HAL
archives-ouvertes.fr

The multi-terminal vertex separator problem : Complexity, Polyhedra and Algorithms

Youcef Magnouche

▶ **To cite this version:**

Youcef Magnouche. The multi-terminal vertex separator problem : Complexity, Polyhedra and Algorithms. Operations Research [cs.RO]. PSL Research University, 2017. English. <NNT : 2017PSLED020>. <tel-01611046>

HAL Id: tel-01611046
https://tel.archives-ouvertes.fr/tel-01611046

Submitted on 5 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Université Paris-Dauphine

The multi-terminal vertex separator problem : Complexity, Polyhedra and Algorithms

École Doctorale de Dauphine — ED 543

Spécialité **Informatique**

COMPOSITION DU JURY :

M. A. Ridha MAHJOUB
Université Paris-Dauphine
Directeur de thèse

M. Denis CORNAZ
Université Paris-Dauphine
Membre du jury

M. Mohamed DIDI BIHA
Université de Caen Basse-Normandie
Rapporteur

M. Nelson MACULAN
Université fédérale de Rio de Janeiro
Rapporteur

Mme Ivana LJUBIC
ESSEC Business School de Paris
Membre du jury

M. Sébastien MARTIN
Université de Lorraine
Membre du jury

M. Frédéric SEMET
École Centrale de Lille
Président du jury

Soutenue le 26.06.2017
par Youcef MAGNOUCHE

Dirigée par **A. Ridha MAHJOUB**

DAUPHINE UNIVERSITÉ PARIS | PSL RESEARCH UNIVERSITY PARIS

Remerciements

Je voudrais en premier lieu remercier Messieurs A. Ridha Mahjoub et Denis Cornaz de m'avoir permis d'effectuer cette thèse.

Je remercie tout particulièrement Monsieur A. Ridha Mahjoub, Professeur à l'Université Paris-Dauphine, de m'avoir apporté son soutien, scientifique et moral, tout au long de ce travail. Il a su me transmettre les connaissances, la rigueur, la motivation et la passion pour la recherche en général et l'optimisation combinatoire en particulier. Il m'a également permis de surmonter les moments difficiles grâce à son très grand intérêt pour mon travail et sa constante disponibilité.

Je tiens aussi à remercier très sincèrement Monsieur Denis Cornaz, maître de conférences à l'Université Paris-Dauphine, pour ses idées et sa détermination qui m'ont permis d'élargir mon domaine de connaissances et le travail réalisé durant cette thèse. Il m'a également appris, grâce à sa haute exigence scientifique et à sa très grande estime de mes capacités, à ne jamais renoncer et à donner le meilleur de moi-même.

Je remercie également vivement Monsieur Sébastien Martin, maître de conférences à l'université de Lorraine, pour toute l'aide qu'il m'a apportée durant cette thèse. Cette aide se situe bien sûr au niveau scientifique, grâce à nos collaborations et aux nombreuses discussions que nous avons eues sur mon travail et à ses conseils et idées.

J'ai été très honoré que Monsieur Mohamed Didi Biha, Professeur des universités à l'Université de Caen Basse-Normandie, ait accepté de rapporter ma thèse. Je remercie également Monsieur Nelson Maculan, Professeur Émérite à l'Université fédérale de Rio de Janeiro, pour l'intérêt qu'il a bien voulu porter à ce travail et pour m'avoir fait l'honneur d'accepter la charge de rapporteur.

Mes remerciements vont également à Madame Ivana Ljubic, Professeur à l'école

supérieure des sciences économiques et commerciales, pour avoir accepté d'examiner mes travaux. Ma gratitude va ensuite à Monsieur Frédéric Semet, Professeur à l'école Centrale de Lille, d'avoir accepté d'examiner ma thèse.

Je tiens également à remercier très chaleureusement mes différents collègues et amis du LAMSADE qui ont partagé avec moi ces dernières années. Les discussions animées et les moments de détente partagés, leur bonne humeur, leurs encouragements et leur amitié, mais aussi les échanges scientifiques et les travaux effectués ensemble, ont grandement contribué à mon épanouissement scientifique et personnel lors de ces années de thèse.

Je tiens à remercier tout particulièrement Khalil, Myriam, Pedro, Thomas, Ian-Christopher, Ioannis et Linda avec qui j'ai partagé mon bureau C605 de Paris-Dauphine. Enfin, mes remerciements ne seraient pas complets si je ne remerciais pas ma famille et mes amis, qui m'ont supporté et appuyé toutes ces années et se sont réellement intéressés à mon travail et au domaine hautement incompréhensible de l'optimisation combinatoire en général.

Abstract

This thesis deals with the multi-terminal vertex separator problem. Given a graph $G = (V \cup T, E)$ with $V \cup T$ the set of vertices, where T is a set of terminals, and a weight function $w : V \to \mathbb{Z}$, associated with nonterminal nodes, the multi-terminal vertex separator problem consists in partitioning $V \cup T$ into $k+1$ subsets $\{S, V_1, \ldots, V_k\}$ such that there is no edge between two different subsets V_i and V_j, each V_i contains exactly one terminal and the weight of S is minimum. In this thesis, we consider the problem from a polyhedral point of view. We give two integer programming formulations for the problem, for one of them, we investigate the related polyhedron and discuss its polyhedral structure. We describe some valid inequalities and characterize when these inequalities define facets. We also derive separation algorithms for these inequalities. Using these results, we develop a Branch-and-Cut algorithm for the problem, along with an extensive computational study is presented.

We also study the multi-terminal vertex separator polytope in the graphs decomposable by one node cutsets. If G is a graph that decomposes into G_1 and G_2, we show that the multi-terminal vertex separator polytope in G can be described from two linear systems related to G_1 and G_2. This gives rise to a technique for characterizing the multi-terminal vertex separator polytope in the graphs that are recursively decomposable. We also obtain a procedure to describe facets for this polytope and show that the multi-terminal vertex separator problem can also be decomposed. Applications of this technique are also discussed.

Moreover, we propose three extended formulations for the problem and derive Branch-and-Price and Branch-and-Cut-and-Price algorithms. For each formulation we present a column generation scheme to solve the linear relaxation, the way to compute the dual bound and the branching scheme. We present computational results and discuss the performance of each algorithm.

Further, we discuss four variants of the multi-terminal vertex separator problem. We

show that all these variants are NP-hard, we give an integer programming formulation and present some valid inequalities.

Key words : Combinatorial optimization, Polytope, Facet, Branch-and-Cut algorithm, Branch-and-Price algorithm, Composition of polyhedra.

Résumé

Dans ce manuscrit nous considérons le problème du séparateur de poids minimum. Étant donné un graphe $G = (V \cup T, E)$, tel que $V \cup T$ représente l'ensemble des sommets où T est un ensemble de terminaux, et une fonction de poids $w : V \to \mathbb{Z}$, associée aux sommets non terminaux, le problème du séparateur de poids minimum consiste à partitionner $V \cup T$ en $k + 1$ sous-ensembles $\{S, V_1, \ldots, V_k\}$ tel qu'il n'y a aucune arête entre deux sous-ensembles V_i et V_j, chaque V_i contient exactement un terminal et le poids de S est minimum. Dans cette thèse, nous étudions le problème d'un point de vue polyédral. Nous donnons deux formulations en nombres entiers pour le problème, et pour une de ces formulations, nous étudions le polyèdre associé. Nous présentons plusieurs inégalités valides, et décrivons leurs conditions de facette. Nous présentons également des algorithmes de séparation pour ces inégalités. En utilisant ces résultats, nous développons un algorithme de génération de coupes et branchement pour le problème, et présentons une analyse approfondie des résultats numériques.

Nous étudions également le polytope des séparateurs dans les graphes décomposables par sommets d'articulation. Si G est un graphe qui se décompose en G_1 et G_2, alors nous montrons que le polytope des séparateurs dans G peut être décrit à partir des deux systèmes linéaires liés à G_1 et G_2. Ceci donne lieu à une technique permettant de caractériser le polytope des séparateurs dans les classes de graphes qui sont récursivement décomposables. Nous obtenons également une procédure de composition de facettes dans ce type de graphes et nous montrons que le problème du séparateur peut être aussi décomposé. Des applications de cette technique sont discutées.

Ensuite, nous étudions des formulations étendues pour le problème et proposons des algorithmes de génération de colonnes et branchements ainsi que des algorithmes de génération de colonnes, de coupes et branchement. Pour chaque formulation, nous présentons un algorithme de génération de colonnes pour résoudre la relaxation linéaire, une procédure pour le calcul de la borne duale ainsi qu'une règle de branchement. Nous

comparons ensuite la performance des algorithmes sur un ensemble d'instances à travers une étude expérimentale extensive.

De plus, nous présentons quatre variantes du problème du séparateur. Nous montrons que les quatre variantes sont NP-difficiles, nous donnons une formulation en nombres entiers et présentons certaines inégalités valides.

Mots clés : Optimisation combinatoire, Polytope, Facette, Algorithme de coupes et branchements, Algorithme de génération de colonnes et branchements, Composition de polyèdres.

Résumé long

Introduction

L'optimisation combinatoire, et en particulier l'approche polyédral, sont très efficaces pour traiter les problèmes combinatoires difficiles. Cette technique, initiée par Edmonds pour le problème de couplage, consiste à réduire le problème à un programme linéaire (ou une séquence de programmes linéaires) en donnant une description complète ou partielle du polytope associé par un système d'inégalités linéaires. L'approche polyédral a été appliquée à de nombreux problèmes d'optimisation combinatoire tels que le problème du voyageur de commerce, le problème de conception de réseau et le problème de la coupe maximum.

Dans cette thèse, nous considérons le *problème du séparateur de poids minimum* PSPM. Étant donné un graphe $G = (V \cup T, E)$ avec $V \cup T$ l'ensemble des sommets, où T est un ensemble de terminaux, et une fonction de poids $w : V \to \mathbb{Z}$, associée aux nœuds non terminaux, le problème du séparateur de poids minimum consiste à partitionner $V \cup T$ en $k+1$ sous-ensembles $\{S, V_1, \ldots, V_k\}$ tel qu'il n'y a pas d'arêtes entre deux sous-ensembles V_i et V_j, chaque V_i contient exactement un terminal et le poids de S est minimum. Ce problème a des applications dans différents domaines comme la conception VLSI, l'algèbre linéaire, les problèmes de connectivité et les algorithmes parallèles. Il a également des applications dans la sécurité des réseaux. Supposons, par exemple, que G représente un réseau de télécommunication où V représente un ensemble de routeurs, T un ensemble de clients, et une arête entre deux sommets exprime la possibilité de transférer des données entre eux. Supposons qu'avec chaque sommet il soit associé un coût, et nous voulons configurer un système de surveillance sur les routeurs afin de surveiller toutes les données échangées entre les clients. L'ensemble

des routeurs sur lequel le système de surveillance est configuré est un séparateur.

Le problème du séparateur de poids minimum est une variante du problème de *k-separator* qui consiste, étant donné un graphe $G = (V, E)$, à partitionner V en $k + 1$ sous-ensembles $\{S, V_1, \ldots, V_k\}$ de telle sorte que $|S|$ soit minimum et qu'il n'y ait aucune arête entre deux sous-ensembles V_i et V_j. De nombreuses variantes de ce problème ont été considérées dans la littérature [14, 25, 26, 32, 42, 48, 95]. Dans cette thèse, nous traitons le PSPM d'un point de vue polyédral. Nous donnons deux formulations en nombres entiers pour le problème, pour une de ces formulations, nous étudions le polyèdre associé et discutons de sa structure polyédral. Nous décrivons certaines inégalités valides et étudions les conditions pour lesquelles ces inégalités définissent des facettes. Nous présentons également des algorithmes de séparation pour ces inégalités. Nous développons ainsi un algorithme de Branch-and-Cut pour le problème et présentons une étude expérimentale approfondie.

Nous étudions également le polytope des séparateurs dans les graphes décomposables par des noeuds d'articulation. Si G est un graphe qui se décompose en G_1 et G_2, nous montrons que le polytope des séparateurs pour G peut être décrit à partir des deux systèmes linéaires liés à G_1 et G_2. Cela donne lieu à une technique de caractérisation du polytope des séparateurs dans les classes de graphes qui sont récursivement décomposables.
Les techniques de composition de polyèdres ont été étudiées dans la littérature pour plusieurs problèmes d'optimisation combinatoire [17], [18], [19], [15], [71], [89]. Dans [15], Barahona, Fonlupt et Mahjoub décrivent une technique de composition pour le polyèdre des sous-graphes acycliques. À l'aide de cette technique, ils donnent une description complète de ce polyèdre pour les graphes sans mineur $K_{3,3}$. Dans [17], Barahona et Mahjoub étudient des compositions similaires pour les polyèdres des sous-graphes induits équilibrés. Dans [18] et [19], ils considèrent le polyèdre des stables et donnent une description complète de ce polytope dans les graphes non contractibles à W_4 (la roue sur 5 sommets). Margot [89] étudie une approche générale de la composition des polyèdres basée sur les techniques de projection.

De plus, nous proposons des formulations étendues pour le problème et développons des algorithmes de Branch-and-Price et Branch-and-Cut-and-Price. Pour chaque formulation, nous présentons un algorithme de génération de colonnes pour résoudre la relaxation linéaire, une procédure pour calculer la borne duale et la règle de branchement. Nous présentons des résultats expérimentaux afin d'analyser la performance de

chaque algorithme.

Nous présentons également quatre variantes du problème du séparateur de poids minimum. Nous montrons que toutes ces variantes sont NP-difficiles, nous donnons une formulation en nombres entiers et présentons certaines inégalités valides.

Ce manuscrit est organisé comme suit. Dans le chapitre 1, nous présentons des notions de base concernant l'optimisation combinatoire. Ce chapitre comprend également un état de l'art du problème du séparateur de poids minimum. Dans le chapitre 2, nous présentons un algorithme de Branch-and-Cut. Ceci est basé sur une étude approfondie du polytope associé. Le chapitre 3 traite la composition des polyèdres pour les graphes obtenus par la 1-somme. Dans le chapitre 4, nous présentons les algorithmes de Branch-and-Price et Branch-and-Cut-and-Price. Le chapitre 5 est consacré à l'étude des quatre variantes du problème du séparateur de poids minimum.

Notions préliminaires et état de l'art

Le premier chapitre est consacré à l'introduction de quelques notions préliminaires concernant l'optimisation combinatoire, les méthodes exactes en général et l'approche polyédral en particulier. Nous donnons notamment un aperçu des méthodes des plans coupants et de génération de colonnes, ainsi que des algorithmes de coupes et branchements, et de génération de colonnes et branchements. Nous donnons alors quelques définitions basiques sur la théorie des graphes et introduisons la terminologie et les notations utilisées dans ce manuscrit. Enfin, nous présentons un état de l'art sur le problème du séparateur de poids minimum.

Le problème du séparateur de poids minimum : Analyse polyédral

Dans ce chapitre, nous considérons le problème du séparateur de poids minimum. Nous montrons d'abord que le problème est NP-difficile, puis nous donnons deux for-

mulations en nombres entiers pour le problème. Pour une de ces formulations, nous étudions le polyèdre associé et discutons sa structure polyédral. Nous décrivons ensuite plusieurs inégalités valides, comme les inégalités de star tree, terminal tree, clique star et de terminal cycle. Nous présentons les conditions pour lesquelles ces inégalités définissent des facettes et nous proposons également des algorithmes de séparation. Toutes les contraintes valides identifiées dans ce chapitre sont intégrées dans un algorithme de coupes et branchements. Nous avons également proposé deux heuristiques pour le problème et analysé leur garantie de performance. L'amélioration d'une heuristique s'est avérée très efficace en résolvant plusieurs instances à l'optimum. De plus nous avons étudié certaines opérations de pré-traitement de graphes, afin de réduire sa taille et ainsi améliorer le temps de résolution. Par ailleurs, une étude expérimentale est également proposée dans ce chapitre, permettant d'avoir un aperçu sur l'efficacité, en pratique, de notre approche.

Le problème du séparateur de poids minimum : Composition de polyèdres

Dans ce chapitre, nous nous sommes intéressés à caractériser le polytope des séparateurs dans les graphes qui sont décomposables par des sommets d'articulation. Lorsque G se décompose en G_1 et G_2, nous proposons une procédure pour composer des polyèdre, des facettes et des algorithmes pour le problème du séparateur de poids minimum. Nous présentons également le problème de couverture de cycles enracinés de poids minimum qui généralise le problème du séparateur de poids minimum. Nous utilisons le théorème de Menger pour fournir une caractérisation de tous les graphes enracinés pour lesquels le nombre maximum de cycles enracinés disjoints est égal à la taille minimum d'un sous-ensemble de sommets intersectant tous les cycles enracinés.

Ce chapitre est organisé comme suit. Dans la section 1, nous donnons deux systèmes linéaires décrivant le polytope des séparateurs pour les star trees et les clique stars. Ensuite, nous montrons que ces deux systèmes linéaires sont totalement duales intégrales. Dans la section 2, nous étudions une technique de composition (décomposition) du polytope des séparateurs, pour les graphes qui sont décomposables par des sommets d'articulation. Ensuite, nous proposons une procédure de composition de facettes et d'algorithmes. Dans la section 3, nous proposons une autre procédure de composi-

tion pour le polytope des séparateurs, par la fusion des terminaux. La section 4 est consacrée au problème de couverture de cycles enracinés de poids minimum.

Le problème du séparateur de poids minimum : Branch-and-Cut-and-Price

Dans ce chapitre nous présentons trois formulations étendues pour le problème du séparateur de poids minimum. La première formulation est basée sur les ensembles terminaux (un ensemble de sommets qui déconnecte un terminal de tous les autres terminaux). En effet, toute solution du problème du séparateur de poids minimum peut être vue comme une partition de l'ensemble des sommets en $k+1$ sous-ensembles, de sorte que k d'entre eux sont des ensembles terminaux disjoints. Ainsi, les variables de cette formulation sont sur les ensembles terminaux. Donc le problème du séparateur de poids minimum se réduit à trouver k ensembles terminaux disjoints tels que la cardinalité de ces sous ensembles soit maximum.

Nous présentons ensuite une formulation à base de séparateurs isolants (un ensemble de sommets déconnectant un terminal de tous les autres terminaux). En effet, toute solution du problème du séparateur de poids minimum peut être vue comme une union de plusieurs séparateurs isolants, chacun intersecte tous les chemins entre un terminal spécifique et tous les autres terminaux. Donc le problème du séparateur de poids minimum se réduit à trouver un ensemble de séparateurs isolants tels que la cardinalité de leur union soit minimum. Pour cette formulation, nous considérons deux familles de variables, une associée aux séparateurs isolants et l'autre aux sommets.

La troisième formulation est basée sur les séparateurs de paires de terminaux. En effet, tout séparateur de G est l'union de plusieurs séparateurs de paires de terminaux et le problème du séparateur de poids minimum se réduit à trouver un ensemble de séparateurs de paires de terminaux tels que la cardinalité de leur union soit minimum. Pour cette formulation, nous considérons deux familles de variables, une associée aux séparateurs de paires de terminaux et l'autre aux sommets.

Pour chacune de ces formulations, nous développons un algorithme de Branch-and-Price et nous présentons le sous problème associé, la règle de branchement et une méthode pour calculer la borne duale durant la phase de génération de colonnes. De

plus, pour les deux dernières formulations nous développons un algorithme de Branch-and-Cut-and-Price en utilisant quelques inégalités valides introduites dans le Chapitre 2. Des résultats expérimentaux sont présentés sur un ensemble d'instances afin de comparer les différents algorithmes.

Les variantes du problème du séparateur de poids minimum

Dans ce chapitre nous considérons quatre variantes du problème du séparateur de poids minimum. Dans la première section, nous considérons le problème du séparateur des composantes connexes. Étant donné un graphe $G = (V \cup T, E)$ et une fonction de poids $w : V \to \mathbb{Z}$, le problème du séparateur des composantes connexes consiste à trouver un sous-ensemble de sommets $S \subseteq V$ de poids minimum tel que $G \setminus S$ a $|T|$ composantes connexes, chacune contenant exactement un terminal.

Dans la seconde section nous considérons le problème du séparateur connexe de terminaux. Étant donné un graphe $G = (V \cup T, E)$ et une fonction de poids $w : V \to \mathbb{Z}$, le problème consiste à trouver un sous-ensemble de sommets $S \subseteq V$ de poids minimum tel que $G \setminus S$ a $|T|$ composantes (pas nécessairement connexes), chacune contient exactement un terminal et le graphe induit par S est connexe.

La troisième section est consacrée au problème du k-séparateur de poids minimum. Étant donné un graphe $G = (V \cup T, E)$, un nombre entier k et une fonction de poids $w : V \to \mathbb{Z}$, le problème consiste à trouver un sous-ensemble de sommets $S \subseteq V$ de poids minimum tel que le graphe $G \setminus S$ a k sous-ensembles disjoints avec au plus un terminal, si $k > |T|$ et au moins un terminal, si $k < |T|$.

Pour chaque variante, nous montrons que le problème est NP-difficile, puis nous le modélisons par un programme linéaire en nombres entiers et nous décrivons plusieurs inégalités valides.

Conclusion

Dans cette dissertation, nous avons étudié le problème du séparateur de poids minimum d'un point de vue polyédral. Dans la première partie de la thèse, nous avons proposé deux formulations en nombres entiers pour le problème, nous avons étudié le polyèdre associé et discuté de sa structure polyédral. Nous avons ensuite présenté plusieurs classes d'inégalités valides et décrit les conditions nécessaires et suffisantes pour que ces inégalités définissent des facettes. À partir de ces résultats nous avons développé un algorithme de Branch-and-Cut pour le problème. Ce dernier a été utilisé afin de résoudre les instances de DIMACS ainsi que des instances aléatoires. Les expérimentations numériques montrent l'efficacité des inégalités valides et les procédures de séparation utilisées dans cet algorithme.

