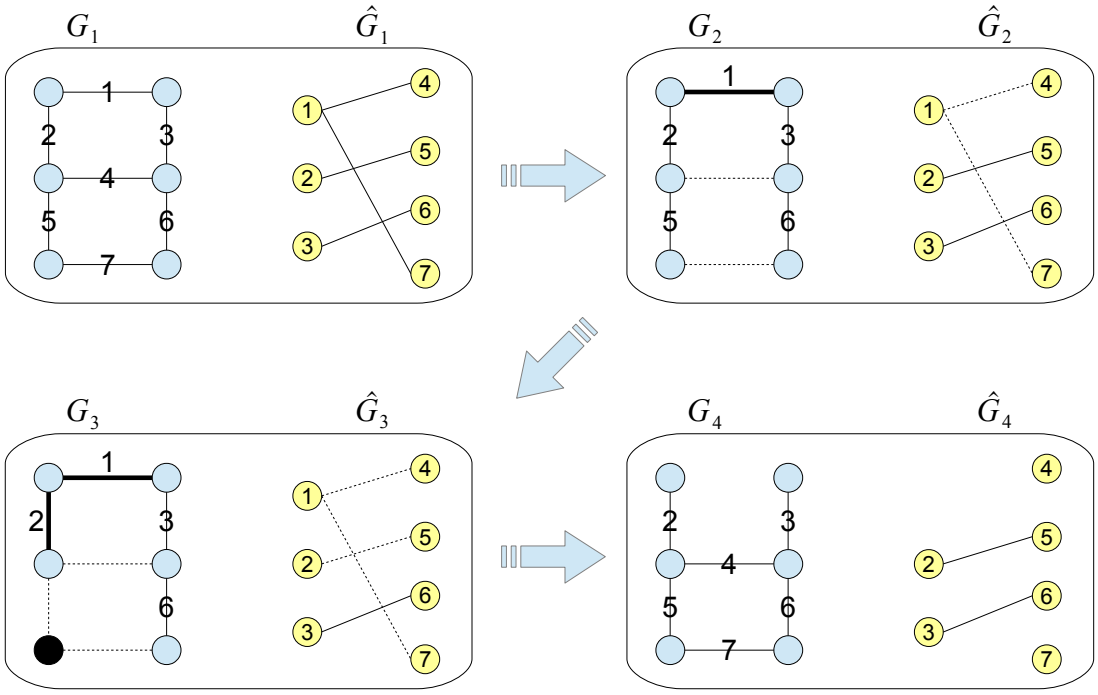

Figure 4.3. Example of the second preprocessing phase, given input (original) graph G_1 and conflict graph $\hat{G}_1$. We probe the instance fixing edge 1 in the solution, which implies removing 4 and 7. Checking for cut-edges, we have implied that 2 (any other edge, in fact) would also be selected. Since the conflict graph includes $\{2,5\}$, we would remove edge 5, and the graph would have another component (the node in black is disconnected from the others). This means that our original probe is wrong: no spanning tree includes edge 1, and we may remove it from the input. The algorithm proceeds by probing new edges.

If an iteration is completed without any update on the third phase, the reduced instance and the objective value offset are output. Note that, if any conflict-free spanning tree of G is obtained during the last two phases, we may store it as a primal feasible solution.

Since all the three phases perform a number of iterations parameterized by the current cut-edges count, to discuss the asymptotic complexity of the procedure would require a specialized average-case analysis, which is beyond the scope of this work. One could point independently that one iteration of the first phase has cost in $O(|V|+|E|)$, that the second phase perform $O(|E|)$ probing chains, while the third performs $O(|E|^2)$. Nevertheless, a worst-case bound on the behavior of the algorithm is of minor significance. In practice, however, we observe that it is sufficiently fast for preprocessing MSTCC benchmark instances, with execution time below one minute in a platform with an Intel Core i7 980 (3.33GHz) CPU.

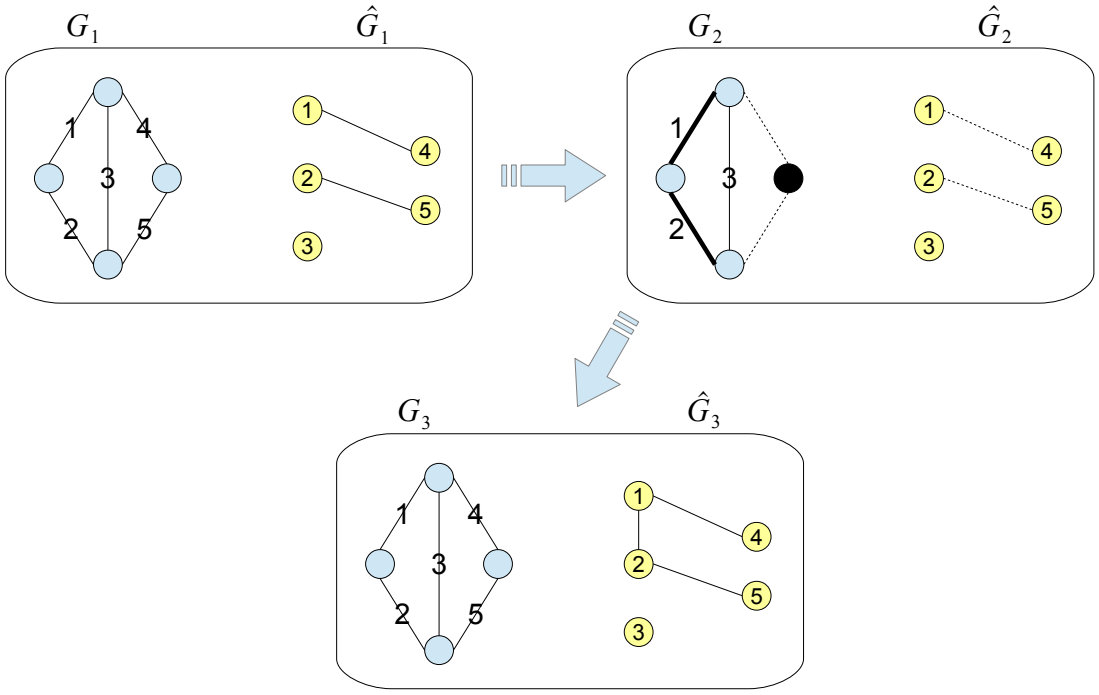

Figure 4.4. Example of the third preprocessing phase, given input (original) graph G_1 and conflict graph $\hat{G}_1$. We probe the instance fixing the pair of edges 1 and 2 in the solution, which implies removing 4 and 5, as they are in conflict. Since this would disconnect the input graph (the node in black would become isolated), no feasible solution can select both edges 1 and 2 simultaneously. I.e. they are in conflict and $\{1,2\}$ can be made explicit in the conflict graph.

4.3 Branch and cut algorithms

After preprocessing the problem instance with the algorithm described in Section 4.2, we propose a branch and cut approach for MSTCC, solving either (3.16) or (3.22) with the generation of cutting planes corresponding to violated odd-cycle inequalities (3.3) and either SEC (3.13) or DCUT (3.17), respectively.

Both formulations consider the intersection with the P_{qstab} polytope, constrained by the exponentially-sized class of clique inequalities (3.4). However, the separation problem associated with it is NP-hard, in general. In fact, the optimization problem over P_{qstab} itself is NP-hard [Grötschel et al., 1988, Section 9.2]. Nevertheless, we verified that conflict graphs in the set of challenging benchmark instances for MSTCC used in the literature have a limited number of maximal cliques: except for one instance, $\hat{G}(E,C)$ has less than $|C|$ maximal cliques, leading to a smaller model than using edge-inequalities (3.1). In fact, after the preprocessing phase, that was the case for the complete benchmark set.