Ensuite, nous avons étudié la caractérisation du polytope des séparateurs. Nous avons donné une description complète du polytope pour les terminal paths, les star trees et les clique stars. En outre, nous avons montré que le système linéaire donné par les inégalités des star trees (resp. cliques stars) et les inégalités triviales est totalement duale intégrale pour les star trees (resp. clique stars). Ensuite, nous avons étudié le polytope des séparateurs dans les graphes décomposables par des sommets d'articulation. Nous avons montré que, si G est un graphe qui se décompose en G_1 et G_2, le polytope des séparateurs pour G peut être décrit à partir des deux systèmes linéaires liés à G_1 et G_2.

Nous avons également proposé trois formulations étendues pour le problème du séparateur de poids minimum. Nous avons développé des algorithmes de Branch-and-Price pour toutes les formulations et des algorithmes de Branch-and-Cut-and-Price pour deux d'entre elles. Pour chaque formulation, nous avons présenté un algorithme de génération de colonnes pour résoudre la relaxation linéaire, une procédure pour le calcul de la borne duale ainsi qu'une règle de branchement. Nous avons présenté des résultats expérimentaux et discuté la performance de chaque algorithme.

Dans la dernière partie de la thèse, nous avons considéré quatre variantes du problème du séparateur de poids minimum. Nous avons montré que chaque problème était NP-difficile, proposé une formulation en nombres entiers et présenté plusieurs inégalités valides.

Il existe de nombreuses directions de recherche pour poursuivre les travaux de cette thèse. Nous pourrons proposer plusieurs amélioration pour l'algorithme de Branch-and-Cut. Des heuristiques de séparation plus efficaces et des méthodes de pré-traitement plus sophistiquées peuvent être développées afin de faciliter la résolution du problème. Par ailleurs, mettre en œuvre des stratégies de branchements plus élaborées pourrait être une piste intéréssante.

En outre, par rapport à la description du polytope des séparateurs, il serait intéressant de caractériser les graphes pour lesquels les inégalités de terminal tree, clique star et de terminal cycle suffisent pour décrire le polytope des séparateurs. En effet, comme il ressort des résultats numériques dans le Chapitre 2, en utilisant ces inégalités, de nombreuses instances ont été résolues dans la racine de l'arbre de branchement dans l'algorithme de Branch-and-Cut. De plus, dans [95], les auteurs caractérisent la classe de graphes pour lesquels le système linéaire donné par les inégalités de terminal path et les inégalités triviales est TDI. Étant donné que les inégalités de terminal tree, clique star et terminal cycle généralisent les inégalités de terminal path, il serait également intéressant de décrire les graphes pour lesquels le système, donné par ces classes d'inégalités et les inégalités triviales, est TDI.

En ce qui concerne les algorithmes de Branch-and-Price et Branch-and-Cut-and-Price, plusieurs améliorations pourraient être réalisées. À partir des tests numériques, nous avons constaté que la borne lagrangienne n'était pas stable. Il serait intéressant d'essayer certaines méthodes de stabilisation de l'algorithme de génération de colonnes afin de réduire le temps de convergence. Il est également intéressant de comparer différentes stratégies de génération de colonnes et différentes heuristiques pour résoudre les sous-problèmes. Pour l'algorithme de Branch-and-Cut-and-Price, il serait intéressant d'ajouter d'autres inégalités valides et d'améliorer les algorithmes de séparation.

Contents

Introduction

Combinatorial optimization, and in particular the polyhedral approach are very powerful for treating hard combinatorial problems. This technique, initiated by Edmonds for the matching problem, consists in reducing the problem to a linear program (or a sequence of linear programs) by giving a complete or partial description of the associated polytope by a system of linear inequalities. The polyhedral approach has been applied to many combinatorial optimization problems such as the Travelling Salesman Problem, the Network Design Problem and the Max-Cut Problem.

In this thesis we consider the *multi-terminal vertex separator problem* (MTVSP). Given a graph $G = (V \cup T, E)$ with $V \cup T$ the set of vertices, where T is a set of terminals, and a weight function $w : V \to \mathbb{Z}$, associated with nonterminal nodes, the multi-terminal vertex separator problem consists in partitioning $V \cup T$ into $k+1$ subsets $\{S, V_1, \ldots, V_k\}$ such that there is no edge between two different subsets V_i and V_j, each V_i contains exactly one terminal and the weight of S is minimum. The MTVSP has applications in different areas like VLSI conception, linear algebra, connectivity problems and parallel algorithms. It has also applications in network security. Suppose, for instance, that G represents a telecommunication network where V represents a set of routers, T a set of customers, and an edge between two vertices expresses the possibility of transferring data between each other. Suppose that with each vertex it is associated a cost, and we want to set up a monitoring system on the routers in order to monitor all data exchanged between customers. The set of routers on which the monitoring system is set up is a multi-terminal vertex separator.

The MTVSP is a variant of the *k-separator problem* which consists, given a graph $G = (V, E)$, in partitioning V into $k + 1$ subsets $\{S, V_1, \ldots, V_k\}$ in such a way that $|S|$ is minimum and there is no edge between two subsets V_i and V_j. Many variants of this problem have been considered in the literature [14, 25, 26, 32, 42, 48, 95]. In this thesis, we deal with the MTVSP from a polyhedral point of view. We give two integer programming formulations for the problem, for one of these formulations, we investigate the related polyhedron and discuss its polyhedral structure. We describe

some valid inequalities and characterize when these inequalities define facets. We also derive separation algorithms for these inequalities. Using these results, we develop a Branch-and-Cut algorithm for the problem, along with an extensive computational study is presented.

We also study the multi-terminal vertex separator polytope in the graphs decomposable by one node cutsets. If G is a graph that decomposes into G_1 and G_2, we show that the multi-terminal vertex separator polytope in G can be described from the two linear systems related to G_1 and G_2. This gives rise to a technique for characterizing the multi-terminal vertex separator polytope in graph classes that are recursively decomposable. Composition techniques have been studied in the litterature for several combinatorial optimization problems [17], [18], [19], [15], [71], [89]. In [15], Barahona, Fonlupt and Mahjoub describe a composition technique for the acyclic spanning subgraphs polyhedron. Using this technique, they give a complete description of this polyhedron for the graphs with no $K_{3,3}$ minor. In [17], Barahona and Mahjoub study similar compositions for the balanced induced subgraphs polyhedron. In [18] and [19], they consider the stable set polyhedron and give a complete description of this polytope in the graphs not contractible to W_4 (the wheel on 5 vertices). Margot [89] studies a general approach of composition of polyhedra based on projection techniques.

Moreover, we propose extended formulations for the problem and derive Branch-and-Price and Branch-and-Cut-and-Price algorithms. For each formulation we present a column generation scheme to solve the linear relaxation, the way to compute the dual bound and the branching scheme. We present computational results are discuss the performance of each algorithm. We also present four variants of the multi-terminal vertex separator problem. We show that all the variants are NP-hard. For each variant we give an integer programming formulation and present some valid inequalities.

This manuscript is organized as follows. In Chapter 1 we present basic notions of combinatorial optimization. This chapter also includes a state-of-the-art on multi-terminal vertex separator problem. In Chapter 2 we present a Branch-and-Cut algorithm. This is based on a deep investigation on the associated polytope. Chapter 3 deals with the composition of polyhedra on graphs obtained by 1-sums. In Chapter 4, we present Branch-and-Price and Branch-and-Cut-and-Price algorithms. Chapter 5 is devoted to the study of the four variants of the multi-terminal vertex separator problem.

Chapter 1

Preliminaries and State-of-the-art

Contents

In this chapter we give some basic notions on combinatorial polyhedra, graph theory and the theory of complexity. We then perform a brief summary of cutting planes and column generation methods as well as Branch-and-Cut and Branch-and-Price algorithms. We finish this chapter with notations and definitions that will be used all along this manuscript and the state-of-the-art on the multi-terminal vertex separator problem.

1.1 Combinatorial optimization

Combinatorial optimization is a branch of computer science and applied mathematics. It concerns the problems that can be formulated as follows: Let $E = \{e_1, \ldots, e_n\}$ a finite set called *basic set*, where each element e_i has a *weight* $c(e_i)$. Let $\mathcal{S}$ a family of subsets of E. If $S \in \mathcal{S}$, then $c(S) = \sum_{e_i \in S} c(e_i)$ is the weight of S. The problem is to determine an element of $\mathcal{S}$, with the smaller (or larger) weights. Such a problem is called a *combinatorial optimization problem*. The set $\mathcal{S}$ is called the *set of solutions of the problem*. In other words,

$$\min(\text{or } \max)\{c(S) : S \in \mathcal{S}\}.$$

The term *combinatorial* refers to the discrete structure of $\mathcal{S}$. In general, this structure is represented by a graph. The term *optimization* signifies that we are looking for the best element in the set of feasible solutions. This set generally contains an exponential number of solutions, therefore, one can not expect to solve a combinatorial optimization problem by exhaustively enumerate all its solutions. Such a problem is then considered as a hard problem.

Various effective approaches have been developed to tackle combinatorial optimization problems. Some of these approaches are based on graph theory, while others use linear and non-linear programming, integer programming and polyhedral approach. Besides, several practical problems arising in real life, can be formulated as combinatorial optimization problems. Their applications are in fields as diverse as telecommunications, transport, industrial production planing or staffing and scheduling in airline companies. Over the years, the discipline got thus enriched by the structural results related to these problems. And, conversely, the progress made in computed science have made combinatorial optimization approaches even more efficient on real-world problems.

In fact, combinatorial optimization is closely related to algorithm theory and computational complexity theory as well. The next section introduces computational issues of combinatorial optimization.

1.2 Computational complexity

Computational complexity theory is a branch of theoretical computer science and mathematics, whose study started with works of Cook [41], Edmonds [54] and Karp [78].

Its objective is to classify a given problem depending on its difficulty. A plentiful literature can be find on this topic, see for instance [60] for a detailed presentation of NP-completeness theory.

A *problem* is a question having some input parameters, and to which we aim to find an answer. A problem is defined by giving a general description of its parameters, and by listing the properties that must be satisfied by a solution. An *instance* of the problem is obtained by giving a specific value to all its input parameters. An *algorithm* is a sequence of elementary operations that allows to solve the problem for a given instance. The number of input parameters necessary to describe an instance of a problem is the *size* of that problem.

An algorithm is said to be polynomial if the number of elementary operations necessary to solve an instance of size n is bounded by a polynomial function in n. We define the class P as the class gathering all the problems for which there exists some polynomial algorithm for each problem instance. A problem that belongs to the class P is said to be "easy" or "tractable".

A *decision problem* is a problem with a *yes* or *no* answer. Let $\mathcal{P}$ be a decision problem and $\mathcal{I}$ the set of instances such that their answer is yes. $\mathcal{P}$ belongs to the *class NP* (Non-deterministic Polynomial) if there exists a polynomial algorithm allowing to check if the answer is yes for all the instances of $\mathcal{I}$. It is clear that a problem belonging to the class P is also in the class NP. Although the difference between P and NP has not been shown, it is a highly probable conjecture.

In the class NP, we distinguish some problems that may be harder to solve than others. This particular set of problems is called *NP-complete*. To determine whether a problem is NP-complete, we need the notion of *polynomial reducibility*. A decision problem P_1 can be polynomially reduced (or transformed) into another decision problem P_2, if there exists a polynomial function f such that for every instance I of P_1, the answer is "yes" if and only if the answer of $f(I)$ for P_2 is "yes". A problem $\mathcal{P}$ in NP is also NP-complete if every other problem in NP can be reduced into $\mathcal{P}$ in polynomial time. The Satisfiability Problem (SAT) is the first problem that was shown to be NP-complete (see [41]).

With every combinatorial optimization problem is associated a decision problem. Furthermore, each optimization problem whose decision problem is NP-complete is said to be *NP-hard*. Note that most of combinatorial optimization problems are NP-hard. One of the most efficient approaches developed to solve those problems is the so-called polyhedral approach.

1.3 Preliminaries and State-of-the-Art

1.3.1 Elements of polyhedral theory

The polyhedral method was initiated by Edmonds in 1965 [55] for a matching problem. It consists in describing the convex hull of problem solutions by a system of linear inequalities. The problem reduces then to the resolution of a linear program. In most of the cases, it is not straightforward to obtain a complete characterization of the convex hull of the solutions for a combinatorial optimization problem. However, having a system of linear inequalities that partially describes the solutions polyhedron may often lead to solve the problem in polynomial time. This approach has been successfully applied to several combinatorial optimization problems. In this Section, we present the basic notions of polyhedral theory. The reader is referred to works of Schrijver [108] and [88].

We shall first recall some definitions and properties related to polyhedral theory.

Let n be a positive integer and $x \in \mathbb{R}^n$. Let say that x is a *linear combination* of x_1, x_2, ..., $x_m \in \mathbb{R}^n$ if there exist m scalar λ_1, λ_2, ..., λ_m such that $x = \sum_{i\in 1}^{m} \lambda_i x_i$. If $\sum_{i=1}^{m} \lambda_i = 1$, then x is said to be an *affine combination* of x_1, x_2, ..., x_m. Moreover, if $\lambda_i \geq 0$, for all $i \in \{1, \ldots, m\}$, we say that x is a *convex combination* of x_1, x_2, ..., x_m.

Given a set $S = \{x_1, \ldots, x_m\} \in \mathbb{R}^{n\times m}$, the *convex hull* of S is the set of points $x \in \mathbb{R}^n$ which are convex combination of x_1, ..., x_m (see Figure 1.1), that is

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

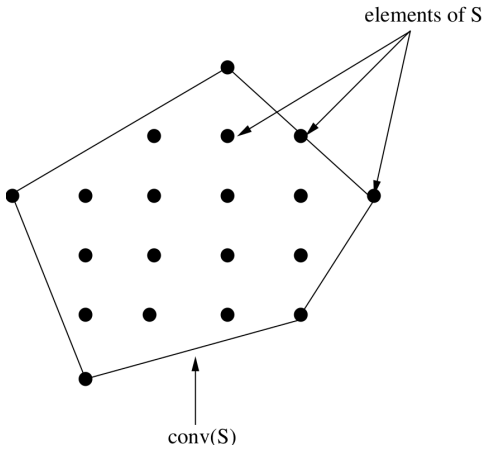


Figure 1.1: A convex hull

The points $x_1, \ldots, x_m \in \mathbb{R}^n$ are *linearly independents* if the unique solution of the system

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

is $\lambda_i = 0$, for all $i \in \{1, \ldots, m\}$. They are *affinely independent* if the unique solution of the system

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

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

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

A polyhedron P is said to be of *dimension* p if it has at most $p+1$ affinely independent solutions. We denote it by $dim(P) = p$. We also have that $dim(P) = n - rank(A^=)$, where $A^=$ is the submatrix of A of *inequalities* that are satisfied with equality by all solutions of P (implicit equalities). The polyhedron P is full dimensional if $dim(P) = n$.

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

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

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

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

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

Figure 1.2 illustrates the main definitions given is this Section.

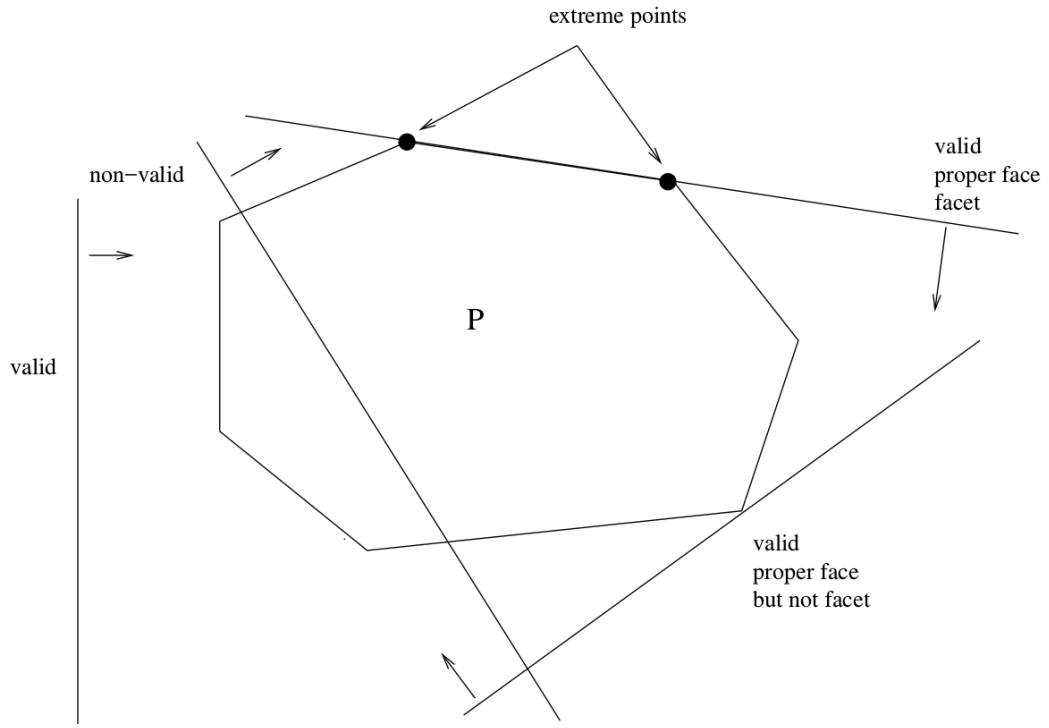


Figure 1.2: Valid inequality, facet and extreme points

Let $Ax \leq b$ be a linear system. The system $Ax \leq b$ is said to be *Totally Dual Integral* (TDI) if for all integer $c \in \mathbb{Z}^n$, the linear program $\min\{b^\top y : A^\top y \geq c; y \geq 0\}$ has an integer optimal solution, if such solution exists.

If $Ax \leq b$ is TDI and b is integral, then the polytope given by $Ax \leq b$ is integral.

1.3.2 Cutting plane method

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

set of a facet defining inequalities. And when all the inequalities of this set are known, then solving $\mathcal{P}$ is equivalent to solve a linear program.

Recall that the objective of the polyhedral approach for combinatorial optimization problems is to reduce the resolution of $\mathcal{P}$ to that of a linear program. This reduction induces a deep investigation of the polyhedron associated with $\mathcal{P}$. It is generally not easy to characterize the polyhedron of a combinatorial optimization problem by a system of linear inequalities. In particular, when the problem is NP-hard there is a very little hope to find such a characterization. Moreover, the number of inequalities describing this polyhedron is, most of the time, exponential. Therefore, even if we know the complete description of that polyhedron, its resolution remains in practice a hard task because of the large number of inequalities.

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

The cutting plane method is based on the so-called *separation problem*. This consists, given a polyhedron P of $\mathbb{R}^n$ and a point $x^* \in \mathbb{R}^n$, in verifying whether if x^* belongs to P, and if this is not the case, to identify an inequality $a^T x \leq b$, valid for P and violated by x^*. In the later case, we say that the hyperplane $a^T x = b$ separates P and x^* (see Figure 1.3).

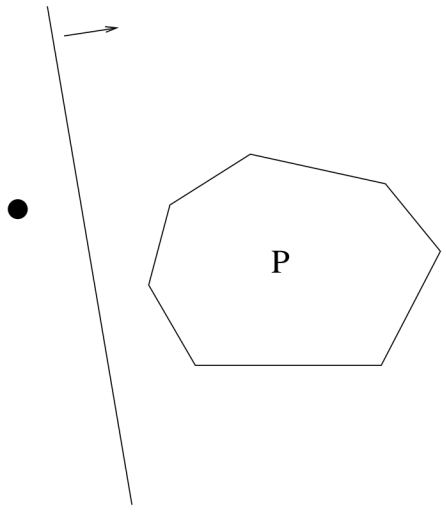


Figure 1.3: A hyperplane separating x^* and P

Grötschel, Lovász and Schrijver [68] have established the close relationship between separation and optimization. In fact, they prove that optimizing a problem over a polyhedron P can be performed in polynomial time if and only if the separation problem associated with P can be solved in polynomial time. This equivalence has permitted an important development of the polyhedral methods in general and the cutting plane method in particular. More precisely, the *cutting plane* method consists in solving successive linear programs, with possibly a large number of inequalities, by using the following steps. Let $LP = \max\{cx, Ax \leq b\}$ be a linear program and LP' a linear program obtained by considering a small number of inequalities among $Ax \leq b$. Let x^* be the optimal solution of the latter system. We solve the separation problem associated with $Ax \leq b$ and x^*. This phase is called the *separation phase*. If every inequality of $Ax \leq b$ is satisfied by x^*, then x^* is also optimal for LP. If not, let $ax \leq \alpha$ be an inequality violated by x^*. Then, we add $ax \leq \alpha$ to LP' and repeat this process until an optimal solution is found. Algorithm 1 summarizes the different cutting plane steps.

Algorithm 1: A cutting plane algorithm

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

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

1.3.3 Branch-and-Cut algorithm

Consider again a combinatorial optimization problem $\mathcal{P}$ and suppose that $\mathcal{P}$ is equivalent to $\max\{cx|Ax \leq b, x \in \{0,1\}^n\}$, where $Ax \leq b$ has a large number of inequalities. A Branch-and-Cut algorithm starts by creating a Branch-and-Bound tree whose root node corresponds to a linear program $LP_0 = max\{cx|A_0x \leq b_0, x \in \mathbb{R}^n\}$, where $A_0x \leq b_0$ is a subsystem of $Ax \leq b$ having a small number of inequalities. Then, we solve the linear relaxation of $\mathcal{P}$ that is $LP = \{cx|Ax \leq b, x \in \mathbb{R}^n\}$ using a cutting plane algorithm whose starting from LP_0. Let x_0^* denote its optimal solution and $A_0'x \leq b_0'$ the set of inequalities added to LP_0 at the end of the cutting plane phase. If x_0^* is integral, then it is optimal. If x_0^* is fractional, then we perform a *branching phase*. This step consists in choosing a variable, say x^1, with a fractional value and adding two nodes P_1 and P_2 in the Branch-and-Cut tree. The node P_1 corresponds to the linear program $LP_1 = \max\{cx|A_0x \leq b_0, A_0'x \leq b_0', x^1 = 0, x \in \mathbb{R}^n\}$ and $LP_2 = \max\{cx|A_0x \leq b_0, A_0'x \leq b_0', x^1 = 1, x \in \mathbb{R}^n\}$. We then solve the linear program $\overline{LP}_1 = \max\{cx|Ax \leq b, x^1 = 0, x \in \mathbb{R}^n\}$ (resp. $\overline{LP}_2 = \max\{cx|Ax \leq b, x^1 = 1, x \in \mathbb{R}^n\}$) by a cutting plane method, starting from LP_1 (resp. LP_2). If the optimal solution of $\overline{LP}_1$ (resp. $\overline{LP}_2$) is integral then, it is feasible for $\mathcal{P}$. Its value is then a lower bound of the optimal solution of $\mathcal{P}$, and the node P_1 (resp. P_2) becomes a leaf of the Branch-and-Cut tree. If the solution is fractional, then we select a variable with a fractional value and add two children to the node P_1 (resp. P_2), and so on.

Note that sequentially adding constraints of type $x^i = 0$ and $x^i = 1$, where x^i is a fractional variable, may lead to an infeasible linear program at a given node of the Branch-and-Cut tree. Or, if it is feasible, its optimal solution may be worse than the best known lower bound of the problem. In both cases, that node is pruned from the Branch-and-Cut tree. The algorithm ends when all nodes have been explored and the optimal solution of $\mathcal{P}$ is the best feasible solution given by the Branch-and-Bound tree.

This algorithm can be improved by computing a good lower bound of the optimal solution of the problem before it starts. The lower bound can be used by the algorithm to prune the node which will not allow an improvement of this lower bound. This would permit to reduce the number of nodes generated in the Branch-and-Cut tree, and hence, reduce the time used by the algorithm. Furthermore, this lower bound may be improved by comparing at each node of the Branch-and-Cut tree a feasible solution when the solution obtained at the root node is fractional. Such a procedure is referred to as a *primal heuristic*. It aims to produce a feasible solution for $\mathcal{P}$ from the solution obtained at a given node of the Branch-and-Cut tree, when this later solution is fractional (and hence infeasible for $\mathcal{P}$). Moreover, the weight of this solution must be as good as possible. When the solution computed is better than the best known lower

bound, it may significantly reduce the number of generated nodes, as well as the CPU time. Moreover, this guarantees to have an approximation of the optimal solution of $\mathcal{P}$ before visiting all the nodes of Branch-and-Cut tree, for instance when a CPU time limit has been reached.

The Branch-and-Cut approach has shown a great efficiency to solve various problems of combinatorial optimization that are considered difficult to solve, such as the Travelling Salesman Problem [10]. Note a good knowledge of the polyhedron associated with the problem, together with efficient separation algorithms (exacts as well as heuristics), might help to improve the effectiveness of this approach. Besides, the cutting plane method is efficient when the number of variables is polynomial. However, when the number of variables is large (for instance exponential), further methods, as column generation are more likely to be used. In what follows, we briefly introduce the outline of this method.

1.4 Column generation and Branch-and-Price

Compact formulations of combinatorial optimization problems often provide a weak linear relaxation. Those problems require then further formulations, whose linear relaxation is closer to the convex hull of feasible solutions. Those reformulations may have a huge number of variables, so that one can not consider them explicitly in the model. We describe a method that suits well to such reformulation, that is the so-called *column generation method*.

1.4.1 Column generation procedure

The column generation method is used to solve linear programs with a huge number of variables only by considering a few number among these variables. This method was pioneered by Dantzig and Wolfe in 1960 [47] in order to solve problems that could not be handled efficiently because of their size (CPU time and memory consumption). Column generation is generally used either for problems whose structure is suitable for a *Dantzig-Wolfe decomposition*, or for problems with a large number of variables. Gilmore and Gomory [64, 65] used this method to solve a *cutting stock problem* belonging to the later class.

The overall idea of column generation is to solve a sequence of linear programs with a restricted number of variables (also referred to as columns). The algorithm starts by

solving a linear program having a small number of variables, and such that a feasible solution for the original problem may be identified using this basis. At each iteration of the algorithm, we solve the so-called *pricing problem* whose objective is to identify the variables which must enter the current basis. These variables are characterized by a negative reduced cost. The reduced cost associated with a variable is computed using the dual variables associated with the constraints of the problem. We then solve the linear program obtained by adding the generated variables, and repeat the procedure until no variable with reduced cost can be identified by the pricing problem. In this case, the solution obtained from the last restricted program is optimal for the original model. The main step of column generation procedure is summarized in Algorithm 2.

Algorithm 2: A column generation algorithm

Data : A linear program MP (Master Problem) with a huge number of variables
Output : optimal solution x^* of MP

1: Consider a linear program RMP (Restricted Master Problem) including only a small subset of variables of the MP;
2: Solve RMP and let x^* be an optimal solution;
3: Solve the pricing problem associated with the dual variables obtained by the resolution of the RMP;
4: **If** there exists a variable x with a negative reduced cost then;
5: add x to RMP.
6: go to 2.
7: **else**
8: x^* is optimal for MP.
9: return x^*.

The column generation method can be seen as the dual of the cutting plane method since it adds columns (variables) instead of rows (inequalities) in the linear program. Furthermore, the pricing problem may be NP-hard. One can then use heuristic procedures to solve it. For more details on column generation algorithms, the reader is referred to [110, 50, 85]. We denote by LMP the linear relaxation of the master problem and by DLMP the dual of the LMP.

1.4.2 Branch-and-Price algorithm

The solution obtained by a column generation procedure may not be integer. Therefore, to solve an integer programming problem, the column generation method has to

be integrated within a Branch-and-Bound framework. This is known a *Branch-and-Price algorithm*. Branch-and-Price is similar to Branch-and-Cut approach, except that procedure focuses on column generation rather than row generation. In fact, generating variables (pricing) and adding inequalities (cutting plane) are complementary operations to strengthen the linear relaxation of an integer programming formulation.

The Branch-and-Price procedure works as follows. Each node of the Branch-and-Bound tree is solved by column generation, so that variables may be added to improve the linear relaxation of the current LP. The branching phase occurs when no columns price out to enter the basis and the solution of the linear program is not integer.

Branch-and-Price approaches have been widely used in the literature to solve large scale integer programming problems. The applications are in various fields, and even real life problems such as Cutting stock problem [8], Generalized Assignment Problem (GAP) [106], Airline Crew Scheduling [21], Multi-commodity Flow Problems [22], etc.

Note that, at each node of the Branch-and-Price tree, column generation may be combined with cutting plane approach, to tighten the LP relaxation of the problem. In this case, the algorithm is called *Branch-and-Cut-and-Price algorithm*. Such a method can be difficult to handle, since adding valid inequalities to the initial model may change the structure and complexity of the pricing problem. However, some successful applications of this algorithm can be found in the literature (see [102], [22] for instance).

1.5 Graph theory

In this section we will introduce some basic definitions and notations of graph theory that will be used throughout the Chapters of this dissertation. For more details, we refer the reader to [108].

1.5.1 Undirected graphs

An undirected graph is denoted $G = (V, E)$ where V is the set of vertices or nodes and E is the set of edges. If e is an edge between two vertices u and v, then u and v are called the ends of e, and we write $e = uv$ or $e = \{u, v\}$. If u is an extremity of e, then u (resp. e) is said to be incident to e (resp. u). Similarly, two vertices u and v forming an edge are said to be adjacent. Since the graph G may have multiple edges, it may be that $e = uv$ and $f = uv$ but $e \neq f$.

If $F \subseteq E$ is a subset of edges, then $V(F)$ represents the node set of edges of F. If $W \subseteq V$ is a subset of vertices, then $E(W)$ denotes the set of edges having their two ends in W. Let $V(H)$ and $E(H)$ be the sets U and F, respectively.

A subgraph $H = (U, F)$ of G is a graph such that $U \subseteq V$ and $F \subset E$. A subgraph $H = (U, F)$ of G is called *covering* or *spanning* if $U = V$. Let $W \subseteq V$, $H = (W, E(W))$ is said to be *subgraph* of G induced by W and will be denoted by $G[W]$.

If $F \subset E$ (resp. $W \subset V$), it is noted in $G \setminus F$ (resp. $G \setminus W$) the graph obtained from G by removing the edges of F (resp. nodes of W and the edges incident to W). If F (resp. W) is reduced to a single edge e (resp. a single vertex v), we write $G \setminus e$ (resp. $G \setminus v$). Let $W \subseteq V$, $\emptyset \neq W \neq V$, a subset of vertices of V. The set of edges having one end in W and the other in $V \setminus W$ is called *cut* and noted $\delta(W)$. By setting $\overline{W} = V \setminus W$, we have that $\delta(W) = \delta(\overline{W})$. If W is reduced to a single vertex v, we write $\delta(v)$. The cardinality of the cut $\delta(W)$ of a subset W is called the degree of W and noted $d(W)$. Given W and W' two disjoint subsets of V, then $[W, W']$ represents the set of edges of G which have one end in W and the other in W'. We denote by $N(W) \subseteq (V \setminus W)$ the set of vertices adjacent to at least one vertex in W. If W is reduced to a single vertex v, we write $N(v)$.

An edge $e = v_1v_2 \in E$ is called a *cut edge* if G is connected and $G \setminus e$ is not connected, with $v_1, v_2 \in V$.

If $\{V_1, \ldots, V_p\}, p \geq 2$, is a partition of V, then $\delta(V_1, \ldots, V_p)$ is the set of edges having one end in V_i and the other one in V_j and $i \neq j$.

The *support graph* of an inequality is the graph induced by the vertices of variables having a non-zero coefficient in the inequality.

Let $G = (V \cup T, E)$ be a graph defined by a set of vertices $V \cup T$ where T is a set of distinguished nodes and E is a set of edges. We denote by $V(H)$, $T(H)$ and $E(H)$ its sets of nodes, terminals and edges, respectively. We denote by $t(G)$ the number of terminal in G, *i.e.*, $|T(G)| = t(G)$. In all Figures of this manuscript, the terminals are represented by triangles.

A *path* P is a set of p distinct vertices $v_1, v_2, \ldots, v_p$ such that for all $i \in \{1, \ldots, p-1\}$, v_iv_{i+1} is an edge. P is called *elementary* if it passes more than once by the same node (except for v_0 and v_k if they represent the same vertex in G). A basic chain is totally identified with its set of edges.

Two paths between two nodes u and v are called edge-disjoint (resp. node-disjoint) if there is no edge (resp. no node different from of u and v) appearing in both chains.

Vertices $v_2, \ldots, v_{p-1}$ are called *the internal vertices* of P. Given a path P between two terminals $t, t' \in T$ such that $P \cap T = \{t, t'\}$, the set of internal vertices of P will be called *a terminal path* and denoted by $P_{tt'}$. A terminal path is *minimal* if it does not strictly contain a terminal path.

Given a graph $G = (V \cup T, E)$ and two subgraphs $G_1 = (V_1 \cup T_1, E_1)$ and $G_2 = (V_2 \cup T_2, E_2)$ of G. Graph G_1 is included in G_2, if $V_1 \cup T_1 \subseteq V_2 \cup T_2$.

1.5.2 Directed graphs

A directed graph is denoted $D = (V, A)$ where V is the set of nodes and A the set of arcs.

If $a \in A$ is an arc connecting a vertex u to vertex v, then u will be called initial end and v final end and we write $a = (u, v)$. We say that a is an outgoing arc of u and v of an incoming arc. The vertices u and v are called ends of a. Vertex v (resp. a) is said to be incident to a (resp. v) if v is an end (initial or final) of a.

If $B \subseteq A$ is a subset of arcs, then $V(B)$ represents the node set of arcs of B. If $W \subseteq V$ is a subset of vertices, $A(W)$ is the set of arcs having their ends in W.

A subgraph $H = (U, F)$ of D is a graph such that $U \subseteq V$ and $F \subset A$. A subgraph $H = (U, F)$ of D is said covering if $U = V$.

If $F \subset A$ (resp. $W \subset V$), we denote by $D \setminus F$ (resp. $D \setminus W$) the graph obtained from D by removing the F arcs (resp. node of W and edges incident to W). If F (resp. W) is reduced to a single arc a (resp. a single vertex v), we write $D \setminus a$ (resp. $D \setminus v$).

Let $W \subseteq V$, $\emptyset \neq W \neq V$, a subset of vertices V. The set of arcs having their initial end in W and their final nodes in $V \setminus W$ is called outgoing cut and denoted $\delta^+(W)$. The cardinality of the outgoing cut $\delta^+(W)$ of a subset W is called outgoing degree of W and denoted $d^+(W)$. If $u \in W$ and $v \in V \setminus W$, then the outgoing cut is also called uv-outgoing cut. If W is reduced to a single vertex v, we write respectively $\delta^+(v)$ and $d^+(v)$ instead of $\delta^+(\{v\})$ and $d^+(\{v\})$. The set of arcs having the final end in W and the initial end in $V \setminus W$ is called incoming cut and denoted $\delta^-(W)$. The cardinality of the incoming cut $\delta^-(W)$ of a subset W is called incoming degree of W and denoted $d^-(W)$. If $u \in W$ and $v \in V \setminus W$, then the incoming cut is also known as uv-incoming cut. If W is reduced to a single vertex v, we write respectively $\delta^-(v)$ and $d^-(v)$ instead of $\delta^-(\{v\})$ and $d^-(\{v\})$.

The cut of a set $W \subseteq V, \emptyset \neq W \neq V$, is denoted $\delta(W)$ and is the union of the arcs of the incoming cut and outgoing cut, *i.e.,* $\delta(W) = \delta^+(W) \cup \delta^-(W)$. The cardinality of the cut is called the degree of W and denoted $d(W)$. If $u \in W$ and $v \in V \setminus W$, then the cut is also called uv-cut. If W is reduced to a single vertex v, we write respectively $\delta(v)$ and $d(v)$ instead of $\delta(\{v\})$ and $d(\{v\})$. If all W associated with the outgoing cut $\delta^+(W)$ contains the vertex u but not the vertex v, then we call it uv-outgoing cut.

Given disjoint subsets $W_1, W_2, \ldots, W_k$ of V, then $[W_1, W_2, \ldots, W_k]$ represents the set of arcs of D having one end in W_i and the other in W_j, $i \neq j$.

A directed graph $D = (V, A)$ is said to be *k-connected graph* if $d^-(W) \geq k$ for all $W \subseteq V, \emptyset \neq W \neq V$. A vertex $v \in V$ is called *cut vertex* of D if the number of connected components of the graph $D \setminus v$ is strictly greater than the number of related components of D.

If a graph $D = (V, A)$ does not contain circuit, then D is said acyclic.

1.6 State-of-the-art on the vertex separator problem

Graph partitioning has become a flourishing area and many optimization problems have been considered in the literature. In this section we discuss two important families of graph partitioning problems, the first one is based on deletion of vertices and the second one on deletion of edges.

1.6.1 Deletion of vertices

The following classes of problems are based on deletion of vertices in order to obtain several disjoint components with some specific properties.

1.6.1.1 The k-way vertex cut problem

Given a graph $G = (V, E)$ and a positive integer k, the *k-way vertex cut problem* consists in finding a subset $S \subseteq V$ of minimum size such that $(V \setminus S, E)$ has at least k disjoint components. In [91], Marx investigates the fixed-parameter (in)tractability of k-way vertex cut. He showed that k-way vertex cut is W[1]-hard with parameter k. In [28], André Berger et al. study the complexity of the k-way vertex cut problem

and its approximation. They show that it is polynomial for bounded tree-width graphs and they present an efficient polynomial-time approximation scheme (EPTAS) for the problem on planar graphs.

In [42], Cornaz et al., consider the balanced version of the k-way vertex cut problem, called *the balanced vertex k-separator problem* defined as follows. Given a graph $G = (V, E)$ and two positive integers k and q, the problem consists in finding a subset $S \subseteq V$ of minimum size, such that the graph $(V \backslash S, E)$ has at least k disjoint subsets $W_1, \ldots, W_k$ and $|W_i| - |W_j| \leq q$ for all $i, j \in \{1, \ldots, k\}$. This problem is NP-hard. The authors present a compact formulation for the problem, investigate it from a polyhedral point of view and present an extended formulation with a polynomial number of constraints. They propose a column generation scheme to solve the linear relaxation. They show that the sub-problem can be solved in a polynomial time and compare the numerical results given by the Branch-and-Cut and the Branch-and-Price algorithms.

1.6.1.2 The k-separator problem

Given a vertex-weighted undirected graph $G = (V, E, w)$ and a positive integer k, the *k-separator problem* consists in finding a minimum-weight subset of vertices whose removal leads to a graph where the size of each connected component is less than or equal to k. When $k = 1$, the problem is equivalent to the vertex cover problem, and for $k = 2$ and unit weight the problem is equivalent to computing the dissociation number of a graph [112]. Thus, the problem is NP-hard even if the graph is bipartite. For any value of k, the problem was considered in [98]. Oosten et al. propose an extended formulation for the problem, they investigate the related polytope and present several valid inequalities and facets. In [26], Ben-Ameur et al. study the complexity of the problem and show that it is polynomial for several classes of graphs, for instance, bounded treewidth, mK_2-free, (G_1, G_2, G_3, P_6)-free, interval-filament, etc. They also present some approximation algorithms and polyhedral results related to the associated polytope.

1.6.1.3 The vertex separator problem

The *vertex separator problem* VSP is very similar to the k-separator problem. Given a simple graph $G = (V, E)$ and a positive integer $\beta(|V|)$, the problem consists in partitioning V into three sets A,B and C such that $|C|$ is minimum, there is no edge between A and B and $\max\{|A|, |B|\} \leq \beta(|V|)$. The VSP is NP-hard [60, 35], even if G is a planar graph [59]. The VSP has been the subject of extensive research and the

first polyhedral approach was done in 2005 by Balas and De Souza [14, 48]. Indeed, they show that the problem is polynomially solvable when $\beta(n) \geq n-1$ and when $\beta(n) = n-k$ for some positive constant k, and the problem is trivial for $\beta(n) = 1$. In [14], Balas and de Souza, present a formulation for the problem and study it from a polyhedral point of view. They give a mixed integer programming formulation of the VSP and a partial characterization of the the VS polytope. In particular, they present a class of valid inequalities based on the minimal connected dominator. They also develop a Branch-and-Cut algorithm. In [32], Didi Biha and Meurs study the vertex separator problem from a polyhedral point of view. They introduce new classes of valid inequalities for the associated polyhedron, a new natural lower bound α_{min} for the optimal solution and present some computational experiments.

1.6.1.4 The multi-terminal vertex separator problem

The *multi-terminal vertex separator problem* MTVSP, also called *vertex multi-terminal cut problem*, or *node multi-way cuts problem* in the literature, is a variant of the k-way vertex cut problem. Indeed, if the original graph G has k specific vertices, *called terminals*, the problem consists in finding a subset of non-terminal vertices S of minimum weight such that $G \setminus S$ has k components, each one contains exactly one terminal. The problem can be seen in another way. Given a graph $G = (V, E)$ and subset $T \subset V$ of k terminals, the problem consists in finding a subset of vertex $S \subseteq V \setminus T$ of minimum size such that each path between two terminals intersects S.

In [63], Garg et al., consider the two versions of the problem (vertex and edge) in directed and undirected graphs. The authors show that the node version is at least as hard as the edge version. Since the edge version is NP-hard by [46], the node version is also NP-hard, even if all weights are 1. They obtained a polynomial-time $(2-2/k)$-approximation algorithm. In [91], Marx obtained a $4^{l^3}n^{O(1)}$-time algorithm for the problem in general graphs. Recently, Chen et al. [37] and Guillemot [69] improved Marx's result to $4^l n^{O(1)}$. In [111], Xiao et al., consider the node and edge version of the problem. They show that the two problems are NP-hard for $k \geq 3$ and polynomial when $k = 2$. They also show that the decision problem of the vertex multi-terminal cut problem can be solved in $O(k^l T(n, m))$ such that $T(n, m) = O(\min(n^{2/3}, m^{1/2})m)$ is the time for finding a cut in the underweighted graph, where $n = |V|$, $m = |E|$ and l is an upper bound on S. In [37], Chen et al., define the edge version of the problem and propose a polynomial algorithm for the problem when the separator is bounded by $O(logn)$. In [45], the authors consider also the vertex version of the problem. In [95], the authors give a linear system for the MTVSP and characterize the class of graphs for which it is total dual integral for any independent set T.

A similar version of this problem of the MTVSP, consists in finding a multi-terminal vertex separator S in G, such that the weight of $\delta(S)$ is minimum. When the graph has two terminals, this problem is called the *minimum cut separator problem* MCSP. In [24], the authors consider the MCSP and show that the associated polytope Q is difficult to characterize, since when the weights are negatives, the problem can be reduced to a max-cut problem. Thus, they study the dominant D of Q and give a complete description of D. They also develop a polynomial algorithm to solve the problem based on the maximum flow problem. In [25], the authors present six different formulations for the problem.

1.6.1.5 Vertex multicut problem

Consider a graph $G = (V, E)$, a collection node pairs $L \subseteq V \times V$, called terminals, and an integer $k \geq 0$. The *restricted vertex multi-cut problem* (RVMCP), consists in finding a subset of non-terminal vertices $V_0 \subseteq V$ with $|V_0| \leq k$ such that all pairs of terminals in L are disconnected in $(V \setminus V_0, E)$. The *unrestricted vertex multi-cut problem* (UVMCP) consists in finding a subset of vertices $V_0 \subseteq V$ with $|V_0| \leq k$ such that all pairs of terminals in L are disconnected in $(V \setminus V_0, E)$.

In [101], Papadopoulos show that the RVMCP is NP-hard even for splits graphs (composed of a clique and a stable set). The authors also show that the UVMCP is polynomial for trees, but NP-hard for complete graphs. They show that the RVMCP can be solved in $O(|E|\sqrt{|V|} + |H||V|)$ for co-bipartie graphs and propose a polynomial algorithm for permutation graphs. In [70], Guo et al., give some complexity results for interval and bounded treewidth graphs.