It was therefore possible to successfully use the maximal clique enumeration algo-

rithm of Tomita et al. [2006] to actually include the corresponding (non-dominated subset of) maximal clique inequalities *a priori*. It is extremely fast in practice, with negligible runtime for MSTCC instances. The authors also prove that the worst-case complexity of the algorithm ($O(3^{m/3})$, in an m-vertex graph) is optimal with respect to m, since there can be at most $T(m) \leq 3^{m/3}$ maximal cliques.

The methodology proposed in the current work might be effective when $T(m)$ is in $O(|C|)$. Alternatively, when using one of these formulations to solve a different instance set, which renders the enumeration method infeasible, one could use a greedy heuristic to: (i) replace edge-inequalities by any maximal clique containing it; (ii) lift any violated triangle identified during the separation procedure for odd-cycle inequalities to a larger clique (Rebennack et al. [2012] suggests that it could be effective in the context of stable set instances).

Now, the algorithms starts as follows:

- in the case of the undirected formulation (3.16) we solve $\min\left\{\sum_{e\in E} c_e x_e\right\}$ subject to (3.14), (3.15) and (3.4). Let $\mathbf{x}$ be the solution to such linear program (LP). Clearly, if $\mathbf{x}$ is integral and a depth-first search from any vertex $i \in V$ reaches every other vertex in $V\backslash\{i\}$, then $\mathbf{x}$ is also the optimal solution of the original ILP.

- in the case of the directed formulation (3.22), we solve $\min\left\{\sum_{a\in A} c_a y_a\right\}$ subject to (3.18), (3.19), (3.20) and (3.4). Let $\mathbf{y}$ be the solution to such LP. Clearly, if $\mathbf{y}$ is integral and there is a directed path from the root r to every other vertex in $V\backslash\{r\}$, then the projection of $\mathbf{y}$ into the x variable space by $x_{\{i,j\}} = y_{i,j} + y_{j,i}$ is also the optimal solution of the original ILP.

Otherwise, we search for constraints violated by the current solution, which strengthen the relaxed polyhedron: we check both for odd-cycle inequalities (3.3), as well as SEC (3.13) in the undirected formulation, or DCUT (3.17) in the directed case. This is performed by the separation procedures we describe in the next section. If any procedure is able to separate the current solution, we add the corresponding cuts globally, and solve the new, reinforced LP. If both separation procedures fail to find a violated inequality, we branch on variables and iterate.

Finally, note that there are several strategies for selecting which cuts to include in the formulation, once different cutting planes were found by the separation procedures. We discuss four alternatives and compare their behavior in Section 4.5.

4.4 Separation procedures

The following is a description of the algorithms for solving the exact separation problems for SEC (3.13), DCUT (3.17), and odd-cycle inequalities (3.3). Apart from details of the current implementation, these are actually quite standard, and the reader is referred to classic expositions.

Subtour elimination constraints and directed cutset inequalities

The algorithms for both procedures are very similar, and are described next in conjunction. Magnanti and Wolsey [1995] describe a standard algorithm to look for violated SEC in a solution $\mathbf{x}$ to the relaxation of P_{sec}. First, a directed network corresponding to $\mathbf{x}$ is built, with the capacity of both arcs (i, j) and (j, i) set to the current value of edge variable $x_{i,j}$. We also set an arbitrary vertex as root r; we use $r = 1$. Now, $\mathbf{x}$ satisfies all subtour elimination constraints if and only if we can send one unit of flow from r to every other vertex in the capacitated network.

Therefore, by performing $|V|-1$ maximum flow (minimum cut) computations, from r to every vertex $i \in V \setminus \{r\}$, we may check in polynomial time if $\mathbf{x}$ is feasible in P_{sec}: if the value of any minimum cut is less than 1, we have found a violated inequality.

The same procedure applies to DCUT, with a slightly simpler implementation. In particular, the network and root node are already defined in advance in this case. Also note that, after finding a minimum cut $(S, V \setminus S)$, the violated DCUT inequality is already defined, while for SEC it is neccessary to check whether the cutset S or its complement yield the violated constraint.

To find a minimum (r, i) cut, we use an implementation of the highest-label preflow-push algorithm of Goldberg and Tarjan [1988], available in the open-source Library for Efficient Modeling and Optimization in Networks (LEMON, see Dezső et al. [2011]). Ahuja et al. [1997] present a comparative study of different implementations of the preflow-push algorithm. They indicate that this version has the best worst-case complexity while simultaneously having the best empirical performance. The implementation in LEMON has a worst-case time-complexity in $O(|V|^2\sqrt{|E|})$; therefore, the current implementation of both separation procedures executes in time proportional to $O(|V|^3\sqrt{|E|})$.

Odd-cycle inequalities

Gerards and Schrijver [1986] introduce an exact separation procedure for odd-cycle inequalities, which is clearly described in the tutorial of Rebennack et al. [2012]. Much like

the separation of SEC, we may check in polynomial time if every odd-cycle inequality is satisfied by computing a minimum cost cycle in an auxiliary graph.

We start by defining a new weight function w for adjacencies in the conflict graph $\hat{G}(E,C)$: let $w(u,v) = \frac{(1-x_u-x_v)}{2}$ for each $\{u,v\} \in C$. Since constraints (3.1) and (3.2) are satisfied *a priori*, we have that $w : C \to [0, \frac{1}{2}]$. We also construct an auxiliary bipartite graph H by duplicating $\hat{G}$ as follows: H has two vertices u^+ and u^- for each $u \in E$, as well as edges $\{u^+, v^-\}$ and $\{u^-, v^+\}$ for each $\{u,v\} \in C$, both with weight $w(u,v)$.

Then, for each $u \in E$, we compute a shortest (u^+, u^-) path in the auxiliary graph H. Note that, as the weight function w is non-negative, we may use Dijkstra's algorithm, stopping its execution as soon as the goal vertex u^- is selected. By the construction of H, the vertices u^+ and u^- are in different sets of the bipartition, implying that the path has odd length. By omitting the $^+$ and $^-$ indices, the path corresponds to a closed odd-walk in $\hat{G}$. However, this walk might include repeated nodes and edges, since the shortest path is determined in H; in fact, H might as well not be connected. An odd-cycle is possibly retrieved after removing such repetitions, by inspecting the vertices in the sequence. Note that the remaining sequence may not be a closed walk, and the shortest path computation for this u yields no cycle in $\hat{G}$ for the current solution $\mathbf{x}$.

The weight of any such odd-cycle $U \subset C$ in the conflict graph, disregarding any chords it might have, is $w(U) = \sum_{\{i,j\} \in U} w(i,j) = \sum_{\{i,j\} \in U} \frac{(1-x_i-x_j)}{2} = \frac{|U|}{2} - \frac{1}{2}\sum_{\{i,j\} \in U}(x_i + x_u) = \frac{|U|}{2} - \sum_{i \in V(U)} x_i$, where $V(U) \subseteq E$ denotes the set of nodes induced by U. That is, $\sum_{i \in V(U)} x_i = \frac{|U|}{2} - w(U)$, implying that $\mathbf{x}$ violates the corresponding odd-cycle inequality $\sum_{i \in V(U)} x_i \leq \frac{|U|-1}{2} = \frac{|U|}{2} - \frac{1}{2}$ if and only if $w(U) < \frac{1}{2}$.

In the present implementation, the shortest path algorithm uses a binary heap, with worst-case time complexity in $O(|E| \times \lg(|V|))$. Therefore, the separation procedure executes in time proportional to $O(|V| \times |E| \times \lg(|V|))$.

4.5 Implementation details

The following is a series of details and design choices regarding the implementation of separation procedures and the enumeration of maximal cliques, thus completing the description of the proposed algorithms.

Separating integral solutions. It is important to highlight that, in the special case of an integral solution $\mathbf{x} \in \mathbb{B}^{|E|}$ (or $\mathbf{y} \in \mathbb{B}^{2|E|}$, in the directed formulation), there can be no violated odd-cycle inequality, since the edge inequalities (3.1) guarantee a stable set in $\hat{G}$.