1.6.1.6 Critical nodes problem

Given a graph $G = (V, E)$ and a positive integer k, the critical nodes problem consists in finding a subset of vertices $S \subseteq V$ of size less or equal to k minimizing the number of pairs of nodes connected by a path in $(V \setminus S, E)$. This problem has important applications in telecommunication as evaluating the robustness of a network. Arulselvan et al. [12] considered the problem and show that it is NP-hard for general graphs. They propose an integer linear formulation for the problem and developed an efficient heuristic. They presented several real applications for the problem, like strategic military planning and immunity network (Find the exact number of individuals to vaccinate, to reduce the transfer of the virus into a network). Boginski and Commander [33]

present applications of CNPs in biology, in order to achieve maximum fragmentation of protein–protein interaction graphs through node deletions. A robust optimization model is considered in [56], when the edges have uncertain weights. In [51], Di summa propose a formulation with a polynomial number of vertices and an exponential number of constraints, and devise a Branch-and-Cut algorithm for solving the problem. They propose some valid inequalities and study the problem from a polyhedral point of view. They compute the relaxation of the problem based on the quadratic reformulation of the problem. Dinh et al. [52] propose algorithms for detecting, in a directed graph, what they call node-disruptors and arc-disruptors, *i.e.,* sets of nodes and arcs to be deleted in order to minimize the number of directed connections surviving in the residual graph.

1.6.2 Deletion of edges

The following classes of problems are based on the deletion of edges in order to obtain several disjoint components with some specific properties.

1.6.2.1 The k-way edge cut

The *k-way edge cut*, also referred as *k-cut* or *minimum k-cut*. This problem can be regarded as the "edge-version" of k-way vertex cut: given a graph $G = (V, E)$ and integers k and s, remove at most s edges so as to split the graph into at least k components. Goldschmidt and Hochbaum [66] have shown that k-way edge cut is NP-hard and admits an $n^{O(k^2)}$-time algorithm. Later, Karger and Stein [77] developed a randomized $n^{O(k)}$-time algorithm for this problem. Kami-dori, et al. [76] presented a deterministic $O(n^{4+o(1)k})$-time algorithm which was further improved by Thorup [109]. Saran and Vazirani [105] gave a polynomial time approximation algorithm that finds at most $2s$ edges which deletion splits the graph into at least k components. For dense graphs with $\Omega(n^2)$ edges, Arora et al. [11] developed a polynomial time approximation scheme for k-way edge cut. Recently, Kawarabayashi and Thorup [80] have shown that k-way edge cut is fixed parameter tractable with respect to parameter s, whereas parameterized by k the problem is W[1]-hard [53]. Therefore, this problem, when considered with parameter s, is one of the few problems known where the vertex version is W[1]-hard, but the edge version is FTP. There are several papers that studied k-way edge cut on planar graphs. For $k = 3$, Hochbaum and Shmoys [73] constructed a quadratic time algorithm for k-way edge cut on planar graphs. He [72] gave an improved algorithm for this special case with running time $O(nlogn)$. Kawarabayashi

and Thorup [80] presented an $2^{O(s^2g^2)}n$-time algorithm for graphs of genus g. Their results were recently improved by Chtinis et al. [38].

1.6.2.2 The multi-way cut problem

The *multi-way cut problem*, consists, given a graph $G = (V, E)$, a set of terminal $A \subset V$ and a weight function $c : E \to \mathbb{Z}$, in finding a separator $S \subseteq E$ of minimum weight such that all terminals of A are disconnected in $(V, E \setminus S)$.

The problem was first proposed in T.C. Hu's 1969 book [74]. Many applications proposed for multiway cut include image processing, chip design and parallel and distributed computing. In the literature the problem has several names the *A-cut problem* and the *multi-terminal cuts problem*. When the set of terminals has cardinality k, it is sometimes called the *k-terminal cut problem* or the *k-way cut problem*.

The problem of computing a minimum weight multiway cut was shown to be NP-hard and MAX SNP-hard even if k is fixed and $k \geq 3$ [46]. The authors show that for trees and 2-trees the problem can be solved in polynomial time. They also describe an approximation algorithm. For planar graphs, the problem can be solved in polynomial time for fixed k but is NP-hard when k is unbounded. For general graphs, there is a simple 2-approximation algorithm disconnecting each terminal from the others by a min-cut, but for any fixed $k \geq 3$, the problem is APX-hard (also in [23]). In [40], Chopra and Rao present a formulation for the problem and investigate the associated polyhedron. They present some graph deletion and contraction operations, and several valid inequalities. They give some cases for which the problem is polynomial and develop a Branch-and-Cut algorithm. In [39], Chopra et al., give an extended formulation for the problem and show that the linear relaxation can be solved in polynomial time. They show that the solution of the linear relaxation is always integer for trees. They present some valid inequalities for cyclic graphs and develop a Branch-and-Cut algorithm. In [44], Cunningham et al., propose an approximation algorithm based on a linear relaxation of a given formulation, using path inequalities. In [36], the authors present a new integer linear formulation for the problem and a new approximation algorithm. For trees and 2-trees graphs, there exists a linear algorithm to solve the problem. In [30], Didi biha consider the linear relaxation $LP(G, A)$ of the 3-terminal cuts polyhedron $P(G, A)$. He characterizes the pairs (G, A) for which $LP(G, A)$ is integer. This result was conjectured by Cunningham. In [23], the authors show that there exists a 2-approximation algorithm for the problem.

1.6.2.3 The balanced graph partitioning problem

In [9], Andreev et al., study the balanced graph partitioning problem, it consists, given a graph $G = (V, E)$ with a weight on each edge and two positive integers v, k, in partitioning V into k subsets, such that each subset contains at most $\frac{v.n}{k}$ nodes, minimizing the weight of edges between subsets. For $k = 2$ and $v = 1$ this problem is equivalent to the well-known Minimum Bisection problem for which an approximation algorithm with a polylogarithmic approximation guarantee has been presented in [83]. The first heuristics for Minimum Bisection were given by Kernighan and Lin [81] and subsequently improved in terms of running time by Fiduccia and Mattheyses [57]. If the balance requirement of the Minimum Bisection problem is relaxed, one gets the well-known graph separator problem. For a comprehensive survey of major results in the area the reader is referred to the book Graph Separators with Applications by Rosenberg and Heath [103]. The first non-trivial approximation algorithm for Minimum Bisection is due to Saran and Vazirani [104] who obtained an approximation ratio of $n/2$.

When $v = 1$, the problem is called *the k-way partitioning problem* in [79], the authors study the problem and propose an algorithm for solving the problem based on reducing the graph using matching. Recently, multilevel recursive bisection algorithm, gave a good method to solve the k-way partitioning problem of a graph.

Chapter 2

The multi-terminal vertex separator problem : Polyhedral analysis

Contents

In this chapter we consider the multi-terminal vertex separator problem. We first show that the problem is NP-hard, then we give two integer programming formulations for the problem. For one of these formulations, we investigate the related polyhedron and discuss its polyhedral structure. We describe some valid inequalities and characterize when these inequalities define facets. We also derive separation algorithms for these inequalities. Using these results, we develop a Branch-and-Cut algorithm for the problem, along with an extensive computational study is presented.

Let $G = (V \cup T, E)$ be a simple graph with $V \cup T$ the set of vertices, where T is a set of k distinguished vertices called *terminals*, and E the set of edges. A *multi-terminal vertex separator* in G is a subset of vertices such that the graph induced by $(V \cup T) \setminus S$ consists of k disjoint components, each with exactly one terminal. Given a weight function $w : V \to \mathbb{Z}$, the *multi-terminal vertex separator problem* (MTVSP for short), consists in finding a multi-terminal vertex separator in G of minimum weight. The MTVSP can also be seen as the problem of finding a vertex subset $S \subseteq V$ of minimum weight such that each path between each pair of terminals intersects S. Indeed, if S intersects all paths between each pair of terminals then the graph induced by $(V \cup T) \setminus S$ would have k components. Moreover, each component contains exactly one terminal. In the case where $k = 2$, the MTVSP can be solved in polynomial time [25].

The chapter is organized as follows, In Section 1 we discuss the complexity aspect of the MTVSP. In Section 2 we give the integer programming formulations for the problem and analyze their linear relaxations. Section 3 is devoted to the polyhedral analysis of the MTVSP and description of some valid inequalities. In Section 4 we discuss some graph reduction operations. In Section 5 we describe separation routines for the inequalities described in Section 3 and develop a Branch-and-Cut algorithm for the MTVSP. Our computational results are presented in Section 6, and finally some concluding remarks are given in Section 7. The rest of this Section is devoted to more definitions and notations.

In what follows we consider the following Hypotheses.

2.1- There is no edge between two terminals, otherwise the problem has no solution.

2.2- For every two different terminals $t, t' \in T$, we have that $N(t) \cap N(t') = \emptyset$. Otherwise, all vertices in $N(t) \cap N(t')$ must belong to the separator. In this case we can remove these vertices from the graph.

2.3- For each vertex $v \in V$, there is at least one terminal path containing v. Otherwise, v cannot belong to a minimal separator. In this case we can delete it from the graph. Checking if a node v belongs to a terminal path can be done in polynomial time.

2.4- Graph G is connected. Otherwise, we consider the MTVSP on each component of the graph.

2.5- The weight of each vertex is 1, *i.e.,* $w(v) = 1$ for each vertex $v \in V$. Otherwise, if there exists a vertex v with $w(v) = 0$, then we can add v in the separator and delete it from the graph. If $w(v) \geq 1$ then we can replace v by a clique $K_{w(v)}$ of size $w(v)$ such that each vertex of $K_{w(v)}$ is adjacent to each vertex of $N(v)$.

2.1 Complexity analysis

In [62], Garg et al. show that the MTVSP is NP-hard. In this section we give a simpler proof of this result using a polynomial reduction from the vertex cover problem (VCP). Given a graph $H = (U, E')$, the VCP consists in finding a minimum cardinality subset of vertices $R \subseteq U$ such that each edge of E' is incident to at least one vertex of R. The VCP is a well-known NP-hard problem [60].

Theorem 2.1 *The MTVSP is NP-hard.*

Proof. Consider the VCP on a graph $H = (U, E')$. Let $G = (V_1 \cup V_2 \cup V_3 \cup T, E)$ be the graph obtained from H as follows

- add three vertices t_1, t_2 and t_3 in T.
- for each vertex $u \in U$, add
  - three vertices v_1^u in V_1, v_2^u in V_2 and v_3^u in V_3.
  - three edges $t_1v_1^u$, $t_2v_2^u$ and $t_3v_3^u$ in E.

– two edges $v_1^u v_3^u$ and $v_2^u v_3^u$ in E.

- for each edge $uw \in E'$, add two edges $v_1^u v_2^w$ and $v_1^w v_2^u$ in E.

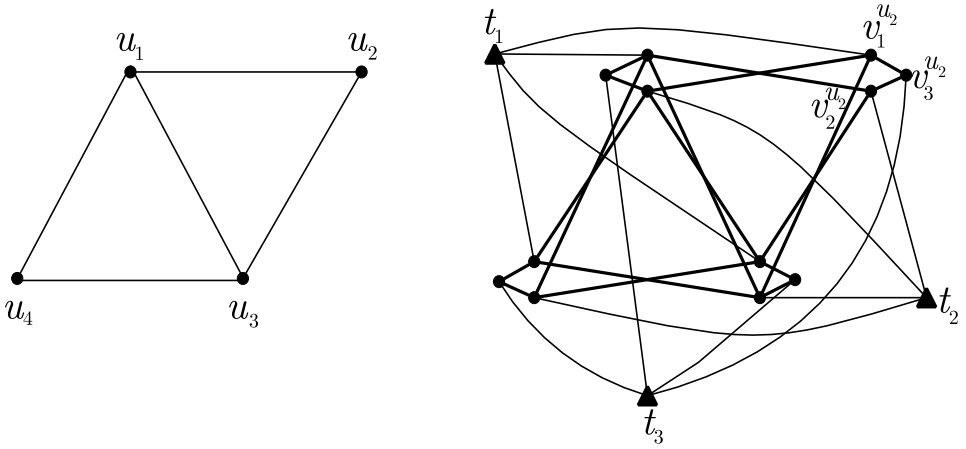


Figure 2.1: Graph transformation from the graph H to the graph G.

Figure 2.1 illustrates the above graph transformation.

Proposition 2.2 *Given a separator S of G and a vertex $u \in U$, either $|\{v_1^u, v_2^u, v_3^u\} \cap S| \geq 2$, $v_3^u \in S$ or both.*

Proof. Let us assume the contrary. Suppose $v_3^u \notin S$ and, say, $\{v_1^u, v_2^u\} \cap S = \{v_1^u\}$. Then, $\{v_2^u, v_3^u\}$ is a terminal path, between t_2 and t_3, not intersecting S, contradicting the fact that S is a separator. ■

Let R be a vertex cover in H and S a separator in G. Let $R^S \subseteq U$ be the set of vertices such that for all $u \in R^S$, $|S \cap \{v_1^u, v_2^u, v_3^u\}| \geq 2$ and let $S^R \subseteq V_1 \cup V_2 \cup V_3$ be defined as follows. For each vertex $u \in U$, if $u \in R$ then we add the two vertices v_1^u, v_2^u in S^R, otherwise we add v_3^u in S^R.

Proposition 2.3 *The set R^S is a vertex cover in H and S^R is a separator in G.*

Proof. Suppose that R^S is not a vertex cover in H. Then, there exists $uw \in E'$ such that $R^S \cap \{u, w\} = \emptyset$. From the construction of R^S and Proposition 2.2 it follows that

$\{v_1^u, v_2^u, v_1^w, v_2^w\} \cap S = \emptyset$ and $\{v_3^u, v_3^w\} \subseteq S$. As consequence, $\{v_1^u, v_2^w\}$ is a terminal path, between t_1 and t_2, not intersecting S, and thus, S is not a separator, a contradiction. Now, suppose that S^R is not a separator of G. Then, there is a terminal path not intersecting S^R in G. We distinguish two cases.

- There exists an edge $uw \in E'$ such that the terminal path $\{v_1^u, v_2^w\}$ or $\{v_1^w, v_2^u\}$, between t_1 and t_2, does not intersect S^R. From the definition of S^R, it follows that $\{u, w\} \cap R = \emptyset$, a contradiction with the fact that R is a vertex cover of H.
- There exists a vertex $u \in U$ such that the terminal path $\{v_1^u, v_3^u\}$, between t_1 and t_3, or $\{v_2^u, v_3^u\}$, between t_2 and t_3, does not intersect S^R. This implies that either $\{v_1^u, v_3^u\} \cap S^R \neq \emptyset$ or $\{v_2^u, v_3^u\} \cap S^R \neq \emptyset$, which is impossible.

■

Proposition 2.4 *If S' is a separator in G with a minimum size, then S' is of size $q + |U|$ where q is the size of the vertex cover $R^{S'}$ in H.*

Proof. First note that, two nodes among $\{v_1^u, v_2^u, v_3^u\}$ suffice to cut all the terminal paths going through these nodes. Hence, as S' is a separator with a minimum size in G, for all $u \in U$, $1 \leq |S' \cap \{v_1^u, v_2^u, v_3^u\}| \leq 2$. By Proposition 2.3, $R^{S'}$ is a vertex cover in H of size q and $S^{R^{S'}}$ is a separator in G. Moreover, $S^{R^{S'}}$ has the same size as S'. From its construction, the size of $S^{R^{S'}}$ is $2q + (|U| - q) = q + |U|$. Hence, that of S' is $q + |U|$. ■

Proposition 2.5 *If R is a vertex cover in H of minimum size, then S^R is a separator in G of minimum size. And if S is a separator in G of minimum size, then R^S is a vertex cover in H of minimum size.*

Proof. Suppose that R is a vertex cover of minimum size q in H but S^R is not of minimum size in G. Note that from the construction of S^R, S^R is of size $q + |U|$. Let $S' \subseteq V_1 \cup V_2 \cup V_3$ be a separator in G of minimum size. From proposition 2.4, $|S'| = q' + |U|$ where q' is the size of $R^{S'}$. Since $|U| + q' < q + |U|$, it follows that $q' < q$ and hence R is not of minimum size in H, a contradiction.
Now, suppose that, S is a separator in G of minimum size $q + |U|$ but R^S, which is of size q, is not of minimum size in H. Let $R' \subseteq U$ be a vertex cover in H of minimum size q'. Then, $S^{R'}$ is a separator in G of size $q' + |U|$. Since $q' < q$, it follows that S is not of minimum size in G, a contradiction. ■

From Proposition 2.5, finding a vertex cover in H of minimum size is equivalent to finding a minimum size three terminal vertex separator in G. Since the vertex cover problem is a NP-hard, the $3-TVSP$ so is. And consequently the MTVSP is NP-hard. ■

2.2 Formulations

In this section we propose two $0-1$ linear formulations for the MTVSP. The first one is a compact formulation with a polynomial number of variables and constraints, and uses double indices. The second has a polynomial number of variables but an exponential number of inequalities.

2.2.1 Double indices formulation

Let $x \in \{0,1\}^{(V \cup T) \times T}$ such that

$$x_{vt} = \begin{cases} 1 & \text{if vertex } v \text{ belongs to subset } V_t, \quad \forall v \in V \cup T, \forall t \in T, \\ 0 & \text{otherwise.} \end{cases}$$

The MTVSP is equivalent to the following integer linear program F^1,

$$\begin{aligned} \min \quad & |V| - \sum_{v \in V} \sum_{t \in T} x_{vt} & & \\ & x_{ut} + \sum_{t' \in T \setminus \{t\}} x_{vt'} \leq 1 & \forall uv \in E, \forall t \in T, & (2.1) \\ & \sum_{t \in T} x_{vt} \leq 1 & \forall v \in V \cup T, & (2.2) \\ & x_{tt} = 1 & \forall t \in T, & (2.3) \\ & x_{vt} \in \{0,1\} & \forall v \in V \cup T, \forall t \in T. & (2.4) \end{aligned}$$

For each pair of terminals $\{t, t'\} \subseteq T$, inequalities (2.1) ensure that there is no edge connecting V_t and $V_{t'}$. Inequalities (2.2) guarantee that each vertex of $V \cup T$ belongs to at most one subset of vertices. Inequalities (2.3) ensure that each terminal belongs to exactly one subset of vertices.

2.2.2 Natural formulation

Let Γ be the set of all terminal paths between the terminals in G. Let $x \in \{0,1\}^V$ such that

$$x(v) = \begin{cases} 1 & \text{if vertex } v \text{ belongs to the separator,} \quad \forall v \in V, \\ 0 & \text{otherwise.} \end{cases}$$

The MTVSP is equivalent to the following integer linear program F^2,

$$\min \quad \sum_{v \in V} w(v)x(v) \tag{2.5}$$

$$\sum_{v \in P_{tt'}} x(v) \geq 1 \qquad \forall P_{tt'} \in \Gamma, \tag{2.6}$$

$$x(v) \leq 1 \qquad \forall v \in V, \tag{2.7}$$

$$x(v) \geq 0 \qquad \forall v \in V, \tag{2.8}$$

$$x(v) \in \{0,1\} \qquad \forall v \in V. \tag{2.9}$$

Inequalities (2.6) guarantee that at least one vertex of each terminal path belongs to the separator.

2.2.3 Comparing the LP-Relaxations

We first present a numerical comparison of the LP-Relaxation values of the two above formulations with the optimal value, for some DIMACS instances [1]. The columns in Table 2.1 represent the name of the instance, the value of the LP-Relaxation of formulation F^1, the value of the LP-Relaxation of formulation F^2 and the value of the optimal separator, respectively.

Instances	k	(2.1) − (2.3)	(2.6) − (2.8)	OPT
DSJR500	10	**32**	18.5	**32**
Games120	10	30.34	18.5	31
Miles250	15	32.18	27.5	35
Myciel6	11	38.34	22.5	40
Myciel7	17	**57**	31	**57**
Queen8_ 12	11	**38**	21.5	**38**
Queen14_ 14	18	**65**	35	**65**
Queen16_ 16	16	**59**	32	**59**

Table 2.1: Comparing LP-Relaxations value of F^1 and F^2 with the optimal one.

It appears from Table 2.1 that the LP-Relaxation of the first formulation is better than the second one. This is shown below.

Proposition 2.6 *$Z_{F^2} \leq Z_{F^1}$, where $Z_{F^1}(Z_{F^2})$ is the optimal value of $F^1(F^2)$.*

Proof. Let assume the contrary, $Z_{F^1} < Z_{F^2}$. Let $\overline{x} \in [0,1]^{V\times T}$(resp. $\overline{y} \in [0,1]^V$) be the optimal solution of F^1(resp. F^2) associated with Z_{F^1}(resp. Z_{F^2}). For $v \in V$, let $y \in [0,1]^V$ such that $y(v) = 1 - \sum_{t\in T} \overline{x}_{vt}$. Clearly, by inequalities (2.2), $y(v) \in [0,1]$. Consider a terminal path $P_{tt'}$ between two terminals t and t' of l edges. By summing inequalities (2.1) associated with the edges of $P_{tt'}$ and t, (by respecting the order of nodes in $P_{tt'}$) we obtain the following inequality

$$\overline{x}_{tt} + \sum_{v\in P_{tt'}} \sum_{t''\in T} \overline{x}_{vt''} + \sum_{t''\in T\setminus\{t\}} \overline{x}_{t't''} \leq l$$

As $\overline{x}_{tt} = 1$ and $\sum_{t''\in T\setminus\{t\}} \overline{x}_{t't''} = 1$, it follows that

$$\sum_{v\in P_{tt'}} y(v) \geq 1,$$

and thus, inequalities (2.6) is satisfied for y. The objective value of y is $Z' = \sum_{v\in V} y(v) = \sum_{v\in V}(1 - \sum_{t\in T} \overline{x}_{vt}) = Z_{F^1}$. Thus, $Z' < Z_{F^2}$, which contradicts the fact that $\overline{y}$ is the optimal solution of F^2. ■

Note that the first formulation has $(n+k)k$ variables and the second one has only n variables. Also, note that inequalities (2.6), which are in an exponential number, can be separated in polynomial time. Since the second formulation has fewer variables and may use less number of inequalities of type (2.6) in the Branch-and-Cut algorithm, we will consider it for our analysis. (For the instance R_1300 in Table 2.3, F^1 has 13100 variables and 3329920 non-trivial inequalities, whereas F^2 has only 1300 variables and uses 122 non-trivial inequalities in the Branch-and-Cut algorithm).