Hence, we try to find a violated SEC or DCUT, according to the adopted formulation. In this case, we need reduced asymptotic complexity than in the fractional case: $\mathbf{x}$ is feasible in P_{sec} (analogously, $\mathbf{y}$ in P_{dcut}) if and only if the corresponding edges induce exactly one connected component in G, which we check in $O(|E|)$ time with DFS. If there are different components, inspecting them yields the subtour (or violating cutset) and the inequality to add, and we terminate the procedure without considering the different algorithms to separate a fractional solution.

Retrieving cycles from a closed walk in OCI separation. The description in Section 4.4 includes a step to retrieve odd-cycles in the conflict graph from a closed walk in an auxiliary, bipartite graph. While it is a simple task of inspecting the walk, one must take care to keep the linear complexity of this step.

The intuition is basically to check the walk sequentially and, as soon as a repeated vertex is identified, to remove (and store) the corresponding cycle.

Improving OCI separation. Rebennack et al. [2012] also indicate some implementation tweaks for the separation of odd-cycle inequalities. Among those, we include two simple and effective adjustments in our approach.

First, the auxiliary graph H can be greatly reduced by removing nodes u^+ and u^- whenever x_u is integer, since no odd-cycle including u could yield a violated constraint, as we explain next. Suppose U is an odd-cycle in $\hat{G}(E,C)$, and that it includes $j \in E$, with $x_j = 0$. Now, $\sum_{i\in U} x_i = \sum_{i\in U, i\neq j} x_i \leq \frac{|U\setminus\{j\}|}{2} = \frac{|U|-1}{2}$, where the inequality holds because every two consecutive nodes can contribute at most 1 to the sum in the left hand side, provided edge inequalities (3.1) are satisfied. Similarly, $x_j = 1$ implies two neighbors in U with null value, and the above argument applies.

The second refinement regards the case of $\mathbf{x}$ such that $x_u + x_v = 1$ for a given $(u,v) \in C$. The definition of the weight function w provides that both edges (u^+, v^-) and (u^-, v^+) in H would have null cost. It is more interesting, though, to add a small weight ϵ instead, to avoid unnecessary vertices in the shortest path. We use $\epsilon = 10^{-6}$, as the authors suggest.

Strategies for cut selection and inclusion. Preliminary experiments indicated that different strategies for reinforcing the relaxed polyhedron when separating the current solution have a major impact on computational performance. Standard strategies evaluated in this work include returning as soon as a first cut is found, looking for the most violated inequality, or including all violated cuts.

The best overall results were verified with an alternative strategy looking for *some* of the best cuts: we include not only the most violated inequality, but also others which are close enough to being orthogonal to it. Pilot studies suggested accepting hyperplanes with an inner product of 0.1 or less. This enhanced strategy seems to balance the strength and diversity of included cuts, allowing to solve the LP relaxation in time similar to that of including all violated cuts, while limiting the model size. Alternative, more robust strategies are proposed in different works, e.g. Lucena and Resende [2004] and Koch and Martin [1998], both in the context of different Steiner problems in graphs

For completeness, details of the corresponding evaluations are as described next:

- Requiring hyperplanes to be orthogonal (to have a null inner product) seems too strict if we are interested in including a considerable fraction of the cuts generated in a single execution of the separation procedure. Therefore, we introduce a tolerance ϵ for accepting hyperplanes as orthogonal, defined by comparing the average ratio of included cuts to the total number of cuts generated, using both the directed and undirected formulations. After evaluating $\epsilon \in \{0, 0.1, 0.2, \ldots, 0.9\}$, it was possible to notice a major increase in the average ratio of included cuts when using $\epsilon \geq 0.2$. Therefore, $\epsilon = 0.1$ was adopted.
- Next, the time required to solve the initial LP relaxation using each strategy was compared. Evaluating both formulations, there is a remarkable difference in the total relaxation time if a single cut is included in each execution of the separation procedures (either the first found or the most violated one), or if several are considered at once (either all cuts found or those close to being orthogonal to the most violated one). The average execution time for fifteen type 1 instances (with $|V| \in \{100, 200\}$) using each strategy is presented in Table 4.1. We decided to use the enhanced strategy with orthogonal cuts, as its performance is comparable to the strategy of including all cuts, while restricting the model size – another driver of the overall performance in a branch and cut setting (on average, 40% less cuts are included, per execution of the separation procedures).

Efficient enumeration of maximal cliques. The algorithm of Tomita et al. [2006] is quite simple to implement but, since it is a recursive procedure based on depth-first search, it is remarkably output-sensitive. The authors indicate this behavior and suggest alternatives for iteratively storing the cliques. In the current algorithm, the corresponding incidence vector is iteratively updated (technically, an object repre-

Table 4.1. LP relaxation execution time with different cut selection strategies.

Strategy	Average time (s)
First cut only	142.1
Most violated cut only	154.1
Most violated and orthogonal cuts	8.7
All violated cuts	4.5

senting it is a parameter passed by reference) and, when a maximal clique is determined, the corresponding inequality is appended to the model.

Furthermore, the algorithm depends on efficient set intersection and union operations. In this implementation, a bitset representation of the adjacency lists of the graph is used. Finally, it is worth remarking that the description of the algorithm by Cazals and Karande [2008] is rather simpler and easier to follow.

4.6 Final remarks

Together with Chapter 3, this one completes the description of the methodology adopted in this work. In summary, a general, two-phase solution method for MSTCC was developed, with preprocessing and branch and cut algorithms. The latter builds on separation procedures for SEC, DCUT and OCI, as well as an enumeration technique for maximal clique inequalities.

Although some exploratory experiments were performed with different branching configurations (e.g. adjusting variable priorities, using strong branching or tree node selection options), no conclusion could be draw thereof. It could be interesting to leverage more established techniques from the stable set literature, e.g. the balanced branching rule of Balas and Yu [1986].

It would also be interesting to extend the separation procedure for OCI as Rebennack et al. [2012] suggests. If a step to remove chords from the cycles is included, odd-hole inequalities could be separated (while keeping the exactness of the procedure).

Some of the design choices focus into achieving better computational performance. In this sense, further research could include an evaluation of tailing-off effects in optimization, such as described by Padberg and Rinaldi [1991]. Also, note that all the separation procedures described in this work are suitable for parallel implementations, with different maximum flow (in the case of SEC and DCUT) and shortest paths (for OCI) computations being performed simultaneously in a multicore platform. Finally, the shrinking heuristic of Padberg and Grötschel [1985] to reduce the network size could be used to speed up the maxflow algorithm; for instance, it is rather effective in the context of the

branch and cut algorithm of Lucena and Beasley [1998] for the Steiner problem in graphs.

Chapter 5

Computational experiments

This chapter completes the main body of contributions in this dissertation, describing a series of computational experiments involving the use of the algorithms proposed in this study. Section 5.1 outlines the experimental design and objectives. The next two sections consider the isolated impact of the algorithm building blocks, discussing issues such as: how effective is the preprocessing algorithm? How does separating OCI or including maximal clique inequalities *a priori* helps to improve dual bounds for MSTCC?

Finally, Section 5.4 presents a direct comparison of the proposed formulations against the best results previously available in literaure. The baseline is the work by Zhang et al. [2011], and we indicate how stronger the solution bounds provided by the current approach are, and claim that it is the best known algorithm for the MSTCC problem.

5.1 Overview

The main contribution of this work is the approach for the exact solution of MSTCC. The goals of the computational evaluation we present are thus twofold: to assess the impact of the main design decisions on the computational performance of our implementation, and to indicate how stronger the bounds it provides are as compared to those of Zhang et al. [2011].

With respect to design choices, we refer mainly to the methodology of observing the conflict graph representation of conflicting edge pairs. In this sense, one of the reasons to consider both the undirected and the directed formulations, (3.16) and (3.22) respectively, is to evaluate the impact on two different settings.