2.3 Polyhedral analysis

Let $P(G,T)$ be the convex hull of the solutions of formulation F^2, that is,

$$P(G,T) = conv\{x \in \{0,1\}^V |\ \ x \textit{ satisfies } (2.6)\}.$$

In this Section, we will discuss $P(G,T)$, we will give its dimension, identify several classes of valid inequalities and describe necessary and sufficient Conditions for these inequalities to be facet defining.

2.3.1 Dimension

We first establish the dimension of $P(G,T)$.

Theorem 2.7 *$P(G,T)$ is full dimensional.*

Proof.

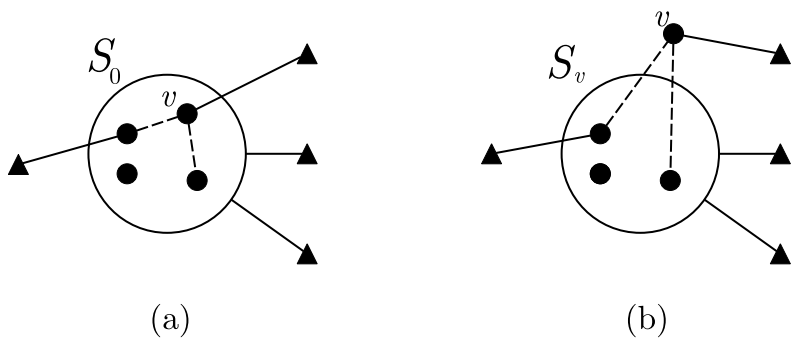


Figure 2.2: Two separators S_0 and S_v.

We need to exhibit $n+1$ separators such that their incidence vectors are affinely independent. Let $S_0 = V$. Clearly, S_0 is a separator of G. For each $v \in V$, let $S_v = V \setminus \{v\}$. By Hypothesis 2.2, v is not adjacent to two terminals. It then follows that S_v is a separator. (Figure 2.2 illustrates the two above separators.) This constitutes a set of $n+1$ separators of G. Moreover, their incidence vectors are affinely independent. ■

Now we characterize when inequalities (2.6),(2.7) and (2.8) define facet of $P(G,T)$.

Theorem 2.8 *For $v \in V$, inequality $x(v) \leq 1$ defines a facet of $P(G,T)$.*

Proof. We need to exhibit n separators containing v such that their incidence vectors are affinely independent. Let $S_0 = V$, and for each vertex $u \in V \setminus \{v\}$, let $S_u = V \setminus \{u\}$. Clearly, these sets are separators in G. Moreover, their incidence vectors are affinely independent. ■

Theorem 2.9 *For $v \in V$, inequality $x(v) \geq 0$ defines facet of $P(G,T)$ if and only if v does not belong to a terminal path $P_{tt'}$ containing exactly two internal vertices.*

Proof. ($\Rightarrow$) Suppose there exists a terminal path $P_{tt'}$ containing exactly two internal vertices u and v. Then, we have the valid inequalities $x(u) + x(v) \geq 1$ and $x(u) \leq 1$. Inequality $x(v) \geq 0$ can then be obtained as a linear combination of these inequalities, and cannot then define a facet.
($\Leftarrow$)

Let $S_0 = V \setminus \{v\}$ and $S_u = S_0 \setminus \{u\}$ for all $u \in V \setminus \{v\}$. Since v does not belong to a terminal path of two vertices, these sets are separators of G. Moreover, their incidence vectors are affinely independent. ■

2.3.2 Path inequalities

Theorem 2.10 *Inequality* (2.6)*, associated with a terminal path* $P_{tt'}$*, defines a facet of* $P(G,T)$ *if and only if*

(a) $P_{tt'}$ *is minimal.*

(b) No vertex $v \in V \setminus P_{tt'}$ *is adjacent to a terminal* $t'' \in T \setminus \{t,t'\}$ *and to two vertices of* $P_{tt'}$*.*

Proof. $(\Rightarrow)$

(a) If $P_{tt'}$ is not minimal, then there exists a non-terminal vertex $v \in P_{tt'}$ adjacent to a terminal $t'' \in T \setminus \{t,t'\}$. Inequality (2.6) can then be obtained by summing inequality (2.6) associated with $P_{t''t'}$ and inequalities (2.8) associated with the vertices of $P_{tt'} \setminus P_{t''t'}$. Hence, inequalities (2.6) cannot define a facet.

(b) Suppose there exists $v \in V \setminus P_{tt'}$ which is adjacent to $t'' \in T \setminus \{t,t'\}$ and to two vertices of $P_{tt'}$. Thus, we have the following valid inequalities, $x(P_{tt'}) \geq 1$, $x(P_{tt''}) \geq 1$, $x(P_{t't''}) \geq 1$ and $x(u) \geq 0$ for all $u \in P_{tt'} \setminus (P_{tt''} \cup P_{t't''})$. By summing these inequalities, devising by 2 and rounding up the right hand side, we obtain the valid inequality $x(P_{tt'}) + x(v) \geq 2$. Inequality (2.6) associated with terminal path $P_{tt'}$ can then be obtained from the above inequality and inequality (2.7) associated with v, and it cannot therefore be facet defining.

$(\Leftarrow)$ Denote by $ax \geq \alpha$ inequality (2.6). Let $bx \geq \beta$ be an inequality that defines a facet of $P(G,T)$. Suppose that $\{x \in P(G,T) : ax = \alpha\} \subseteq \{x \in P(G,T) : bx = \beta\}$. We will show that there exists ρ such that $b = \rho a$.

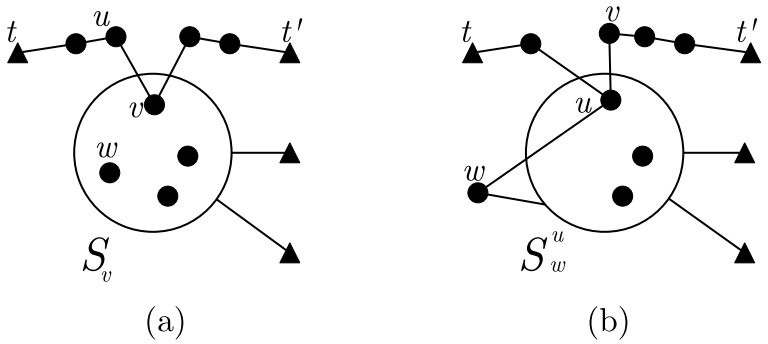


Figure 2.3: Two separators S_v and S_w^u.

For $v \in P_{tt'}$, let $S_v = (V \setminus P_{tt'}) \cup \{v\}$. Figure 2.3.(a) represents the set S_v. By Condition (a), each terminal path contains v or a vertex from $V \setminus P_{tt'}$. Thus, S_v is a separator of G. Moreover, $ax^{S_v} = \alpha$. Hence, $bx^{S_u} = bx^{S_v}$ for every $u, v \in P_{tt'}$. Therefore,

$$b(u) = b(v) = \rho \quad \text{for all } u, v \in P_{tt'} \text{ and some scalar } \rho \in \mathbb{R}.$$

For $w \in V \setminus P_{tt'}$, let $u \in P_{tt'}$ be a vertex adjacent to w. If there is no vertex in $P_{tt'}$ adjacent to w, then u is arbitrarily chosen in $P_{tt'}$. Set $S_w^u = S_u \setminus \{w\}$. From Condition (b) we have that S_w^u is a separator of G. (Figure 2.3.(b) represents the set S_w^u.) Moreover, the incidence vector of S_w^u satisfies inequality (2.6) with equality. Hence, $bx^{S_u} = bx^{S_w^u}$, and thus, $b(w) = 0$. We then obtain that

$$b(w) = 0 \quad \text{for all } w \in V \setminus P_{tt'}.$$

Therefore, we have that $b = \rho a$, which ends the proof. ■

In what follows we describe further classes of valid inequalities of $P(G, T)$ and discuss their facial structure.

2.3.3 Star tree inequalities

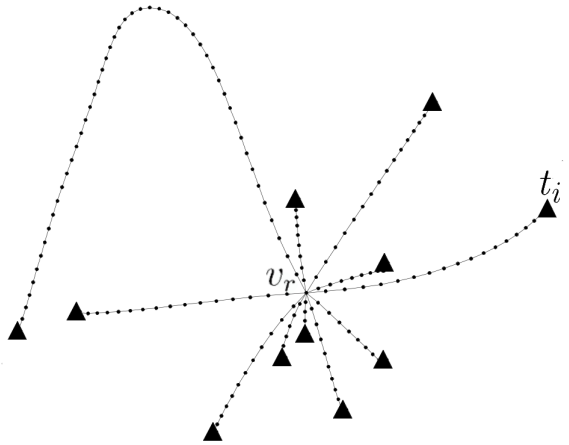


Figure 2.4: A Star tree.

A *star tree* $H = (V(H) \cup T(H), E(H))$ of G is a tree where the pending nodes are the terminal nodes of the tree, and all the other (non-terminal) nodes, different from the root node, are of degree two. The star tree H with q terminals $t_1, \ldots, t_q$, can also be seen as the concatenation of q paths $P_{t_i},\ i = 1, \ldots, q$ between the root v_r and $t_i,\ i = 1, \ldots, q$. The paths $P_{t_1}, \ldots, P_{t_q}$ will be called *branches*. A star tree with 2 terminals is nothing but a terminal path. Figure 2.4 displays a star tree with 11 branches.

Theorem 2.11 *If $H = (V(H) \cup T(H), E(H))$ is a star tree of G with root v_r, then the inequality*

$$x(V(H) \setminus \{v_r\}) + (q-1)x(v_r) \geq q - 1 \tag{2.10}$$

is valid for $P(G,T)$.

Proof. Given a separator S of G. if v_r belongs to S, then $x(v_r) = 1$ and the inequality is satisfied by x^S. If v_r is not in S, then S must contain at least $q-1$ nodes of $V(H)\setminus\{v_r\}$ in order to cut all the terminal paths of H. ■

Lemma 2.12 *If there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$ and to an internal vertex in each branch of H, then the following inequality*

$$x(V(H) \setminus \{v_r\}) + (q-1)x(v_r) + x(w) \geq q \tag{2.11}$$

is valid for $P(G,T)$.

Proof. Consider a separator S. If $w \in S$, then either $v_r \in S$ or $q-1$ vertices of $V(H) \setminus \{v_r\}$ belong to S. In both cases, inequality (2.11) is satisfied by x^S. If $w \notin S$, then suppose that S contains v_r or $q-1$ vertices of $V(H) \setminus \{v_r\}$. In both cases, there exists a terminal path between t' and at least one terminal of $T(H)$ not intersecting S. Thus, S contains another vertex from $V(H)$. And hence inequality (2.11) is again satisfied. ■

Lemma 2.13 *If there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$, and to two different vertices $u_1, u_2 \in P_t$, for some $t \in T(H)$, the following inequality*

$$x(V(H) \setminus \{v_r\}) + (q'-2)x(v_r) + x(w) \geq q' - 1 \tag{2.12}$$

is valid for $P(G,T)$, where $q' = q+1$.

Proof. We can suppose that $u_1 \neq v_r$. Given a separator S of G, if v_r belongs to S, then S must contains another vertex $v \in P_{tt'} \subseteq P_t \cup \{w\}$, where $P_{tt'}$ is the terminal path between t and t' not containing v_r. Then, the inequality (2.12) is satisfied by x^S. If v_r is not in S, then S contains $q-1$ vertices from $V(H) \setminus \{v_r\}$. Since w is adjacent to t' and to two vertices of P_t, S must contain another vertex from $(V(H) \cup \{w\}) \setminus \{v_r\}$. And hence the inequality (2.12) is satisfied by x^S. ■

Theorem 2.14 *Given a star tree $H = (V(H) \cup T(H), E(H))$ of G, inequality* (2.10) *defines a facet of $P(G,T)$ if and only if the following hold*

(a) There is no vertex of $V(H)$ adjacent to a terminal of $T \setminus T(H)$.

(b) The subgraph induced by $V(H)$ in G contains no cycle.

(c) There is no vertex of $V \setminus V(H)$ adjacent to a terminal of $T \setminus T(H)$, and to two vertices of the same branch.

(d) There is no vertex of $V \setminus V(H)$ adjacent to a terminal of $T \setminus T(H)$ and to an internal vertex of each branch of H.

(e) v_r is not adjacent to a terminal of $T(H)$.

Proof. ($\Rightarrow$)

(a) If a vertex $u \in P_t \setminus \{v_r\}$, for some $t \in T(H)$, is adjacent to a terminal $t' \in T \setminus T(H)$, then consider star tree H' in H with leaf set $T(H') = T(H) \setminus \{t\}$. Inequality (2.10) associated with H can be obtained from the star tree inequality associated with H', the inequality associated with the terminal path $P_{tt'}$ between t and t' and the trivial inequalities $x(w) \geq 0$ for all w of $P_t \setminus P_{tt'}$. If v_r is adjacent to a terminal $t' \in T \setminus T(H)$, then consider the star tree H'' where $T(H'') = T(H) \cup \{t'\}$. Inequality (2.10), associated with H, can then be obtained by summing the star tree inequality induced by H'' and the trivial inequality $-x(v_r) \geq -1$. So in both cases, inequality (2.10) can be obtained as a linear combination from valid inequalities and they cannot be facet defining.

(b) Suppose that the subgraph induced by $V(H)$ in G contains a cycle. It follows that there exist two vertices $u, v \in V(H)$ such that $uv \in E \setminus E(H)$. We distinguish two cases

Case 1. $u \in P_t$ and $v \in P_{t'}$ for two different terminals $t, t' \in T(H)$. Then, the following inequality

$$x(V(H) \setminus \{v_r\}) + (q-2)x(v_r) \geq q-1 \tag{2.13}$$

is valid for $P(G,T)$ and dominates inequality (2.10) associated with H. Indeed, given a separator S of G, if v_r belongs to S, then S must contains another vertex $w \in P_{tt'}$, where $P_{tt'}$ is the terminal path between t and t' not containing v_r. Then, the inequality is satisfied by x^S. If v_r is not in S, then S must contain at least $q-1$ nodes of $V(H) \setminus \{v_r\}$ in order to cut all the terminal paths of H. And the inequality is again satisfied. Thus, inequality (2.10) cannot be facet defining. It will be shown later in Section 2.3.5 that inequality (2.13) can be obtained as a lifted inequality from the star tree inequality (2.10).

Case 2. $u, v \in P_t$ for some $t \in T(H)$. Let $P_{uv} \subset P_t$ be the set of the internal vertices of the path between u and v, in the branch P_t. Then, $((V(H) \setminus P_{uv}) \cup T(H), E(H))$ is also a star tree. Then, the following inequality

$$x(V(H) \setminus (P_{uv} \cup \{v_r\})) + (q-1)x(v_r) \geq q-1$$

is a star tree inequality that is valid for $P(G,T)$. However, this inequality dominates inequality (2.10) associated with H. Therefore, the latter cannot be facet defining.

(c) Suppose there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$, and to two different vertices $u_1, u_2 \in P_t$, for some $t \in T(H)$. From Lemma 2.13, the

following inequality

$$x(V(H) \setminus \{v_r\}) + (q' - 2)x(v_r) + x(w) \geq q' - 1 \tag{2.14}$$

is valid for $P(G,T)$, where $q' = q + 1$. Hence, inequality (2.10) can be obtained by summing inequality (2.14) and $-x(w) \geq -1$ and replacing q' by $q + 1$. Thus, inequality (2.10) cannot be facet defining.

(d) Suppose there is a vertex $w \in V \setminus V(H)$ adjacent to a terminal $t' \in T \setminus T(H)$ and to an internal vertex in each branch of H. From Lemma 2.12 the following inequality

$$x(V(H) \setminus \{v_r\}) + (q - 1)x(v_r) + x(w) \geq q \tag{2.15}$$

is valid for $P(G,T)$. Inequality (2.10) can be obtained by summing inequality (2.15) and $x(w) \leq 1$. Therefore, inequality (2.10) cannot be facet defining.

(e) If there exists a branch P_t with no internal vertices, (that is to say, v_r is adjacent to a terminal of $T(H)$), then let H' be the star tree obtained from H by deleting a branch $P_{t'}$ with at least one internal vertex. Inequality (2.10) can be obtained by summing the star tree inequality associated with H' together with the terminal path inequality associated with $P_t \cup P_{t'}$.

($\Leftarrow$) Suppose that Conditions (a)-(e) hold. Denote by $ax \geq \alpha$ inequality (2.10). Let $bx \geq \beta$ be an inequality that defines a facet of $P(G,T)$. Suppose that $\{x \in P(G,T) : ax = \alpha\} \subseteq \{x \in P(G,T) : bx = \beta\}$. We will prove that there exists ρ such that $b = \rho a$.

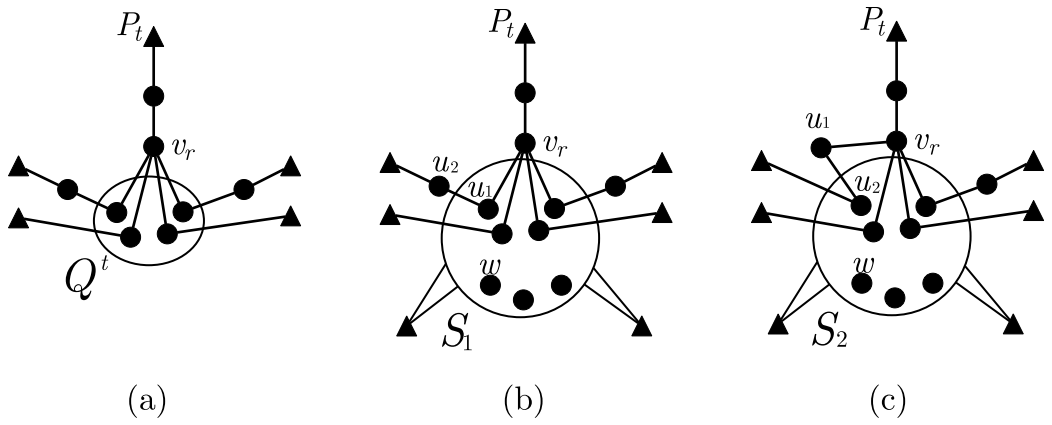


Figure 2.5: A ring Q^t and two separators S_1 and S_2.

By Condition (e), it follows that each branch P_t, $t \in T(H)$, contains at least one internal node. For a terminal $t \in T(H)$, a *ring* of $V(H) \setminus P_t$ is a subset of $q-1$ vertices containing exactly one vertex from each branch different from P_t. Hence, if Q^t is a ring of $V(H) \setminus P_t$ then for all $t' \in T(H) \setminus \{t\}, |(P_{t'} \setminus \{v_r, t'\}) \cap Q^t| = 1$, see Figure 2.5.(a). Consider two vertices $u_1, u_2 \in V(H) \setminus \{v_r\}$ that belong to different branches P_{t_1}, P_{t_2} of H. Let $Q_1^{t_1}, Q_2^{t_2}$ be two rings such that $u_1 \in Q_1^{t_1}$, $u_2 \in Q_2^{t_2}$ and $Q_1^{t_1} \setminus \{u_1\} = Q_2^{t_2} \setminus \{u_2\}$. Let $S_1 = (V \setminus V(H)) \cup Q_1^{t_1}$ and $S_2 = (V \setminus V(H)) \cup Q_2^{t_2}$. See Figure 2.5.(b) and Figure 2.5.(c). Sets S_1 and S_2 are separators in G. Indeed, by Conditions (a) and (b) each terminal path between a terminal in $T(H)$ and a terminal in T, contains a vertex of $Q_i^{t_i}$ with $i = 1, 2$ or it contains a vertex of $V \setminus V(H)$. Hence, S_1 and S_2 are two separators of G. Moreover, incidence vectors x^{S_1} and x^{S_2} satisfy inequality (2.10) with equality. Hence, $bx^{S_1} = bx^{S_2}$, implying that $b(u_1) = b(u_2)$. As t, u_1 and u_2 are arbitrarily chosen, we have that

$$b(u) = b(v) = \rho \qquad \text{for all } u, v \in V(H) \setminus \{v_r\} \text{ and a scalar } \rho \in \mathbb{R}.$$

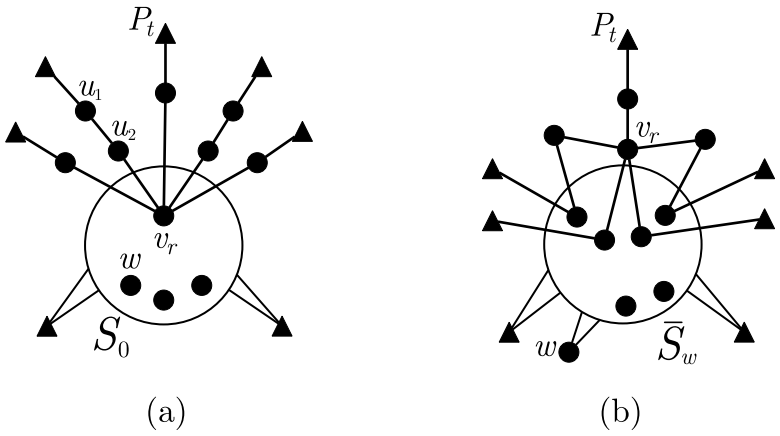


Figure 2.6: Two separators S_0 and $\overline{S}_w$.

Set $S_0 = (V \setminus V(H)) \cup \{v_r\}$. By Conditions (a) and (b) each terminal path between a terminal in $T(H)$ and a terminal in T, contains v_r or a vertex from $V \setminus V(H)$. Thus, S_0 is a separator of G. See Figure 2.6.(a) for an illustration. Also, we have that $ax^{S_0} = \alpha$. Hence, $bx^{S_1} = bx^{S_0}$, yielding

$$b(v_r) = \sum_{v \in Q_1^t} b(v) = (q-1)\rho.$$

Now consider a vertex $w \in V \setminus V(H)$. We distinguish two cases.

Case 1. If w is not adjacent to the root vertex v_r, then by Condition (c), w is adjacent to at most one vertex of each branch and by Condition (d), there exists at least one branch P_t such that no internal vertex of P_t is adjacent to w. Consider the ring Q of H such that $N(w) \cap V(H) \subseteq Q$ and each vertex of $Q \setminus (N(w) \cap V(H))$ is adjacent to a terminal of $T(H) \setminus \{t\}$. Let $S' = (V \setminus V(H)) \cup Q$ and $\overline{S}_w = S' \setminus \{w\}$. Since Q is a ring, S' is a separator of G, as defined before. Set $\overline{S}_w$ is also a separator of G. Otherwise, since S' is a separator, w is adjacent to a terminal in $T \setminus T(H)$ and to a vertex in P_t. Contradiction with Condition (d).

Case 2. If w is adjacent to the root vertex v_r then, let $\overline{S}_w = S_0 \setminus \{w\}$. Set $\overline{S}_w$ is a separator of G. Indeed, by Condition (c), since w is adjacent to v_r, w cannot be adjacent to vertices of $V(H) \setminus \{v_r\}$. Therefore, every terminal path between a terminal of $T \setminus T(H)$ and a terminal of T intersects $\overline{S}_w$

In both cases, $\overline{S}_w$ is a separator in G. (See Figure 2.6.(b) for an illustration). And its incidence vector satisfies inequality (2.10) with equality. Hence, $ax^{\overline{S}_w} = ax^{S_0}$, therefore $bx^{\overline{S}_w} = bx^{S_0}$. This implies that $b(w) = 0$. Hence, we have that

$$b(u) = 0 \qquad \text{for all } u \in V \setminus V(H).$$

In consequence we have that $b = \rho a$, and the proof is complete. ■

2.3.4 Clique star inequalities

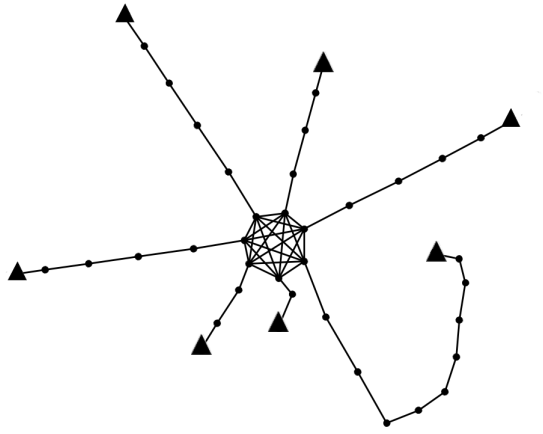


Figure 2.7: Clique star of 7 terminals.

A graph $Q = (V(Q) \cup T(Q), E(Q))$ is called a *clique star* if Q consists of a clique K_q on q vertices of $V(Q)$, q terminals $t_1, \ldots, t_q$ (that is $T(Q) = \{t_1, \ldots, t_q\}$) and q disjoints

paths $P_{t_1}, \dots P_{t_q}$ such that each path P_{t_i} is between a different vertex of K_q and t_i. The paths P_t are called *branches*. Figure 2.7 shows a clique star with 7 branches. In what follows we will suppose that $q \geq 3$, otherwise the clique star is either a terminal path or a branch.

Theorem 2.15 *Given a clique star $Q = (V(Q) \cup T(Q), E(Q))$ of G, the following inequality is valid for $P(G,T)$*

$$x(V(Q)) \geq q - 1. \tag{2.16}$$

Proof. If $S \cap K_q = \emptyset$, then clearly, S must contain at least $q-1$ vertices of $V(Q)$ to cut all the terminal paths of Q. So suppose that $S \cap K_q = \{v_1, \dots, v_l\}$, $l \leq q-2$. (If $l \geq q-1$, then inequality (2.16) is satisfied by x^S). Note that each vertex $v_i \in K_q$ is an extremity of branch P_{t_i}. Hence, $q-l$ terminals remain to separate. For this at least $q-l-1$ vertices from $V(Q) \setminus K_q$ are needed to cut all the terminal paths between the remaining terminals. ■

2.3.5 Lifting procedure for star tree inequalities

In what follows we are going to describe a lifting procedure for the star tree inequalities. This will permit to extend these inequalities to a more general class of valid inequalities. Let $G = (V \cup T, E)$ be a simple graph. Consider a star tree H, of G. Let a *branch clique* $W \subseteq (V(H) \setminus \{v_r\})$ be a clique such that for each pair of vertices $u, v \in W$, u and v belong to different branches of H. Observe that each branch clique yields a clique star in the graph. (See Figure 2.8 for illustration where three branch cliques are displayed).

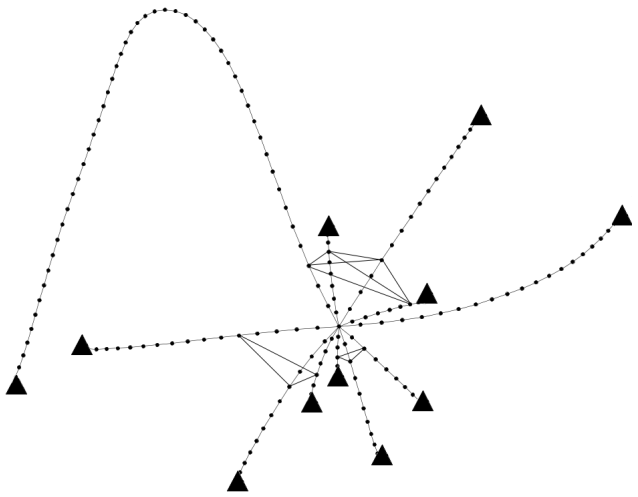


Figure 2.8: Star tree with clique branches.

Theorem 2.16 *Let $\Pi = \{W_1, \ldots, W_s\}$ be a set of vertex disjoint branch cliques in H. Let $\alpha_\Pi = \sum_{i=1}^{s}(|W_i| - 1)$. Then, the inequality*

$$x(V(H) \setminus \{v_r\}) + (q - 1 - \alpha_\Pi)x(v_r) \geq q - 1 \tag{2.17}$$

is valid for $P(G, T)$.

Proof. Given a separator S of G, we distinguish two cases. If $v_r \in S$, then for each clique star Q^{W_i} associated with the branch clique W_i, $x^S(V(Q^{W_i})) \geq |W_i| - 1$ by the clique star inequality (2.7). By adding these inequalities we obtain that $x^S(\bigcup_{i=1}^{p} Q^{W_i}) \geq \alpha_\Pi$. As $\bigcup_{i=1}^{p} Q^{W_i} \subset V(H) \setminus \{v_r\}$, it follows that $x^S(V(H) \setminus \{v_r\}) \geq \alpha_\Pi$. As $v_r \in S$, we have that x^S satisfies inequality (2.17). If $v_r \notin S$, then $x^S(V(H)) \geq q - 1$ by the star tree inequality. Thus, x^S satisfies also inequality (2.17). ■

2.3.6 Terminal tree inequalities

A *terminal tree* $H = (V(H) \cup T(H), E(H))$, is a tree such that the terminals of $T(H)$ are the leaves of H. (See Figure 2.3.6 for illustration). A *leaf branch* P_t of a terminal tree H is a path between a terminal t and a vertex of $V(H)$ of degree greater or equal to 3, and where all internal vertices are of degree 2. For $v \in V(H)$, let $d_H(v)$ be the degree of v in H. If $|T(H)| \leq 3$, the terminal tree is either a star tree or a terminal path.

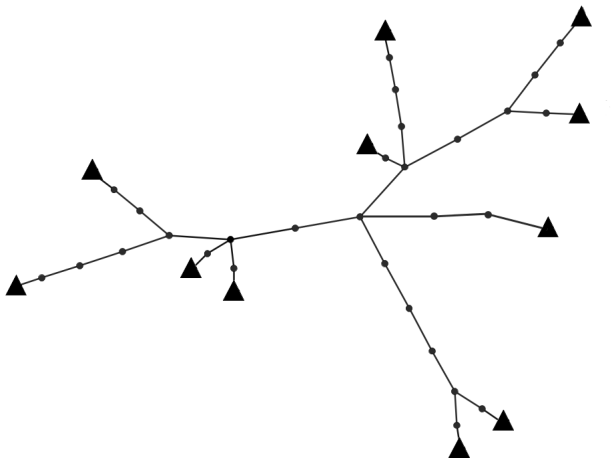


Figure 2.9: Terminal tree of 10 terminals.

Theorem 2.17 *Let $G = (V \cup T, E)$ be a graph and H a terminal tree of G. Let $q = |T(H)|$. Then, the following inequality is valid for $P(G,T)$,*

$$\sum_{v \in V(H)} (d_H(v) - 1)x(v) \geq q - 1. \tag{2.18}$$

Proof. The proof is by induction on q. If $q \leq 3$, inequality (2.18) is nothing but a star tree inequality (2.10) associated with H and then it is valid. Let us suppose $q \geq 3$, and that for each terminal tree with less than $q-1$ terminals, the associated inequality (2.18) is valid for $P(G,T)$. Since $q \geq 3$, there must exist a vertex u belonging to two leaf branches P_t and $P_{t'}$. Let us consider the terminal tree H^1 (resp. H^2) obtained from H by removing the vertices of $P_t \setminus \{u\}$ (resp. $P_{t'} \setminus \{u\}$). Thus, H^1 and H^2 have each one $q-1$ leaves and hence $q-1$ terminals. By the induction Hypothesis, the terminal tree inequalities (2.18) associated with H^1 and H^2

$$\sum_{v \in V(H) \setminus (P_t \cup P_{t'})} (d_H(v) - 1)x(v) + \sum_{v \in P_{t'} \setminus \{u\}} (d_H(v) - 1)x(v) + (d_H(u) - 2)x(u) \geq q - 2,$$
$$\sum_{v \in V(H) \setminus (P_t \cup P_{t'})} (d_H(v) - 1)x(v) + \sum_{v \in P_t \setminus \{u\}} (d_H(v) - 1)x(v) + (d_H(u) - 2)x(u) \geq q - 2.$$

are valid for $P(G,T)$. By summing these inequalities together with the inequality $x(P_t \cup P_{t'}) \geq 1$ induced by the terminal path $P_t \cup P_{t'}$, and inequality $x(u) \geq 0$, we obtain the inequality

$$\sum_{v \in V(H) \setminus (P_t \cup P_{t'})} 2(d_H(v) - 1)x(v) + \sum_{v \in P_t \cup P_{t'}} (2(d_H(v) - 1))x(v) \geq 2q - 3. \tag{2.19}$$

is valid for $P(G,T)$. By dividing by 2 and rounding up the right hand side, we obtain inequality (2.19). ■

Remark 2.18 *If the terminal tree contains some terminals that are not leaves, then the associated inequality (2.18) remains valid.*

2.3.7 Lifted terminal tree inequalities

Let R be a terminal tree of G. For a vertex v, let F_v be the union of leaf branches incident to v. Clearly, F_v is a star tree in R. Let Π_{F_v} be a set of vertex disjoint branch cliques in F_v (see, lifted star tree inequalities, subsection 2.3.5).

Theorem 2.19 *The following inequality*

$$\sum_{v \in V(R)} (d_R(v) - 1 - \alpha_{\Pi_{F_v}})x(v) \geq q - 1$$

if valid for $P(G,T)$.

Proof. The proof is similar to the one of Theorem 2.16. ■

2.3.8 Terminal cycle inequalities

A *terminal cycle* $J = (V(J) \cup T(J), E(J))$, where $T(J)$ is a set of q terminals is a graph given by a cycle C of q vertices and q disjoint edges between the vertices of C and the terminals of $T(J)$. (See Figure 2.10 displaying a terminal cycle where $q = 5$).

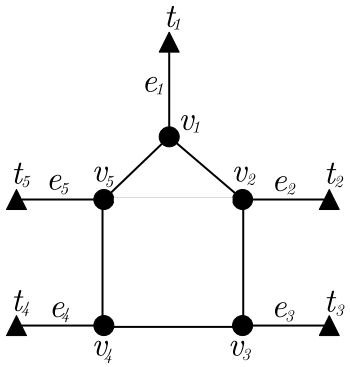


Figure 2.10: Terminal cycle of 5 terminals.

Theorem 2.20 *If $J = (V(J) \cup T(J), E(J))$ is a terminal cycle of G, then the following inequality is valid for $P(G,T)$,*

$$x(C) \geq \lceil \frac{q}{2} \rceil. \tag{2.20}$$

Proof. Let S be a separator in G. Suppose that there exists an edge $uv \in C$ such that $S \cap \{u, v\} = \emptyset$. Since u and v are connected to two different terminals, there is a terminal path not intersecting S, a contradiction. So, for each edge $e \in C$, at least one vertex of e belongs to S. Thus, the result follows. ■

Theorem 2.21 *Given a terminal cycle $J = (V(J) \cup T(J), E(J))$ of G, with $|T(J)| = q$, inequality (2.20) defines a facet of $P(G,T)$ if and only if the following hold.*

(a) q is odd.

(b) C is chordless.

(c) For $w \in V \setminus C$ not adjacent to a terminal, there exists a vertex cover of C of size $\lceil \frac{q}{2} \rceil$ containing at least $|C'| - 1$ vertices of C', where $C' = N(w) \cap C$.

(d) For $w \in V \setminus C$ adjacent to a terminal, there exists a vertex cover in C of size $\lceil \frac{q}{2} \rceil$ containing C', where $C' = N(w) \cap C$.

Proof. Let $t_1, \ldots, t_q$ be the terminals of $T(J)$, $v_1, \ldots, v_q$ the nodes of C and $e_i = v_i t_i$, for $i = 1, \ldots, q$. For $i = 1, \ldots, q-1$, we will let $\overline{P}_{t_i}$ denote the terminal path $(t_i, v_i, v_{i+1}, t_{i+1})$.
($\Rightarrow$)

(a) If q is even, the terminal cycle inequality associated with J can be obtained by summing the terminal path inequalities associated with $\{\overline{P}_{t_1}, \overline{P}_{t_3}, \ldots, \overline{P}_{t_{q-1}}\}$. Thus, inequality (2.20) cannot be facet defining.

(b) If $G[C]$ contains a chord uv, then, J can be decomposed into two terminal cycles J_1 and J_2 with a common edge uv, with q_1 and q_2 terminals, respectively. Suppose, without loss of generality, that J_1 has an odd number of terminals and J_2 has an even one. Also, we may suppose that $v = v_1$ and $v_1, \ldots, v_{q_2}$ are the non-terminal nodes of J_2. Let $\mathcal{P}$ be the set of the $\frac{q_2}{2} - 1$ disjoint terminal paths $\{v_2, v_3\}, \{v_4, v_5\}, \ldots, \{v_{q_2-2}, v_{q_2-1}\}$. The terminal cycle inequality associated with J can then be obtained by summing the terminal cycle inequality related to J_1 and the terminal path inequalities related to the terminal paths of $\mathcal{P}$.

(c) Suppose there exists a vertex $w \in V \setminus C$ not adjacent to a terminal, and adjacent to vertices of $C' \subseteq C$, such that each vertex cover of C of size $\lceil \frac{q}{2} \rceil$ contains at most $|C'| - 2$ vertices of C'. Observe that for every two nodes u, u' of C', (t, u, w, u', t') is a terminal path, where t (t') is the terminal adjacent to u (u'). As a separator of G must contain a vertex cover of C, it follows that any separator of G contains at least $\lceil \frac{q}{2} \rceil + 1$ vertices of $C \cup \{w\}$. Hence, the following inequality

$$x(C) + x(w) \geq \lceil \frac{q}{2} \rceil + 1, \tag{2.21}$$

is valid for $P(G, T)$. However, inequality (2.20) can be obtained by summing the above inequality and $-x(w) \geq -1$, and hence cannot define a facet.

(d) Suppose there exists a vertex $w \in V \setminus C$ adjacent to a terminal t' such that each vertex cover of C of size $\lceil \frac{q}{2} \rceil$ contains at most $|C'| - 1$ vertices of C'. Then, each vertex cover of C of size $\lceil \frac{q}{2} \rceil$, does not intersect at least one terminal path (t, u, w, t'), where $u \in C'$ and t is the terminal adjacent to u. Thus, the inequality (2.21) is valid for $P(G, T)$, but this implies, as before, that inequality (2.20) can be obtained from the above inequality and $-x(w) \geq -1$.

($\Leftarrow$) Suppose that all the Conditions $(a)-(d)$ are satisfied. Denote by $ax \geq \alpha$ inequality (2.20) and let $bx \geq \beta$ be an inequality that defines a facet of $P(G, T)$ such that $\{x \in P(G, T) : ax = \alpha\} \subseteq \{x \in P(G, T) : bx = \beta\}$. Since $P(G, T)$ is full dimensional, we need to prove that there exists ρ such that $b = \rho a$.

Let $S_1 = (V \setminus C) \cup \{v_1, v_3, \ldots, v_q\}$ and $S_2 = (S_1 \setminus \{v_1\}) \cup \{v_2\}$. As C is chordless and q is odd, we have that S_1 and S_2 are separators of G, and their incidence vectors satisfy (2.20) with equality. Hence, $bx^{S_1} = bx^{S_2} = \beta$, which implies that $b(v_1) = b(v_2)$. By symmetry, we obtain that

$$b(u) = b(v) = \rho \qquad \text{for all } u, v \in C, \text{ for some } \rho \in \mathbb{R}.$$

Now let $w \in V \setminus C$. Let $C' = N(w) \cap C$. If w is not adjacent to a terminal, by Condition (c), there exists a vertex cover U of C of size $\lceil \frac{q}{2} \rceil$ containing at least $|C'| - 1$ vertices of C'. Let $\overline{S}_w = ((V \setminus C) \cup U) \setminus \{w\}$. By Condition (b), for all $u_i, v_j \in C \setminus C'$, $u_i v_j \notin E$, thus, the path t_i, u_i, v_j, t_j does not exist. By Condition (c), for all $u, v \in N(w)$, u or v belongs to U or $V \setminus C$, thus, each terminal path containing w, intersects $\overline{S}_w$. Therefore, $\overline{S}_w$ is a separator of G. The incidence vectors of $\overline{S}_w \cup \{w\}$ and $\overline{S}_w$ satisfy inequality (2.20) with equality. Therefore, $bx^{\overline{S}_w \cup \{w\}} = bx^{\overline{S}_w}$ and thus, $b(w) = 0$. If w is adjacent to a terminal, we can show along the same lines that $b(w) = 0$. Therefore,

$$b(v) = 0 \qquad \text{for all } v \in V \setminus C.$$

Consequently, we have that $b = \rho a$ and the proof is complete. ■

2.3.9 Extended terminal cycle inequalities

An *extended terminal cycle* $J = (V(J) \cup T(J), E(J))$ is a general form of a terminal cycle configuration such that each vertex of the cycle C is connected to zero, one or several terminals by means paths, which will be called *branches*. (See Figure 2.11 for illustration). Let $V^1 \subseteq V(J)$ be the set of vertices of degree greater or equal to 3 and $V^2 \subseteq V(J)$ the set of vertices of degree equal to 2. For each vertex $v \in V(J)$, let $d_J(v)$ be the degree of v in J. Let $\beta(J) = \sum_{v \in V^1} (d_J(v) - 3)$.

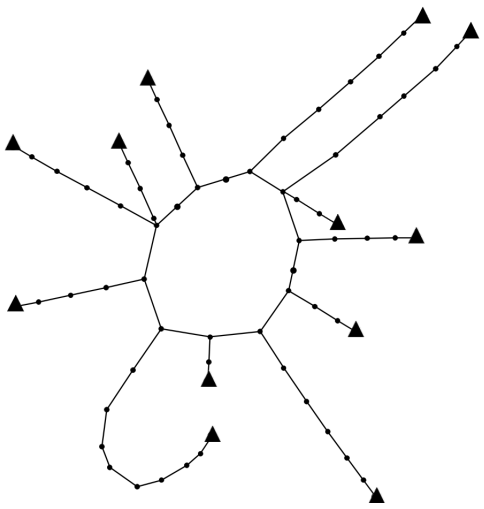


Figure 2.11: Extended terminal cycle with $\beta(J) = 2$.

Then, we have the following theorem

Theorem 2.22 *The inequality*

$$\sum_{v \in V^2} x(v) + \sum_{v \in V^1} (d_J(v) - 2)x(v) \geq \lceil \frac{|V^1|}{2} \rceil + \beta(J) \tag{2.22}$$

is valid for $P(G, T)$.

Proof. The proof is by induction on $\beta(J)$. If $\beta(J) = 0$, then the proof is similar to that of Theorem 2.20. So suppose that $\beta(J) \geq 1$, and that inequality (2.22) is valid for every extended terminal cycle with $\beta(J) \leq s$. We will show that it remains valid when $\beta(J) = s + 1$. So suppose that $\beta(J) = s + 1$. Since $\beta(J) \geq 1$, there must

exist two branches P_t and $P_{t'}$ with a common vertex $u \in C$. Let J^1 (resp. J^2) be the extended terminal cycle obtained from J by removing the vertices of $P_t \setminus \{u\}$ (resp. $P_{t'} \setminus \{u\}$). Remark that $\beta(J^1) = \beta(J^2) = s$. From the induction Hypothesis, the following inequalities are valid for $P(G,T)$.

$$\sum_{v \in V^2 \setminus P_{t'}} x(v) + \sum_{v \in V^1} (d_J(v) - 2)x(v) - x(u) \geq \lceil \frac{|V^1|}{2} \rceil + \beta(J) - 1$$

$$\sum_{v \in V^2 \setminus P_t} x(v) + \sum_{v \in V^1} (d_J(v) - 2)x(v) - x(u) \geq \lceil \frac{|V^1|}{2} \rceil + \beta(J) - 1$$

By summing these inequalities together with $x(P_t \cup P_{t'}) \geq 1$ and $x(u) \geq 0$, dividing the resulting inequality by 2 and rounding up the right hand side, we obtain inequality (2.22). ■

2.4 Reduction operations

In this Section, we are going to describe some graph reduction operations. These can be used in a preprocessing phase in order to reduce the graph and then accelerate the resolution of the problem.