The algorithms we developed are implemented in *C++*, using the callback mechanism of the Concert API of CPLEX 12.5. All preprocessing, heuristics and cut generation options of the solver are turned off – only user cuts are separated. We consider a numer-

ical precision of 10^{-5}, even when looking for violated constraints. All experiments were carried on a machine with an Intel Core i7 980 (3.33GHz) CPU, with 24GB of RAM. We use the MSTCC benchmark instances proposed by Zhang et al. [2011]; as these are integer-valued, we set the absolute MIP gap tolerance parameter of CPLEX to 0.9999. We set an overall (wall-clock) time limit of 5000 seconds, as done in Zhang et al. [2011]. Note that their hardware platform has quite similar computing capability (described solely as "a PC with 3.4GHz Intel Pentium processor with 2GB of RAM").

The benchmark includes *type 1* and *type 2* instances. The first set is associated with harder problems, and several instances have no optimality or feasibility certificates available for them. The latter set is much easier to solve in practice, possibly because its instances are made feasible through a heuristic. Therefore, most of the following evaluations are based on type 1 instances. We refer to an instance defined on a graph (V, E) and conflict set C by the identifier $|V| - |E| - |C|$.

Finally, concerning the claim that the current approach provides the best known results for this set of benchmark instances, the main argument is that LP relaxation bounds for both formulations (3.16) and (3.22) are stronger than the previous known bounds.

5.2 Effectiveness of the preprocessing algorithm

The preprocessing algorithm has quite different impact considering type 1 and type 2 instances, as presented in the upper and lower sections of Table 5.1, respectively. The first three columns indicate the instance dimensions. The fourth column consists of the total number of edges fixed, while the fifth reports the number of new conflict pairs included in the last phase of the algorithm. The next column indicates the resulting instance dimensions, or the certificate provided (when that is the case), followed by the total execution time.

Recalling that type 2 instances have denser conflict graphs, these are more amenable to the probing techniques we apply. Actually, the algorithm had major impact on this set, and all instances become trivial problems. In most cases, the conflict set is empty, resulting in a standard MST problem (and we thus say that it is solved to optimality).

On the other hand, no similar effect was verified for type 1 instances. In most cases, no edge could be fixed at all, even though the conflict graph could be extended by the pair probing technique of the last phase in the algorithm. Moreover, actually solving the resulting models indicated a negligible impact on solution bounds (in comparison to solving the original instance without preprocessing), with both the undirected and the directed

Table 5.1. Instance reduction using the preprocessing algorithm.

$\|V\|$	$\|E\|$	$\|C\|$	# **Edges Fixed**	# **Conflicts Included**	**Resulting Instance**	**Time (s)**
50	200	199	0	0	$\|V\|=50,\|E\|=200,\|C\|=199$	0.05
50	200	398	0	0	$\|V\|=50,\|E\|=200,\|C\|=398$	0.05
50	200	597	0	0	$\|V\|=50,\|E\|=200,\|C\|=597$	0.06
50	200	995	0	11	$\|V\|=50,\|E\|=200,\|C\|=1006$	0.13
100	300	448	0	23	$\|V\|=100,\|E\|=300,\|C\|=471$	0.4
100	300	897	1	135	$\|V\|=100,\|E\|=299,\|C\|=1026$	0.77
100	300	1344	1	188	$\|V\|=100,\|E\|=299,\|C\|=1472$	0.9
100	500	1247	0	0	$\|V\|=100,\|E\|=500,\|C\|=1247$	0.79
100	500	2495	0	0	$\|V\|=100,\|E\|=500,\|C\|=2495$	0.86
100	500	3741	0	2	$\|V\|=100,\|E\|=500,\|C\|=3743$	1.81
100	500	6237	0	31	$\|V\|=100,\|E\|=500,\|C\|=6268$	1.99
100	500	12474	8	2747	$\|V\|=100,\|E\|=492,\|C\|=12720$	13.8
200	600	1797	0	126	$\|V\|=200,\|E\|=600,\|C\|=1923$	3.96
200	600	3594	0	504	$\|V\|=200,\|E\|=600,\|C\|=4098$	7.34
200	600	5391	–	–	**Infeasible**	9.05
200	800	3196	0	6	$\|V\|=200,\|E\|=800,\|C\|=3202$	7.75
200	800	6392	0	27	$\|V\|=200,\|E\|=800,\|C\|=6419$	8.46
200	800	9588	0	175	$\|V\|=200,\|E\|=800,\|C\|=9763$	8.93
200	800	15980	1	1220	$\|V\|=200,\|E\|=799,\|C\|=16558$	55.44
300	800	3196	–	–	**Infeasible**	41.42
300	1000	4995	0	201	$\|V\|=300,\|E\|=1000,\|C\|=5196$	46.32
300	1000	9990	1	661	$\|V\|=300,\|E\|=999,\|C\|=10477$	42.04
300	1000	14985	–	–	**Infeasible**	60.23
50	200	3903	159	1	$\|V\|=33,\|E\|=41,\|C\|=12$	0.05
50	200	4877	167	3	$\|V\|=27,\|E\|=33,\|C\|=10$	0.03
50	200	5864	175	1	$\|V\|=21,\|E\|=25,\|C\|=7$	0.09
100	300	8609	287	0	**Optimal**	0.05
100	300	10686	291	0	**Optimal**	0.03
100	300	12761	291	0	**Optimal**	0.06
100	500	24740	464	35891	$\|V\|=32,\|E\|=36,\|C\|=2$	2.25
100	500	30886	469	0	**Optimal**	0.24
100	500	36827	465	0	$\|V\|=33,\|E\|=35,\|C\|=1$	0.21
200	400	13660	368	0	$\|V\|=30,\|E\|=32,\|C\|=1$	0.01
200	400	17089	382	0	$\|V\|=17,\|E\|=18,\|C\|=1$	0.01
200	400	20469	392	0	**Optimal**	0.01
200	600	34504	567	0	**Optimal**	0.59
200	600	42860	584	0	**Optimal**	0.19
200	600	50984	588	0	**Optimal**	0.09
200	800	62625	785	0	**Optimal**	0.29
200	800	78387	755	0	**Optimal**	0.24
200	800	93978	786	0	**Optimal**	0.69
300	600	31000	–	–	**Optimal**	0.45
300	600	38216	555	0	$\|V\|=44,\|E\|=45,\|C\|=1$	0.02
300	600	45310	575	0	**Optimal**	0.02
300	800	59600	795	0	**Optimal**	0.03
300	800	74500	775	0	**Optimal**	0.03
300	800	89300	780	0	**Optimal**	0.04
300	1000	96590	984	0	**Optimal**	2.08
300	1000	120500	–	–	**Optimal**	9.12
300	1000	144090	–	–	**Optimal**	17.18

formulations.

Finally, note that nineteen instances are solved during the preprocessing phase. Sixteen type 2 instances are reduced to a problem defined on a tree without conflicting edges, in which case the solution is unique. Three type 1 instances were proved to be infeasible. Note that these are among the hardest problems on the benchmark (the subset

Table 5.2. Improvement on dual bounds of the undirected formulation (with SEC).

Instance	LP Relaxation Bound			MIP Lower Bound		
	OCI	Cliques	OCI+Cliques	OCI	Cliques	OCI+Cliques
50 – 200 – 199	0.0	0.0	0.0	0.0	0.0	0.0
50 – 200 – 398	1.7	1.4	1.4	0.1	0.0	0.0
50 – 200 – 597	2.3	0.7	2.0	0.1	0.0	0.1
50 – 200 – 995	22.9	7.1	23.3	0.0	0.0	0.0
100 – 300 – 448	0.0	0.0	0.0	0.0	0.0	0.0
100 – 300 – 897	6.2	2.4	6.4	0.0	0.0	0.0
100 – 300 – 1344	18.3	6.5	18.5	-1.5	2.8	-0.9
100 – 500 – 1247	0.0	0.0	0.1	0.0	0.0	0.0
100 – 500 – 2495	6.6	1.6	6.6	0.0	0.0	-0.7
100 – 500 – 3741	17.0	7.6	17.0	2.6	4.6	2.7
100 – 500 – 6237	30.4	23.0	30.8	13.5	15.9	14.6
100 – 500 – 12474	44.1	62.2	62.8	6.3	19.8	14.1
200 – 600 – 1797	3.7	0.7	3.7	-0.5	-0.1	-1.2
200 – 600 – 3594	26.2	10.5	26.2	14.2	7.2	16.5
200 – 800 – 3196	3.2	0.8	3.4	-0.9	0.5	-0.4
200 – 800 – 6392	24.1	9.1	24.0	18.6	8.5	18.8
200 – 800 – 9588	38.5	31.2	39.2	26.5	24.2	26.4
200 – 800 – 15980	48.1	52.6	53.7	31.8	46.0	37.8
300 – 1000 – 4995	15.8	3.3	15.8	12.5	2.8	12.0
300 – 1000 – 9990	33.8	21.8	33.8	26.9	17.0	25.4
Average	17.2	12.1	18.4	7.5	7.5	8.3

for which no feasible solution is known, and therefore no duality gap is available). Interestingly, while the branch and cut algorithm could also prove the first two problems (200 – 600 – 5391 and 300 – 800 – 3196) to be infeasible within the time limit, the certificate for the largest one (300 – 1000 – 14985) was only provided by the preprocessing algorithm.

5.3 Impact of odd-cycle and clique inequalities

We consider next the improvement provided by the constraints obtained from the polytope of stable sets in the conflict graph. In practice, it corresponds to the effect of separating odd-cycle inequalities (3.3) and including *a priori* maximal clique inequalities in (3.4), as described in Section 4.3.

Tables 5.2 and 5.3 present the percentual improvement on dual bounds over the plain formulation without these inequalities: i.e. columns *OCI* correspond to the impact of intersecting the spanning tree polytope (P_{sec} for Table 5.2, P_{dcut} for Table 5.3) with the cycle-constrained relaxation P_{cstab} instead of the simplest relaxation P_{rstab}; analogously, columns *Cliques* use P_{qstab} instead of P_{rstab}, while *OCI+Cliques* compare $P_{qstab} \cap P_{cstab}$ with P_{rstab}. We report on percentual strengthening on both the initial LP relaxation and on the final bound provided by the branch and cut algoritm, using all type 1 instances which are not proved to be infeasible.

In general, OCI contributes significantly more to tightening the LP dual bound: in both the directed and undirected formulations, its improvement outperforms that pro-

Table 5.3. Improvement on dual bounds of the directed formulation (with DCUT).

Instance	LP Relaxation Bound			MIP Lower Bound		
	OCI	Cliques	OCI+Cliques	OCI	Cliques	OCI+Cliques
50 – 200 – 199	0.0	0.0	0.0	0.0	0.0	0.0
50 – 200 – 398	1.2	1.2	1.2	0.0	0.0	0.0
50 – 200 – 597	3.1	0.9	3.1	0.0	0.0	0.0
50 – 200 – 995	22.9	7.5	22.9	0.0	0.0	0.0
100 – 300 – 448	0.0	0.0	0.0	0.0	0.0	0.0
100 – 300 – 897	5.5	1.6	5.5	0.0	0.0	0.0
100 – 300 – 1344	17.0	5.4	17.0	9.4	6.4	9.5
100 – 500 – 1247	0.0	0.0	0.0	0.0	0.0	0.0
100 – 500 – 2495	6.9	2.2	6.9	2.6	1.8	2.4
100 – 500 – 3741	16.4	7.5	16.4	11.8	5.8	11.9
100 – 500 – 6237	29.6	22.3	30.0	21.1	18.8	21.3
100 – 500 – 12474	43.4	61.1	61.6	24.8	41.2	39.2
200 – 600 – 1797	3.9	0.7	3.9	2.3	1.2	2.9
200 – 600 – 3594	26.1	10.8	26.1	22.8	10.7	22.3
200 – 800 – 3196	4.2	1.0	4.2	3.5	1.6	3.6
200 – 800 – 6392	23.9	9.3	23.9	21.1	9.5	21.0
200 – 800 – 9588	38.3	31.2	39.0	33.0	29.9	33.0
200 – 800 – 15980	47.9	52.2	53.3	40.6	58.2	50.0
300 – 1000 – 4995	15.8	3.2	15.8	14.8	3.8	14.8
300 – 1000 – 9990	33.7	21.8	33.7	24.1	16.6	24.1
Average	17.0	12.0	18.2	11.6	10.3	12.8

vided solely by maximal clique inequalities for most instances. Still, the complete formulation using both classes is never worse (disregarding a factor of 0.1%), as expected.

The point supporting the methodology of using the strongest formulation is clearer for the final branch and cut bounds. On average, OCI and clique inequalities provide equivalent contribution, though one of them is remarkably more effective in each particular instance. The intuition on using both of them is therefore to capture both scenarios in the proposed algorithm. In this sense, the complete formulation (with *OCI+Cliques*) performs better, on average, than isolated counterparts. It is worth noting that the improvement is greater for larger problem instances.

We also note that, while both formulations present the same behavior, strengthening the MIP in the directed formulation is considerably more effective. A possible factor driving its better performance could be the somewhat simpler implementation of the separation procedure for DCUT than for SEC (despite the equivalent asymptotic complexity), as discussed in Section 4.4.

We also comment on the overall execution time, considering the subset of type 1 instances solved to optimality. We verify in Table 5.4 the expected tradeoff between using formulations providing stronger bounds, and the execution time to compute them, disregarding preprocessing time. The complete formulation (with *OCI+Cliques*) takes longer time to close the gap in most cases, considering both the directed and the undirected formulations. We also note that, though the directed model requires less execution time for most instances, it failed to close the gap for the last one within the time limit (indicated in

Table 5.4. Solution time of type 1 instances solved to optimality.

Instance	Undirected Formulation (SEC)				Directed Formulation (DCUT)			
	Plain	OCI	Cliques	OCI+Cliques	Plain	OCI	Cliques	OCI+Cliques
50 – 200 – 199	0.07	0.07	0.05	0.06	0.01	0.01	0.01	0.01
50 – 200 – 398	0.20	0.08	0.13	0.15	0.03	0.00	0.00	0.00
50 – 200 – 597	0.48	0.67	0.50	0.97	0.65	0.90	0.44	0.58
50 – 200 – 995	10.07	21.42	4.07	21.13	64.53	19.87	10.67	13.46
100 – 300 – 448	0.81	0.82	1.32	1.31	0.03	0.04	0.03	0.03
100 – 300 – 897	133.14	697.87	119.12	1297.34	151.06	430.92	435.40	470.53
100 – 500 – 1247	0.66	0.48	0.54	0.53	0.01	0.01	0.01	0.01
100 – 500 – 2495	567.26	4825.29	528.10	4947.23	4936.85 *	4943.08 *	4889.89 *	4936.82 *

Table 5.4 with an asterisk).

Finally, it is interesting to acknowledge the overall shorter execution time of using *Cliques* instead of the plain formulation. As discussed in Section 4.3, Table 5.5 reports the number of maximal cliques in the conflict graphs of type 1 instances, which is less than the cardinality of the corresponding conflict set itself (except for instance 100 – 500 – 12474).

5.4 Comparison with results from the literature

The goal of this last set of experiments is to indicate how stronger are the bounds provided by the present approach, comparing them with the best results available in the literature. As presented in Section 2.2, these correspond to the Lagrangean relaxation scheme of

Table 5.5. Number of maximal cliques in type 1 instances.

$\|V\|$	$\|E\|$	$\|C\|$	# **Maximal Cliques**
50	200	199	199
50	200	398	369
50	200	597	540
50	200	995	782
100	300	448	448
100	300	897	842
100	300	1344	1154
100	500	1247	1192
100	500	2495	2182
100	500	3741	2914
100	500	6237	4434
100	500	12474	**16508**
200	600	1797	1715
200	600	3594	3063
200	600	5391	4052
200	800	3196	2992
200	800	6392	5300
200	800	9588	6852
200	800	15980	13292
300	800	3196	2992
300	1000	4995	4609
300	1000	9990	7958
300	1000	14985	10744

Table 5.6. Comparing results with Zhang et al. [2011] on previously open instances.