2.4.1 Deletion of a subgraph connected to two terminals

Let H be a subgraph of G such that there exist two terminals $t, t' \in T$ adjacent to $V(H)$, and no vertex in $V(H)$ is adjacent to vertices of $(V \cup T) \setminus (V(H) \cup \{t, t'\})$. See Figure 2.12 for illustration.

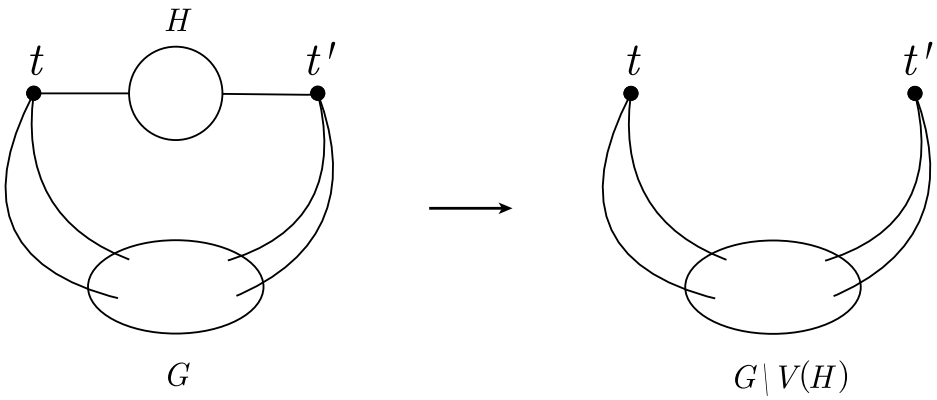


Figure 2.12: Deleting a subgraph connected to two terminals.

Lemma 2.23 *Let S^1 (S^2) be a separator of minimum size in graph $G[V(H) \cup \{t, t'\}]$ ($G[(V \cup T) \setminus V(H)]$). Then, $S^1 \cup S^2$ is a minimum separator in G.*

Proof. Suppose that $S^1 \cup S^2$ is not a separator in G. It follows that there exists a terminal path P between two terminals not intersecting $S^1 \cup S^2$. Then, P is either in $G[V(H) \cup \{t, t'\}]$ or in $G[(V \cup T) \setminus V(H)]$. If P is in $G[V(H) \cup \{t, t'\}]$ ($G[(V \cup T) \setminus V(H)]$), then P does not intersect S^1 (S^2). Thus, S^1 (S^2) is not a separator in $G[V(H) \cup \{t, t'\}]$ ($G[(V \cup T) \setminus V(H)]$), a contradiction. Consequently, $S^1 \cup S^2$ is a separator in G.

Now suppose that $S^1 \cup S^2$ is not minimum. Let $\overline{S} \subset V$ be a minimum separator in G. Let $\overline{S}^1$ and $\overline{S}^2$ be the restriction of $\overline{S}$ in $G[V(H) \cup \{t, t'\}]$ and $G[(V \cup T) \setminus V(H)]$, respectively. Since $\overline{S}$ is a separator in G, it follows that $\overline{S}^1$ ($\overline{S}^2$) intersects all terminal paths in $G[V(H) \cup \{t, t'\}]$ ($G[(V \cup T) \setminus V(H)]$). Since $w(\overline{S}) = w(\overline{S}^1) + w(\overline{S}^2)$, $w(S^1 \cup S^2) = w(S^1) + w(S^2)$ and $w(\overline{S}) < w(S^1 \cup S^2)$, it follows that $w(S^1) < w(\overline{S}^1)$ or $w(S^2) < w(\overline{S}^2)$, a contradiction. ■

As a consequence of Lemma 2.23, graph G can be decomposed into H and $G \setminus H$, and the minimum separator in H can be computed from those of H and $G \setminus H$.

2.4.2 Contraction of a subgraph connected to two vertices

Let H be a subgraph of G such that there exist two vertices $u, v \in V \setminus V(H)$ adjacent to $V(H)$, and no vertex in $V(H)$ is adjacent to vertices of $(V \cup T) \setminus (V(H) \cup \{u, v\})$. Let G' be the graph obtained from G by deleting the vertices of H and adding an edge

between u and v. See Figure 2.13 for illustration. We recall that, from Hypothesis 2.5, each vertex of G has a weight 1.

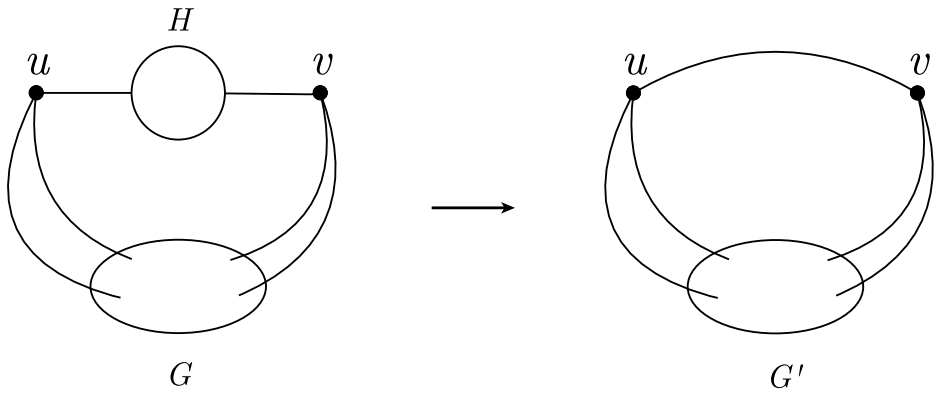


Figure 2.13: Contraction of a subgraph connected to two vertices.

.

Lemma 2.24 *A minimum separator in G' is also a minimum separator in G.*

Proof. Let S be a minimum separator in G'. We first show that S is a separator in G. Suppose that S is not a separator in G. It follows that there exists a terminal path P in G not intersecting S. Since S is a separator in G', it follows that P contains a path $P_v^u \subseteq V(H) \cup \{u, v\}$, between u and v. Let $P' = P \setminus (P_v^u \setminus \{u, v\})$. Thus, S does not intersect the terminal path P' in G', a contradiction.

Now suppose that S is not minimum in G. Let S' be a minimum separator in G. If $S' \cap V(H) = \emptyset$, then it follows that S' is also a separator in G', a contradiction with the fact that S is a minimum separator in G'. If $S' \cap V(H) \neq \emptyset$, let $S'' = (S' \setminus V(H)) \cup \{u\}$. Clearly, S'' is a separator in G and G'. Since $w(S') \leq w(S)$ and $w(S'') = w(S')$, it follows that $w(S'') \leq w(S)$, but this contradicts the fact that S is minimum in G'. ■

2.4.3 Deletion of useless components

Consider two subgraphs H_1 and H_2 of G such that $V = V(H_1) \cup V(H_2)$, $V(H_1) \cap V(H_2) = \{v\}$ and $T(H_2) = \emptyset$. See Figure 2.14 for illustration.

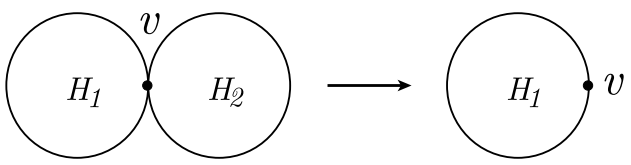


Figure 2.14: Deletion of useless components.

.

Lemma 2.25 *A minimum separator in H_1 is also a minimum separator in G.*

Proof. Remark that there is no terminal path intersecting $V(H_2) \setminus \{v\}$. Thus, no minimal separator intersects $V(H_2) \setminus \{v\}$. ■

It is easily seen that the three operations given by Lemma 2.23, 2.24 and 2.25 can be performed in polynomial time. These operations are used within our Branch-and-Cut algorithm that we present in the next Section.

2.5 Branch-and-Cut Algorithm

In this Section, we describe a Branch-and-Cut algorithm for the MTVSP. Our aim is to address the algorithmic applications of the theoretical results presented in the previous Sections and describe some strategic choices made in order to solve that problem. So, let us assume that we are given a graph $G = (V \cup T, E)$. We assume that all the operations 2.4.1,2.4.2 and 2.4.3 have been performed in a preprocessing phase, and thus, no operation among 2.4.1,2.4.2 and 2.4.3 can be applied for G.

We now describe the framework of our algorithm. To start the optimization, we consider the linear program given by the terminal path inequalities associated with the terminal paths $P^*_{tt'}$, of minimum length (in terms of number of edges), between each pair of terminals $t, t' \in T$ of graph G together with the trivial inequalities, that is

$$\begin{aligned} \min \quad & \sum_{v \in V} x(v) & \\ & x(P^*_{tt'}) \geq 1 \qquad & \forall t, t' \in T, \\ & x(v) \leq 1 \qquad & \forall v \in V, \\ & x(v) \geq 0 \qquad & \forall v \in V. \end{aligned}$$

The optimal solution $x^* \in \mathbb{R}^V$ of this relaxation of the MTVSP is feasible for the problem if x^* is an integer vector that satisfies all the terminal path inequalities. Usually, the solution x^* is not feasible for the MTVSP, and thus, in each iteration of the Branch-and-Cut algorithm, it is necessary to generate further inequalities that are valid for the MTVSP but violated by the current solution x^*. For this, one has to solve the so-called separation problem. This consists, given a class of inequalities, in deciding whether the current solution x^* satisfies all the inequalities of this class, and if not, in finding an inequality that is violated by x^*. An algorithm solving this problem is called a separation algorithm. The Branch-and-Cut algorithm uses the inequalities previously described.

We remark that all inequalities are global (i.e. valid for all the Branch-and-Cut tree) and several inequalities may be added at each iteration. Our strategy is to try to detect violated inequalities at each node of the Branch-and-Cut tree in order to obtain the best possible lower bound and thus, limit the generated nodes.

Now, we describe the separation procedures used in our Branch-and-Cut algorithm. We first show that the separation the terminal path inequalities and the star tree inequalities can be done in polynomial time.

Theorem 2.26 *Inequalities* (2.6) *can be separated in polynomial time.*

Proof. Let $x^* \in [0,1]^V$. For a terminal $t \in T$, add a super terminal t_s to T adjacent to all terminals of $T \setminus \{t\}$. Let $x^*_T \in \mathbb{R}^T$ such that $x^*_T(t_i) = 0$ for all $t_i \in T$. Let $y^* \in [0,1]^E$ be the vector such that $y^*(uv) = \frac{x^*(u)+x^*(v)}{2}$ for all $uv \in E$ and $u, v \in V$, and $y^*(ut_i) = \frac{x^*(u)+x^*_T(t_i)}{2}$ for all $ut_i \in E$, $u \in V$ and $t_i \in T$. Finding a shortest path, between t and any terminal of $T \setminus \{t\}$, in graph G w.r.t y^* is equivalent to finding a shortest path P_t in graph G w.r.t x^* between t and t_s. Consider the path $\widehat{P}$ among the paths P_t for all $t \in T \setminus \{t\}$, of a minimum weight, say $\widehat{z}$. If $\widehat{z} < 1$, then inequality (2.6), associated with $\widehat{P}$, is violated. Otherwise, all inequalities (2.6) are satisfied. ■

Algorithm 3: Separating algorithm for inequalities (2.6)

Data: Graph $G = (V \cup T, E)$ and $x^* \in [0,1]^V$
Result: An inequality (2.6) violated by x^*
begin
  for $(t \in T)$ **do**
    Add a terminal t_s to T adjacent to all terminals of $T \setminus \{t\}$;
    Construct $y^* \in \mathbb{R}^E$ from x^* ;
    Find a shortest path P_{tt_s}, between t and t_s, in G w.r.t y^* ;
    Return inequality (2.6) associated with P_{tt_s} if it is violated by x^*;
    Delete t_s from T;
  end
end

The above algorithm discussed for separating inequalities (2.6) runs in $O(k(n+m)log(n))$-time.

Theorem 2.27 *The star tree inequalities (2.10) can be separated in polynomial time.*

Proof. We will show that the separation problem for the star tree inequalities can be reduced to a minimum cost flow problem. Let $x^* \in [0,1]^V$. For a vertex $v \in V$, let $D_v = (V' \cup T', A')$ be the graph obtained from G as follows

1- $T' = T \cup \{t_s\}$.

2- add a vertex v' in V'.

3- for each vertex $u \in V \setminus \{v\}$, add two vertices u_1 and u_2 in V', and the arc (u_1, u_2) in A'.

4- for each edge $uw \in E$ such that $u \neq v$ and $w \neq v$, add two arcs (u_2, w_1) and (w_2, u_1) in A'.

5- for all $u \in N(v) \setminus T$, add an arc (v', u_1) to A', and for all $t \in N(v) \cap T$, add an arc (v', t) in A'.

6- for all $t \in T$, and $u \in N(t) \setminus \{v\}$, add an arc (u_2, t) to A'.

7- for all $t \in T \setminus t_s$, add an arc (t, t_s) in A'.

Figure 2.15 illustrates the graph transformation. All arcs are given a capacity 1. For all $u \in V \setminus \{v\}$, arc (u_1, u_2) has a weight x^*_u and all other arcs have weight 0.

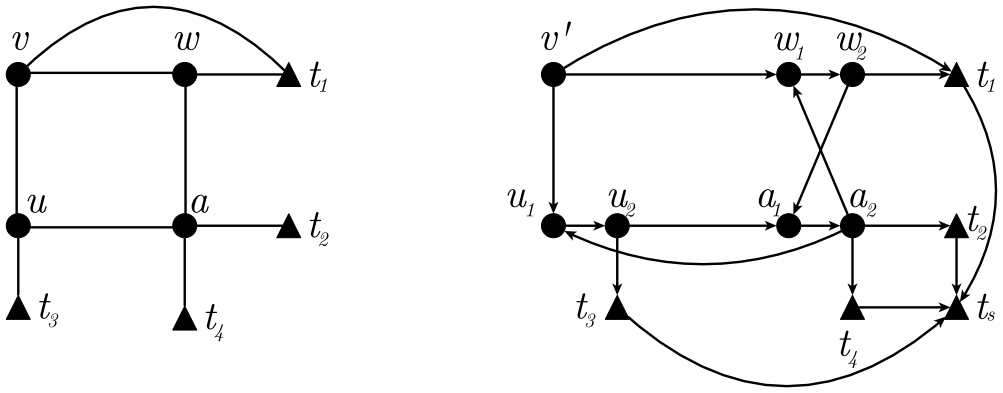


Figure 2.15: Transformation graph from G to D_v

For an integer value $q \in [3, \ldots, |T|]$, we look for a flow of value q of minimum cost, between v' and t_s in D_v. Let A be the set of arcs used by this flow and z' the cost of this flow. We let $W \subseteq V$ be the set of vertices u of V such $(u_1, u_2) \in A$. It can easily seen that $z' + (q-1)x^*_{v'} < q - 1$, then the star tree inequality associated with the star tree given by W is violated. ■

The separation of the star tree inequalities runs in $O(kn^4 mlog(n))$-time.

The polynomial algorithm for separating the star tree inequalities discussed above may be time-consuming. In what follows we devise a heuristic to speed up the separation. The idea of the heuristic is to construct a start tree using the shortest paths between a root node and terminals. The heuristic works as follows. We construct a graph G' obtained from G by adding a super terminal t_s adjacent to all terminals of T. Then, for each integer $q \in [3, \ldots, k]$ and each vertex $v \in V$, we perform the following procedure
We set $S = \{v\}$ and the weight of each vertex $u \in V \setminus \{u\}$ to x^*_u and those of $\{t_s, v\} \cup T$ to 0. Then, we look for the shortest path P^v between t_s and v. If P^v does not exist, then we stop. Otherwise, we update S, V and T as $S = S \cup (P^v \setminus \{t_s\})$, $V = V \setminus (P^v \setminus \{v\})$ and $T = T \setminus (P^v \setminus \{t_s\})$. If the star tree associated with S has q branches, then we

stop. Otherwise, we look for a new shortest path between t_s and v, and so on.
If set S represents a star tree of G of root vertex v and q branches, and if $x^*(S \setminus (T \cup \{v\})) + (q-1)x^*_v < q-1$, then the star tree inequality associated with S is violated. This heuristic is given in Algorithm 4.

Algorithm 4: Separation heuristic for the star tree inequalities.

Data: $G = (V \cup T, E)$, $v \in V$, $q \in \{3, \ldots, k\}$ and a vector $x^* \in [0,1]^V$
Result: Star tree inequality violated by x^*
begin
  $S = \{v\}$;
  for $u \in V$ **do**
    $c_u = x^*_u$;
  end
  for $t \in T \cup \{t_s\}$ **do**
    $c_t = 0$;
  end
  for $i = \{1, \ldots, q\}$ **do**
    $P_i \leftarrow$ the shortest path in $(V \cup T \cup \{t_s\}, E)$ w.r.t c, between v and t_s;
    if $P_i = \emptyset$ **then**
      Stop algorithm.
    end
    else
      $S \leftarrow S \cup (P_i \setminus \{t_s\})$;
      $V \leftarrow V \setminus (P_i \setminus \{v\})$;
      $T \leftarrow T \setminus (P_i \setminus \{t_s\})$;
    end
  end
  If S is a star tree of root v and q branches then check if the associated star tree inequality is violated by x^* ;
end

The heuristic runs in $O(k^2 n(n+m)log(n))$-time.

Now we turn our attention to the separation of the clique star inequalities. This is also performed using a heuristic algorithm. The idea of the heuristic is to generate a set $L(G)$ of cliques in graph G, and for all clique K_q of q vertices, we construct a clique star using the shortest paths between each vertex of K_q and terminals.

The heuristic works as follows. We initialize the list $L(G)$ with all cliques in G of size 3. Then, for a fixed number of iterations $l \in \{1, \ldots, k\}$, for each clique $K \in L(G)$ and for all $v \in N(K)$, we check if $K \cup \{v\}$ and we put it in $L(G)$. Let G' be the graph obtained from G by adding a super terminal t_s adjacent to all terminals of T. During the separation procedure, for each clique $K_q \in L(G)$ of q vertices, we perform the following procedure in G'. We set $S = K_q$ and the weight of each vertex $v \in V$ to x^*_v and those of $\{t_s\} \cup T$ to 0. Then, for a vertex $v \in K_q$, find the shortest path P^v between t_s and v in $G' \setminus (K_q \setminus \{v\})$. If P^v does not exist, then we stop. Otherwise, we update S, V and T as $S = S \cup (P^v \setminus \{t_s\})$, $V = V \setminus (P^v \setminus \{t_s, v\})$ and $T = T \setminus (P^v \setminus \{t_s\})$. If the clique star associated with S has q branches, then we stop. Otherwise, we look for a new shortest path between t_s and another vertex of K_q, and so on. If set S represents a clique star of G of clique K_q and q branches, and if $x^*(S) < q - 1$, then the clique star inequality associated with S is violated. This heuristic is given in Algorithm 5.

Algorithm 5: Separation heuristic for the clique star inequalities.

Data: $G = (V \cup T, E)$, a clique $K_q \in L(G)$ of q vertices and a vector $x^* \in [0,1]^V$
Result: Clique star inequality violated by x^*
begin
  $S = K_q$;
  for $v \in V$ **do**
    $c_v = x^*_v$;
  end
  for $t \in T \cup \{t_s\}$ **do**
    $c_t = 0$;
  end
  for $v \in K_q$ **do**
    $P^v \leftarrow$ the shortest path in graph $((V \setminus K_q) \cup \{v\} \cup T \cup \{t_s\}, E)$ w.r.t c, between v and t_s;
    if $P^v = \emptyset$ **then**
      Stop.
    end
    else
      $S \leftarrow S \cup (P^v \setminus \{t_s\})$;
      $V \leftarrow V \setminus (P^v \setminus \{v\})$;
      $T \leftarrow T \setminus (P^v \setminus \{t_s\})$;
    end
  end
  return S;
  If S is a clique star of clique K_q and q branches then check if the associated clique star inequality is violated by x^* ;
end

The heuristic runs in $O(|L(G)|k(n+m)log(n))$-time.

Now we discuss the separation of the terminal tree inequalities. For this we devised the following heuristic. We look for a spanning tree R of minimum weight in the graph G w.r.t x^* where the terminals have weight 0. Let $\overline{R}$ be the tree obtained from R by deleting the leaf vertices that are not terminals. $\overline{R}$ may have terminals of degree greater than or equal to 2. This means that $\overline{R}$ is not minimal, but the associated terminal tree inequality remains valid for $P(G,T)$. Then, we check whether the corresponding terminal tree inequality is violated. For determining the minimum spanning tree we use

Kruskal's algorithm. Thus, our algorithm can be implemented in $O(mlog(m))$-time. The graph in Figure 2.16, illustrates the deletion of all non-terminal leaves.

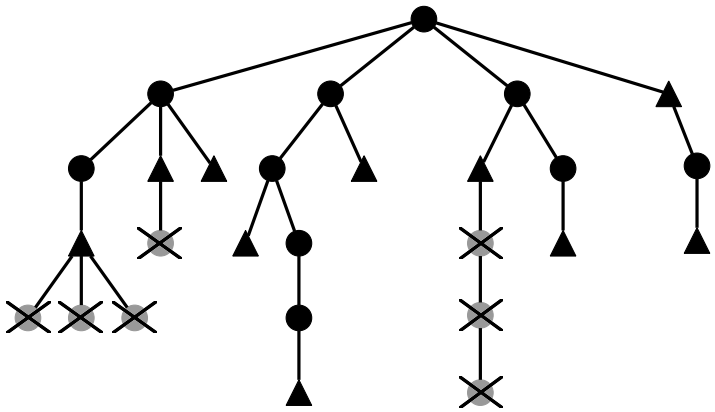


Figure 2.16: Deletion of non-terminal leaves.