Instance	Bounds of Zhang et al. [2011]	LP Bound (3.16)	LP Bound (3.22)	MIP Bounds (3.16)	MIP Bounds (3.22)	(%)
100 – 300 – 1344	[4681.27 , –]	6059.3	6081.6	[6621.2 , –]	[6561.8 , –]	41.4
100 – 500 – 6237	[4968.99 , –]	7189.2	7214.9	[7568.7 , –]	[7346.4 , –]	52.3
100 – 500 – 12474	[5194.67 , –]	9011.5	9037.8	[9816.9 , –]	[9679.7 , –]	89.0
200 – 600 – 1797	[11425.8 , –]	12906.2	12979.3	[13072.9 , **14707.0**]	[13128.6 , **14107.0**]	14.4
200 – 600 – 3594	[12487 , –]	16791.5	16804.3	[17532.7 , –]	[17563.0 , –]	40.4
200 – 600 – 5391	[12873.2 , –]	infeasible	infeasible	–	–	–
200 – 800 – 3196	[17992.6 , –]	20303.2	20632.5	[20744.2 , **21852.0**]	[20775.5 , **21979.0**]	15.3
200 – 800 – 6392	[19705.7 , –]	25929.1	25975.4	[26361.3 , –]	[25982.4 , –]	33.8
200 – 800 – 9588	[20684.8 , –]	29230.0	29268.4	[29443.6 , –]	[29294.2 , –]	42.3
200 – 800 – 15980	[20226.9 , –]	32271.9	32283.5	[33345.1 , –]	[33231.1 , –]	64.9
300 – 800 – 3196	[30190.1 , –]	infeasible	infeasible	–	–	–
300 – 1000 – 4995	[40732.7 , –]	51066.3	51181.8	[51451.3 , –]	[51181.8 , –]	26.3
300 – 1000 – 9990	[42902.5 , –]	59884.6	59921.4	[60907.8 , –]	[59921.4 , –]	42.0
300 – 1000 – 14985	[44639.1 , –]	infeasible	infeasible	–	–	–

Zhang et al. [2011], where a maximum edge clique partitioning subproblem is solved. As for the primal bounds of their approach, the best result achieved by one of the heuristics they propose is reported here.

Table 5.6 presents the results of the branch and cut algorithm with both formulations (3.16) and (3.22), considering the *hardest* problem instances in the benchmark, namely those for which no feasible solution was known in the work of Zhang et al. [2011]. The third and fourth columns indicate LP relaxation bounds, while columns 5 and 6 present [primal, dual] bounds provided by branch and cut (see Section 5.1 for the complete experimental design). The last column depicts percentual improvement on the best dual bound previously available when using the undirected formulation (3.16).

Two new feasibility certificates are provided, yielding the first primal bounds on instances 200 – 600 – 1797 and 200 – 800 – 3196 (values in bold on Table 5.6). Moreover, we have described in Section 5.2 that three new infeasibility certificates are also presented. Interestingly, the improvement on previous results varies much among instances, ranging from 14% to 89%, motivating further work and different approaches for the MSTCC problem.

A key result in this work regards the consistent improvement of previous known dual bounds by the initial LP relaxation bound of the proposed formulations. In particular, these are tighter for the directed formulation (3.22): on average, the bounds are 37% stronger than those achieved by the Lagrangean lower bounding scheme of Zhang et al. [2011] after hours of computation; in fact, they report execution times of up to 28421.5 seconds.

For completeness, the results on the remaining type 1 instances are presented in Tables 5.7 and 5.8. Columns 2 and 3 present the results reported by Zhang et al. [2011]. Note that the authors report some gap values retrieving the optimal cost from the solution of a

Table 5.7. Comparing undirected formulation with Zhang et al. [2011] on remaining type 1 instances.

Instance	Zhang et al. [2011]		Branch and cut with (3.16)				
	Bounds	Gap	Bounds	MIP Gap	LP Gap	# Nodes	Time (s)
50 – 200 – 199	[702.793 , 708]	0.74	708	0.0	0.99	33	0.1
50 – 200 – 398	[757.816 , 785]	1.58	770	0.0	1.56	55	0.2
50 – 200 – 597	[807.745 , 1044]	11.91	917	0.0	6.99	136	1.0
50 – 200 – 995	[877.495 , 1424]	33.72	1324	0.0	10.94	462	21.1
100 – 300 – 448	[3991.18 , 4102]	1.23	4041	0.0	1.23	165	1.3
100 – 300 – 897	[4624.24 , –]	–	**5658**	**0.0**	8.15	3018	1297.3
100 – 500 – 1247	[4165.68 , 4293]	2.56	4275	0.0	0.64	35	0.5
100 – 500 – 2495	[4805.40 , 6603]	37.4*	[5951.4 , 6006]	0.9	7.48	8079	4947.2
100 – 500 – 3741	[4871.27 , 8787]	80.38*	[6510.8 , 9440]	31.0	33.69	2279	4943.6

flow formulation in a MIP solver, instead of considering the primal bound they achieve; these are marked in column 3 with an asterisk. Columns 4 and 5 correspond to the final solution bounds and duality gap, while column 6 describes the root LP relaxation gap. The last two columns describe number of nodes in the enumeration tree and the total execution (real) time.

We remark the new optimality certificate for instance 100 – 300 – 897, and rather tighter bounds on the two instances which are not solved to optimality. Also note that, on the one hand, one could indicate that the directed formulation has a more efficient performance on the context of branch and cut: smaller LP gaps, less enumeration nodes and shorter execution time, when an optimal solution is reached by both formulations. Nevertheless, it is not as effective as the undirected one in closing the duality gap for the most difficult instances. Not only does it fails to provide a primal bound for the last instance (100 – 500 – 3741) on Table 5.8, but it is also yields weaker dual bounds for most instances in the previous Table 5.6.

Finally, it is worth mentioning that easier, type 2 instances consist of trivial input to the algorithms under discussion. Both formulations systematically solve them to optimality, instantly (executing in negligible wall clock time), in the root LP relaxation node. We therefore choose not to tabulate similar values for these instances, since it would provide no further information.

5.5 Final remarks

The computational experiments presented in this chapter report on the results obtained for the very first approach using cutting planes for the MSTCC problem. The focus is to evaluate the strength of the proposed formulations and the designed algorithms on standard instances currently evaluated in the literature of the problem. The preprocessing algorithm is also evaluated, and was highly successful in turning a set of instances into

Table 5.8. Comparing directed formulation with Zhang et al. [2011] on remaining type 1 instances.

Instance	Zhang et al. [2011]		Branch and cut with (3.22)				
	Bounds	Gap	Bounds	MIP Gap	LP Gap	# Nodes	Time (s)
50 – 200 – 199	[702.793 , 708]	0.74	708	0.0	0.4	3	0.0
50 – 200 – 398	[757.816 , 785]	1.58	770	0.0	0.0	0	0.0
50 – 200 – 597	[807.745 , 1044]	11.91	917	0.0	2.4	10	0.6
50 – 200 – 995	[877.495 , 1424]	33.72	1324	0.0	10.6	269	13.5
100 – 300 – 448	[3991.18 , 4102]	1.23	4041	0.0	0.1	3	0.0
100 – 300 – 897	[4624.24 , –]	–	**5658**	**0.0**	7.0	1360	470.5
100 – 500 – 1247	[4165.68 , 4293]	2.56	4275	0.0	0.0	0	0.0
100 – 500 – 2495	[4805.40 , 6603]	37.4*	[5875.8 , 5997]	2.0	6.4	5365	4936.8
100 – 500 – 3741	[4871.27 , 8787]	80.38*	[6496.8 , –]	–	–	2817	4941.7

trivial computational tasks.

We want to stress the methodology for exploring the conflict graph, which makes sense as it clearly improves both formulations we discuss. Moreover, it provided the best currently known algorithm for solving the MSTCC problem.

We are able to consistently improve the best solution bounds previously available in the literature. While doing so already in the initial LP relaxation, we make a point on the strength of the proposed formulations, and regard it as a main argument on its relative superiority and on the contribution of this work. Instances with up to five hundred edges are solved to optimality, and infeasibility certificates are provided for yet larger instances. Nevertheless, we could not acknowledge any pattern indicating how does the solution gap or execution time scale with instance size, and further work could elaborate on the limits of tractability of MTSCC problem instances.

An extended discussion could explore different factors driving the performance of an algorithmic approach such as the one described here. For instance, evaluating alternative branching rules, heuristics to derive primal bounds, or even different strategies or reformulations to solve the general integer programs. These are left as future work.

Finally, note that the presented experiments are limited to the (randomly built) benchmark instances previously studied in the literature. Further evaluations with different experimental designs could be interesting to support the current claims. In particular, one could evaluate DIMACS stable set benchmark problems [Rebennack et al., 2012] as conflict graphs of different MSTCC problem instances. Alternatively, quadratic MST benchmarks could be converted into MSTCC instances by defining a threshold on interaction costs, beyond which a conflict would be included in the input problem.

Chapter 6

Conclusion

This final chapter briefly conveys our general conclusions on the investigation carried out in this dissertation. First, the observations are established in the context of previous knowledge on the problem, clarifying our contribution. Next, we indicate what might be the forms of further research on MSTCC, describing open issues and different algorithmic approaches and experiments which could yield insight into the tractability of the problem.

6.1 General conclusions

In summary, this work contributes with an exact solution approach to a disjunctively constrained generalization of the MST problem, namely the minimum spanning tree under conflict constraints problem (MSTCC).

The general methodology of exploiting the conflict graph to approach the conflict-constrained problem is the main feature guiding the designed algorithms. We present a general preprocessing method based on implications from problem-specific feasibility conditions, which might be integrated with different techniques for the problem solution. IP formulations are introduced, building on classic polyhedral descriptions to represent the feasibility of a solution with respect to both input graphs $G(V,E)$ and $\hat{G}(E,C)$ of a MSTCC instance.

Computational results support the contribution of that methodology. The preprocessing algorithm succeeds in reducing all problem instances in a benchmark set consisting of denser conflict graphs to trivial problems, which can be solved with negligible computational effort. As for the branch and cut algorithms, we could verify in practice a consistent improvement on the best results previously available in the literature, and

provide new feasibility and optimality certificates for the set of challenging benchmark instances of the problem.

To the best of our knowledge, the present approach is interesting because it: (i) provides certificates or sistematically stronger solution bounds; (ii) provides such bounds with much reduced computational effort, even considering only the initial LP relaxation of the formulations we propose; (iii) presents an underlying methodology of interest by itself, which could be extended in the context of related problems.

6.2 Future work

In the closing section for the main chapters in this thesis, we indicate possible extensions to the modeling and solution techniques we introduced. The reader is thus referred to Sections 3.4, 4.6 and 5.5 for ideas more directly linked to this work. The following is a free discussion on open issues which the author acknowledges might or might not be of interest.

Extending the methodology of exploiting the conflict graph to a further level, we also verify that the MSTCC problem consists of optimizing over k-cardinality stable sets in $\hat{G}(E,C)$, with $k = |E| - 1$, which do not induce cycles in G. The complexity of k-cardinality stable set problems was considered in the literature before [Janssen and Kilakos, 1999]. A possible research line would investigate if this combinatorial structure provides a better approach for MSTCC. Also, in the same model of the work by Darmann et al. [2011], different problems closely related to MSTCC might be conceived, e.g. allowing different types of disjunctive constraints in a single problem instance.

From a graph-theoretical point of view, further investigation on the complexity in particular graph classes could yield further cases which are solvable in polynomial time and infeasibility tests, as well as combinatorial relaxations providing algorithmic approaches. The latter was the case, for instance, in one of the Lagrangean relaxation schemes of Zhang et al. [2011] for MSTCC, where an edge-clique partitioning subproblem arise.

As in the case of related problems in graph theory, one could wonder, if the conflict graph $\hat{G}$ is such that the stable set problem can be solved in polynomial time, whether that would extend for the solution of a MSTCC instance defined over it as well.

For some problems defined on trees, e.g. the k–cardinality tree problem [Simonetti et al., 2013], boundary values of an instance parameter correspond to polinomially-solvable particular cases. One could investigate whether a similar result could be derived for MSTCC, since the sparsity of the conflict graph $\hat{G}$ has a major impact on the tractabil-

ity of instances, with the problem being well-solved in practice for rather sparse or overly dense $\hat{G}$.

We also seek to gain insight into the hardness and tractability of related problems under disjunctive constraints, such as the ones described by Darmann et al. [2011]. Note that the approach we present in this study might be appropriate for these problems as well.

Finally, an interesting investigation would consider the intersection of the *independence systems* at hand: noting that MSTCC regards the optimization over the independence system formed by stable sets in $\hat{G}$, and the graphic matroid of G. Polyhedral results from both could be leveraged to bring insight into the new problem.

Bibliography

Achterberg, T. (2007). Conflict analysis in mixed integer programming. *Discrete Optimization*, 4(1):4 – 20. ISSN 1572-5286.

Ahuja, R. K., Kodialam, M., Mishra, A. K., and Orlin, J. B. (1997). Computational investigations of maximum flow algorithms. *European Journal of Operational Research*, 97(3):509 – 542. ISSN 0377-2217.

Akeb, H., Hifi, M., and Mounir, M. E. O. A. (2011). Local branching-based algorithms for the disjunctively constrained knapsack problem. *Computers & Industrial Engineering*, 60(4):811 – 820. ISSN 0360-8352.

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA. ISBN 0691129932, 9780691129938.

Assad, A. and Xu, W. (1992). The quadratic minimum spanning tree problem. *Naval Research Logistics (NRL)*, 39(3):399–417. ISSN 1520-6750.

Atamtürk, A., Nemhauser, G. L., and Savelsbergh, M. W. (2000). Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, 121(1):40 – 55. ISSN 0377-2217.

Balas, E. (2010). Disjunctive programming. In *50 Years of Integer Programming 1958-2008*, pages 283–340. Springer Berlin Heidelberg.

Balas, E. and Yu, C. (1986). Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4):1054–1068.

Bertsimas, D. and Weismantel, R. (2005). *Optimization Over Integers*. Dynamic Ideas. ISBN 978-0975914625.

Bodlaender, H. and Jansen, K. (1993). On the complexity of scheduling incompatible jobs with unit-times. In Borzyszkowski, A. and Sokolowski, S., editors, *Mathematical Foundations of Computer Science 1993*, volume 711 of *Lecture Notes in Computer Science*, pages 291–300. Springer Berlin / Heidelberg. 10.1007/3-540-57182-5_21.

Bodlaender, H. L., Jansen, K., and Woeginger, G. J. (1994). Scheduling with incompatible jobs. *Discrete Applied Mathematics*, 55(3):219 – 232. ISSN 0166-218X.

Bondy, J.-A. and Murty, U. S. R. (2007). *Graph theory*. Graduate texts in mathematics. Springer, New York, London. ISBN 978-1-8462-8969-9.

Borndörfer, R. and Weismantel, R. (2000). Set packing relaxations of some integer programs. *Mathematical Programming*, 88(3):425–450. ISSN 0025-5610.

Cazals, F. and Karande, C. (2008). A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1–3):564 – 568. ISSN 0304-3975.

Darmann, A., Pferschy, U., and Schauer, J. (2009). Determining a minimum spanning tree with disjunctive constraints. In Rossi, F. and Tsoukias, A., editors, *Algorithmic Decision Theory*, volume 5783 of *Lecture Notes in Computer Science*, pages 414–423. Springer Berlin Heidelberg.

Darmann, A., Pferschy, U., Schauer, J., and Woeginger, G. J. (2011). Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726 – 1735. ISSN 0166-218X. 8th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2009).

de Queiroz, T. A. and Miyazawa, F. K. (2012). Problema da mochila 0-1 bidimensional com restrições de disjunção. In *Anais do XVI CLAIO / XLIV SBPO – Simpósio Brasileiro de Pesquisa Operacional / Congreso Latino-Iberoamericano de Investigación Operativa*, pages 1–12.

Desrochers, M. and Laporte, G. (1991). Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27 – 36. ISSN 0167-6377.

Dessmark, A., Jansson, J., Lingas, A., Lundell, E.-M., and Persson, M. (2007). On the approximability of maximum and minimum edge clique partition problems. *International Journal of Foundations of Computer Science*, 18(02):217–226.

Dezső, B., Jüttner, A., and Kovács, P. (2011). LEMON – an Open Source C++ Graph Template Library. *Electronic Notes in Theoretical Computer Science*, 264(5):23 – 45. ISSN 1571-0661.

Edmonds, J. (1971). Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136. ISSN 0025-5610.

Engels, B. (2011). *A Generalized Network Model for Freight Car Distribution*. PhD thesis, Universität zu Köln. ISBN: 978-3-86853-991-2. URI: http://kups.ub.uni-koeln.de/id/eprint/4262.

Fischetti, M. and Lodi, A. (2003). Local branching. *Mathematical Programming*, 98:23–47. ISSN 0025-5610.

Gerards, A. and Schrijver, A. (1986). Matrices with the edmonds—johnson property. *Combinatorica*, 6(4):365–379. ISSN 0209-9683.

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *J. ACM*, 35(4):921--940. ISSN 0004-5411.

Goossens, D. and Spieksma, F. (2009). The transportation problem with exclusionary side constraints. *4OR*, 7:51–60. ISSN 1619-4500.

Grötschel, M., Lovász, L., and Schrijver, A. (1988). *Geometric algorithms and combinatorial optimization*. Springer Berlin Heidelberg. ISBN 9783642782428.

Hifi, M. and Michrafy, M. (2007). Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Comput. Oper. Res.*, 34(9):2657–2673. ISSN 0305-0548.

Hoffman, K. L. and Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682.

Jansen, K. and Öhring, S. (1997). Approximation algorithms for time constrained scheduling. *Information and Computation*, 132(2):85 – 108. ISSN 0890-5401.

Janssen, J. and Kilakos, K. (1999). Bounded stable sets: Polytopes and colorings. *SIAM J. Discret. Math.*, 12(2):262--275. ISSN 0895-4801.

Johnson, D. S. (2002). A theoretician's guide to the experimental analysis of algorithms. In Goldwasser, M. H., Johnson, D. S., and McGeoch, C. C., editors, *Data Structures near Neighbor Searches and Methodology: Fifth and Sixth Implementation Challenges*, pages 215–250. AMS.

Johnson, E. L. and Padberg, M. W. (1982). Degree-two inequalities, clique facets, and biperfect graphs. *North-Holland Mathematics Studies*, 66:169--187.

Kann, V. (1993). Polynomially bounded minimization problems which are hard to approximate. In Lingas, A., Karlsson, R., and Carlsson, S., editors, *Automata, Languages and Programming*, volume 700 of *Lecture Notes in Computer Science*, pages 52–63. Springer Berlin / Heidelberg. 10.1007/3-540-56939-1_61.

Karamanov, M. and Cornuéjols, G. (2011). Branching on general disjunctions. *Mathematical Programming*, 128(1-2):403–436. ISSN 0025-5610.

Koch, T. and Martin, A. (1998). Solving steiner tree problems in graphs to optimality. *Networks*, 32(3):207--232. ISSN 1097-0037.

Korte, B. and Vygen, J. (2012). *Combinatorial Optimization: Theory and Algorithms*. Springer, 5th edition. ISBN 3642244874, 9783642244872.

Lucena, A. and Beasley, J. E. (1998). A branch and cut algorithm for the steiner problem in graphs. *Networks*, 31(1):39--59. ISSN 1097-0037.

Lucena, A. and Resende, M. G. (2004). Strong lower bounds for the prize collecting steiner problem in graphs. *Discrete Applied Mathematics*, 141(1–3):277 – 294. ISSN 0166-218X.

Magnanti, T. L. and Wolsey, L. A. (1995). Chapter 9 Optimal trees. In M.O. Ball, T.L. Magnanti, C. M. and Nemhauser, G., editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503 – 615. Elsevier.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329. ISSN 0004-5411.

Öncan, T., Zhang, R., and Punnen, A. P. (2013). The minimum cost perfect matching problem with conflict pair constraints. *Computers & Operations Research*, 40(4):920 – 930. ISSN 0305-0548.

Padberg, M. (1973). On the facial structure of set packing polyhedra. *Mathematical Programming*, 5(1):199–215. ISSN 0025-5610.

Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100.

Padberg, M. W. (1979). Covering, packing and knapsack problems. In Hammer, P. L., Johnson, E. L., and Korte, B. H., editors, *Annals of Discrete Mathematics*, volume 4, pages 265 – 287. Elsevier.

Padberg, M. W. and Grötschel, M. (1985). Polyhedral computations. In *The traveling salesman problem: a guided tour of combinatorial optimization*, volume 3, pages 307 – 360. Wiley Chichester.

Pferschy, U. and Schauer, J. (2009). The knapsack problem with conflict graphs. *Journal of Graph Algortihms and Applications*, 13(2):233–249.

Pferschy, U. and Schauer, J. (2011a). The maximum flow problem with conflict and forcing conditions. In Pahl, J., Reiners, T., and Voß, S., editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 289–294. Springer Berlin / Heidelberg.

Pferschy, U. and Schauer, J. (2011b). The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, pages 1–11. ISSN 1382-6905.

Punnen, A. P. and Zhang, R. (2011). Quadratic bottleneck problems. *Naval Research Logistics (NRL)*, 58(2):153–164. ISSN 1520-6750.

Rebennack, S., Reinelt, G., and Pardalos, P. M. (2012). A tutorial on branch and cut algorithms for the maximum stable set problem. *International Transactions in Operational Research*, 19(1-2):161--199. ISSN 1475-3995.

Rossi, F. and Smriglio, S. (2001). A branch-and-cut algorithm for the maximum cardinality stable set problem. *Operations Research Letters*, 28(2):63 – 74. ISSN 0167-6377.

Sadykov, R. and Vanderbeck, F. (2012). Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*.

Samer, P. and Urrutia, S. (2013). Um algoritmo de branch and cut para árvores geradoras mínimas sob restrições de conflito. In *Anais do XLV SBPO – Simpósio Brasileiro de Pesquisa Operacional*, pages 2521–2532.

Simonetti, L., da Cunha, A. S., and Lucena, A. (2013). Polyhedral results and a branch-and-cut algorithm for the $$k$$ -cardinality tree problem. *Math. Program.*, 142(1-2):511–538.

Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28 – 42. ISSN 0304-3975.

Urrutia, S. and Lucena, A. (2014). Characterizing acyclic graphs by labeling edges. *Discrete Applied Mathematics*, 164, Part 2(0):492 – 499. ISSN 0166-218X. LAGOS'11: Sixth Latin American Algorithms, Graphs, and Optimization Symposium, Bariloche, Argentina — 2011.

Wolsey, L. A. (1998). *Integer Programming*. Wiley-Interscience, New York, NY, USA. ISBN 0-471-28366-5.

Yamada, T., Kataoka, S., and Watanabe, K. (2002). Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Transactions of Information Processing Society of Japan*, 43(9):2864 – 2870. ISSN 0387-5806.

Zhang, R., Kabadi, S. N., and Punnen, A. P. (2011). The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191 – 205. ISSN 1572-5286.