These steps are detailed in Algorithm 6.

Algorithm 6: Separation heuristic for the terminal tree inequalities.

Data: $G = (V \cup T, E)$, $x^* \in [0,1]^V$
Result: Terminal tree inequality violated by x^*
begin
  $S \leftarrow$ spanning tree of minimum weight in G w.r.t x^*;
  while *S has a non-terminal vertex of degree* 1 **do**
    $v \leftarrow$ a non-terminal vertex of degree 1 in S;
    $S \leftarrow S \setminus \{v\}$;
  end
  return S;
  Check if the terminal tree inequality induced by S is violated by x^*;
end

We also devised a heuristic for separating the extended terminal cycle inequalities. The idea of the heuristic is to generate a cycle, and construct an extended terminal cycle using the shortest paths between each vertex of the cycle and terminals. The heuristic works as follows. First, we look for a cycle C of minimum weight in the graph $G[V]$ w.r.t x^*. This can be done in a polynomial time. We then construct a graph G' from G by adding a super terminal t_s adjacent to all terminals of T. We set $S = C$, the

weight of each vertex $v \in V$ to x^*_v and those of $\{t_s\} \cup T$ to 0. For each vertex $v \in C$, we look for the shortest path P^v between t_s and v in the graph $G' \setminus (C \setminus \{v\})$. If P^v does not exist, then we stop. Otherwise, we update S, V and T as $S = S \cup (P^v \setminus \{t_s\})$, $V = V \setminus (P^v \setminus \{t_s, v\})$ and $T = T \setminus (P^v \setminus \{t_s\})$. If set S represents an extended terminal cycle of G of q branches, and if $x^*(S \setminus T) < \lceil \frac{q}{2} \rceil$, then the extended terminal cycle inequality associated with the extended terminal cycle induced by S, is violated. This procedure is given in Algorithm 7.

Algorithm 7: Separation heuristic for the extended terminal cycle inequalities.

Data: $G = (V \cup T, E)$, and a vector $x^* \in [0,1]^V$
Result: Extended terminal cycle inequality violated by x^*
begin
  $C \leftarrow$ Cycle of minimum weight in the graph $G[V]$ w.r.t x^*;
  $S \leftarrow C$;
  for $v \in V$ **do**
    $c_v = x^*_v$;
  end
  for $t \in T \cup \{t_s\}$ **do**
    $c_t = 0$;
  end
  for $v \in C$ **do**
    $P^v \leftarrow$ the shortest path in $G' \setminus (C \setminus \{v\})$ w.r.t c, between v and t_s;
    if $P^v = \emptyset$ **then**
      Stop algorithm.
    end
    else
      $S \leftarrow S \cup (P^v \setminus \{t_s\})$;
      $V \leftarrow V \setminus (P^v \setminus \{v\})$;
      $T \leftarrow T \setminus (P^v \setminus \{t_s\})$;
    end
  end
  return S;
  If S is the extended terminal cycle of cycle C and has $|C|$ branches then
   check if the associated extended terminal cycle inequality is violated by x^* ;
end

Thus, our algorithm can be implemented in $O(kn^2(m+n)log(m))$-time.

2.5.1 Heuristics and performance guarantee

In [25], the authors show that the multi-terminal vertex separator problem can be solved in polynomial time when the graph has two terminals. In what follows, we will use this result to develop two different heuristics with performance guarantee for the problem.

2.5.1.1 Disconnecting terminal pairs heuristic

The idea of the heuristic is to construct a separator of G from the union of minimum separators of G disconnecting each pair of terminals in T. The heuristic permits to compute for each pair of terminals $t, t' \in T$, a separator $S^t_{t'}$ of minimum weight $\omega(S^t_{t'})$, intersecting all terminal paths between t and t', and then to return $S = \bigcup_{t,t' \in T} S^t_{t'}$. This heuristic is given in Algorithm 8.

Algorithm 8: Disconnecting terminal pairs heuristic.

```
Data: Graph G = (V ∪ T, E)
Result: Vertex separator
begin
    S = ∅;
    for t, t' ∈ T do
        S^t_{t'} ← The minimum vertex separator of (V ∪ {t, t'}, E);
        S ← S ∪ S^t_{t'};
    end
    return S;
end
```

Theorem 2.28 *The disconnecting terminal pairs heuristic constructs a multi-terminal vertex separator whose weight ω^* is no more than $\frac{k(k-1)}{2}$ times the optimal weight.*

Proof. Let S be the solution given by the disconnecting terminal pairs heuristic of weight ω^*. Let $\overline{S}$ be a separator of minimum weight $\overline{\omega}$ in G. For two different terminals $t, t' \in T$, let $\overline{S}^t_{t'} \subseteq \overline{S}$ be a set of vertices having a minimum weight $\overline{\omega}^t_{t'}$, intersecting all terminal paths between t and t'. Clearly, $\omega(S^t_{t'}) \leq \overline{\omega}^t_{t'}$. Moreover, each vertex in $\overline{S}$ can belong to a separator $\overline{S}^t_{t'}$ for some pair of terminals $t, t' \in T$. So, each vertex

of $\overline{S}$ belongs to at most $\binom{k}{2}$ different separators $\overline{S}^t_{t'}$ disconnecting pairs of terminals. Therefore,

$$\omega^* \leq \sum_{t,t' \in T} \omega(S^t_{t'}) \leq \sum_{t,t' \in T} \overline{\omega}^t_{t'} \leq \binom{k}{2} \overline{\omega}$$

And the result follows. ■

Corollary 2.29 *When $k = 3$, the disconnecting terminal pairs heuristic is a 3-approximation algorithm for the MTVSP.*

Remark 2.30 *If the graph is a star tree, the disconnecting terminal pairs heuristic gives a solution of weight equal to $\frac{k(k-1)}{2}$ times the optimal weight.*

Thus, the bound given above is tight.

2.5.1.2 Isolating terminal heuristic

In [46], the authors propose a heuristic for the multi-terminal cut problem (the edge version of the MTVSP). They show that the solution given by the heuristic is of a weight guaranteed to be no more than $\frac{2(k-1)}{k}$ times the optimal weight. This heuristic can be adapted for the MTVSP. For a terminal $t \in T$, an *isolating terminal* $S^t \subseteq V$ is a set of vertices intersecting all terminal paths between t and all terminals of $T \setminus \{t\}$. The idea of the heuristic is to construct a separator of G, from the union of all minimum isolating terminal sets of G. It works as follows. For each terminal $t \in T$, we compute the separator S^t of minimum weight $\omega(S^t)$, intersecting all terminal paths between t and all terminals of $T \setminus \{t\}$. Let S^{t^*} be the separator, computed before, of maximum

weight. Then, we return $S = \bigcup_{t \in T \setminus \{t^*\}} S^t$. This heuristic is given in Algorithm 9.

Algorithm 9: Isolating terminal heuristic.

```
Data: Graph G = (V ∪ T, E)
Result: Vertex separator
begin
    S = ∅;
    for t ∈ T do
        Merge all terminals of T \ {t} into a new terminal t_s;
        S^t ← The minimum vertex separator of (V ∪ {t, t_s}, E);
    end
    t* ← argmax_{t∈T} {ω(S^t)};
    S = ⋃_{t∈T\{t*}} S^t;
    return S \ S^{t*} ;
end
```

Set S is a separator in G since for all $t \in T$, S contains a subset of vertices in V intersecting all terminal paths between t and all terminals of $T \setminus \{t\}$.

Theorem 2.31 *The isolating terminal heuristic constructs a multi-terminal vertex separator whose weight ω^* is guaranteed to be no more than $(k-1)$ times the optimal weight.*

Proof. Let S be the separator given by the isolating terminal heuristic and $S^t \subseteq S$ be the set of vertices of minimum weight $\omega(S^t)$, intersecting all terminal paths between t and all terminals of $T \setminus \{t\}$. Let $\overline{S}$ be a separator of minimum weight $\overline{\omega}$ in G. For a terminal $t \in T$, let $\overline{S}^t \subseteq \overline{S}$ be a set of vertices having a minimum weight $\overline{\omega}^t$, intersecting all terminal paths between t and all terminals in $T \setminus \{t\}$. Clearly, $\omega(S^t) \leq \overline{\omega}^t$. Moreover, each vertex of $\overline{S}$ can belong to each separator $\overline{S}^t$ for each terminal $t \in T$. Thus, $\sum_{t \in T} \overline{\omega}^t \leq k\overline{\omega}$. Therefore,

$$\omega^* \leq (k-1)\frac{\sum_{t \in T} \omega(S^t)}{k} \leq \frac{k-1}{k} \sum_{t \in T} \overline{\omega}^t \leq k\frac{k-1}{k}\overline{\omega} \leq (k-1)\overline{\omega}$$

Thus, our theorem holds. ■

Corollary 2.32 *When $k = 3$, the isolating terminal heuristic is a 2-approximation algorithm for the MTVSP.*

Remark 2.33 *The star tree is an example on which the isolating terminal heuristic can give a solution of weight equal to $(k-1)$ times the optimal weight.*

Thus, the bound given above is tight.

2.5.1.3 Improved isolating terminal heuristic

This heuristic consists in looking for a separator S by applying the isolating terminal heuristic, and try to pull off one by one each vertex from S, and check if the remaining set is a separator. If not, we put it back in S. It would be more interesting to start by removing the vertices of small degrees. This heuristic is given in Algorithm 10.

Algorithm 10: Improved isolating terminal heuristic.

Data: Graph $G = (V \cup T, E)$
Result: Vertex separator $S \subseteq V$
begin
  $S = \emptyset$;
  for $t, t' \in T$ **do**
    $S^t_{t'} \leftarrow$ The minimum vertex separator of $(V \cup \{t, t'\}, E)$;
    $S \leftarrow S \cup S^t_{t'}$;
  end
  Sort $S = \{v_0, v_1, \ldots, v_{|S|}\}$ from the vertex of the smallest degree to the vertex of the largest degree;
  for $(i \in \{0, \ldots, |S|\})$ **do**
    if $S \setminus \{v_i\}$ *is a separator* **then**
      $S \leftarrow S \setminus \{v_i\}$;
    end
  end
  return S;
end

2.6 Computational Results

The Branch-and-Cut algorithm described in the previous section has been implemented in C++, using Cplex to manage the Branch-and-Cut tree and also as a lp-solver, and all flow problems are solved using Lemon library [2]. It was tested on an Intel Xeon E312xx machine at 2.39 GHz ×1 with 48GB RAM, running under Linux 64 bits. The maximum CPU run time has been fixed to 4 hours. We use two kinds of instances, the *DIMACS* graph coloring instances [1] and random graphs, generated using boost graph library [3] in C++. The terminals are new nodes added to the graphs. Each terminal is randomly connected to at least 2 vertices and at most a given $deg\_T \in \mathbb{N}$ vertices of V. The edges incident to the terminals are added respecting the Hypotheses 2.3-2.4, given in Chapter 2. In the Tables below, the maximum degree of the terminals is fixed in relation with the size of the graph, *i.e.,* the higher the size of graph is, the higher the maximum degree of terminals is. In all our experiments, we have used the reduction operations described in the previous section to reduce the graph. As, the separation algorithms generate less than 10 inequalities per family of inequalities and per iteration, then we perform all of them for each separation phase. Moreover, we tested different separation orders, and we noticed that separating all inequalities, at each separation phase, gives the best results. Each instance is given by its name followed by an extension representing the number of nodes of the graph.

In the following Tables, we have the following entries

n	:	the number of vertices in V.
m	:	the number of edges in E.
k	:	the number of terminal in T.
d %	:	the density of graphs without terminals.
Tp	:	the number of terminal path inequalities separated.
St	:	the number of star tree inequalities separated.
Cs	:	the number of clique star inequalities separated.
Tt	:	the number of terminal tree inequalities separated.
Etc	:	the number of extended terminal cycle inequalities separated.
Nodes	:	the number of node in the branching tree.
Gap %	:	the relative error between the best upper and lower bound obtained at the root node of the branching tree.
CPU_1	:	the CPU time given by the Branch-and-Cut algorithm in second, on the natural formulation.
CPU_2	:	the CPU time given by Cplex in second, on the double indices formulation.
deg_T	:	the maximum degree of each terminal.
Heur_val	:	the objective value of the solution given by the improved isolating terminal heuristic.
Opt_val	:	the optimal objective value.
CPU_Heur	:	the CPU time of the Heuristic, in second.
Heur_Gap %	:	the value of $\frac{\text{Heur\_val}-\text{Opt\_val}}{\text{Opt\_val}}$.

In the first Tables 2.2 and 2.3, we compare the Branch-and-Cut algorithm with Cplex for the compact double indices formulation.

Instances	n	m	k	Tp	St	Cs	Tt	Etc	Nodes	Gap %	CPU_1	CPU_2
Myciel5	47	258	6	48	189	0	28	0	7	35.20	7.96	0.50
Queen8_8	64	1477	6	30	22	21	2	9	1	0.00	0.99	0.32
Huck	74	624	6	64	51	3	5	1	23	19.20	3.39	1.51
Jean	80	533	6	69	100	7	1	0	47	28.10	6.94	1.66
David	87	835	6	31	147	25	26	3	26	28.50	8.58	4.24
Myciel6	95	778	6	35	38	0	26	15	1	0.00	1.26	0.31
Queen8_12	96	2762	6	33	36	35	1	18	1	0.00	1.45	1.41
Queen10_10	100	2967	8	56	31	32	4	6	1	0.00	**1.55**	2.38
Games120	120	1307	8	64	377	138	8	18	9	35.40	**16.50**	23.95
DSJC125	125	764	8	105	70	12	64	12	1	0.00	**2.33**	3.87
Miles250	128	804	8	103	127	16	4	2	29	28.10	5.75	0.42
Miles500	128	2370	8	97	386	62	9	9	6	31.80	16.44	2.41
Miles750	128	4256	8	63	60	49	13	15	1	0.00	3.62	2.87
Miles1000	128	6462	8	53	30	24	10	12	1	0.00	**2.03**	6.38
Anna	138	1022	8	92	344	55	0	1	74	16.60	17.99	0.72
Queen12_12	144	5224	8	85	20	18	9	9	1	0.00	**1.44**	8.18
Mulsol.i.2	188	3920	8	116	63	34	2	13	1	0.00	**2.69**	6.32
Myciel7	191	2387	8	58	27	0	13	13	1	0.00	**1.34**	17.09
Queen14_14	196	8399	8	56	19	15	12	11	1	0.00	**1.24**	22.52
Mulsol.i.1	197	3952	8	71	11	5	10	3	1	0.00	**0.58**	5.61
Zeroin.i.3	206	3576	8	28	47	48	1	2	1	0.00	**2.03**	5.49
Queen16_16	256	12674	8	88	27	19	5	11	1	0.00	**2.10**	41.55
School1	385	19129	8	109	16	15	12	5	1	0.00	**1.08**	103.00
DSJR500	500	7140	8	29	374	331	3	141	4	40.00	32.35	19.80

Table 2.2: Results from DIMACS instances.

Table 2.2 presents the results obtained for DIMACS instances associated to graphs having up to 500 nodes. The number of terminals is fixed to 6 for graphs with less than 99 nodes and 8 for the others. As we can remark in the table, all instances were solved by the Branch-and-Cut algorithm in less than 33 seconds. The bold values in the CPU_1 column represent the instances for which the Branch-and-Cut algorithm solves the instance faster than the Cplex with the double indices formulation. We notice that in 50% of the instances the Branch-and-Cut algorithm beats the Cplex. The valid inequalities (2.4),(2.7),(2.18) and (2.20) have been efficient for strengthening the formulation. Indeed 62% of the instances were solved by the Branch-and-Cut algorithm in the first node of the branching tree. We also notice that all the valid inequalities appear in the instances Queen8_8, Queen8_12, Queen10_10, Queen12_12, Queen14_14, Queen15_15, since underlying graphs are without holes. However, no clique star inequality appears for instances Myciel5, Myciel6 and Myciel7 since they are induced by triangle free graphs. Moreover, we can remark that the Branch-and-

Cut algorithm generates a few number of valid inequalities. Only 4 instances over 24 were solved by the Branch-and-Cut algorithm in more than 10 seconds. For these four instances, the number of star tree inequalities is greater than 340, in contrast with the other instances, where the number of star tree inequalities is less than 200.

Instances	n	m	k	Tp	St	Cs	Tt	Etc	Nodes	Gap %	CPU_1	CPU_2
R_50	50	511	7	48	33	22	31	6	1	0.00	1.24	0.44
R_70	70	993	7	44	18	14	19	4	1	0.00	**0.64**	0.78
R_100	100	1985	7	45	13	13	12	5	1	0.00	**1.02**	3.07
R_300	300	17792	7	67	8	7	8	2	1	0.00	**2.31**	55.67
R_600	600	70673	7	23	10	9	9	5	1	0.00	**4.04**	298.02
R_700	700	96432	7	48	7	6	5	1	1	0.00	**3.96**	360.79
R_800	800	125978	10	95	26	23	0	13	1	0.00	**20.98**	1323.90
R_900	900	159331	10	96	39	36	1	15	1	0.00	**31.31**	1959.60
R_1000	1000	196771	10	86	35	34	0	16	1	0.00	**34.35**	2395.20
R_1200	1200	283386	10	92	26	24	0	9	1	0.00	**41.01**	4595.10
R_1300	1300	332861	10	47	30	30	0	15	1	0.00	**56.91**	5142.00
R_1500	1500	442444	10	99	33	32	0	16	1	0.00	**92.64**	11735.00
R_1800	1800	637307	15	110	44	40	40	17	1	0.00	**379.95**	-
R_2000	2000	786639	15	108	37	32	35	16	1	0.00	**468.78**	-
R_2100	2100	867430	15	214	34	31	28	15	1	0.00	**641.02**	-
R_2300	2300	1041158	15	217	29	28	25	16	1	0.00	**851.49**	-
R_3000	3000	1770773	15	215	33	30	19	12	1	0.00	**1698.90**	-
R_3300	3300	2142370	15	112	38	37	11	18	1	0.00	**2129.60**	-
R_3800	3800	2841805	15	217	28	27	26	13	1	0.00	**2363.80**	-
R_4200	4200	3472117	15	222	37	35	30	22	1	0.00	**3856.30**	-
R_4500	4500	3987339	15	217	41	40	31	15	1	0.00	**5248.10**	-
R_4800	4800	4537794	15	218	31	22	31	10	1	0.00	**4611.80**	-
R_5000	5000	4922710	15	110	27	26	27	16	1	0.00	**4291.40**	-

Table 2.3: Results from random instances.

Table 2.3 presents the results obtained for random instances on graphs having up to 5000 nodes. The maximum degree of terminals is fixed to 7 for instances with less than 700 nodes, 10 for instances with the number of nodes between 800 and 1500 and to 15 for instances with 1800 nodes and more. As it appears, in 95.6% of instances, the Branch-and-Cut algorithm beats Cplex. Indeed, Cplex was faster than the Branch-and-Cut in only 1 instance (R_1) with a difference of 0.8 seconds. In the other instances, the slack between the CPU_times increases exponentially. For instance, for R_100 the difference is 2.05 seconds and for R_1300 it is 5085.09 seconds. Also note that all instances were solved in the root of the branching tree by the Branch-and-Cut algorithm. Indeed, for random instances our valid inequalities are enough for finding

the optimal solution without branching. Cplex with the Double indices formulation could not solve the instances with 1800 vertices and more. This can be explained by the out of memory due to the number of constraints in the double indices formulation when the number of edges is high. Also random graphs have no specific structure and this let the instance harder to solve. For example, for instance R_1800, the formulation has more than 9.5 millions of constraints. For both DIMACS and random graphs, the number of star tree inequalities, clique star inequalities and terminal tree inequalities is roughly the same.

In Tables 2.4, 2.5 and 2.6, we vary the density of the graph, the number of terminals and the maximum degree of the terminals.

Instances	n	m	k	d %	Nodes	Gap %	CPU_1
R_1000	1000	90682	10	18	1	0.00	15.18
		196771		39	1	0.00	34.35
		275346		55	1	0.00	84.88
R_2000	2000	362335	15	18	1	0.00	172.00
		786639		39	1	0.00	468.70
		1100921		55	1	0.00	1183.00
R_3000	3000	815760	15	18	1	0.00	622.30
		1770773		39	1	0.00	1698.00
		2478823		55	1	0.00	1650.00
R_4000	4000	1450789	15	18	1	0.00	1010.15
		3150332		39	1	0.00	2502.08
		4410131		55	1	0.00	3198.85
R_5000	5000	2267316	15	18	1	0.00	1904.00
		4922710		39	1	0.00	4291.00
		6890000		55	1	0.00	5802.00

Table 2.4: Results from random instances with different density.

Tables 2.4 represents the results for random graphs, with different densities, 18%, 39% and 55%. We notice that in almost all the instances, the higher the density is, the higher CPU time is.

Instances	n	m	k	Nodes	Gap %	CPU_1
R_1000	1000	196756	7	1	0.00	14.35
		196771	10	1	0.00	34.35
		196787	13	1	0.00	103.00
R_2000	2000	786625	12	1	0.00	499.90
		786639	15	1	0.00	468.70
		786652	18	1	0.00	990.70
R_3000	3000	1770758	12	1	0.00	1222.21
		1770773	15	1	0.00	1698.00
		1770787	18	1	0.00	2472.00
R_4000	4000	3150315	12	1	0.00	2274.11
		3150332	15	1	0.00	2502.08
		3150347	18	1	0.00	4360.06
R_5000	5000	4922700	12	1	0.00	3367.00
		4922710	15	1	0.00	4291.00
		4922728	18	1	0.00	7919.00

Table 2.5: Results from random instances with different number of terminals.

Tables 2.5 represents the results for random graphs with different number of terminals. In almost all the instances, the higher number of terminals is, the higher the CPU time is. All instances are solved in the root node of the branching tree.

Instances	n	m	k	deg_T	Nodes	Gap %	CPU_1
R_1000	1000	196751	10	5	1	0.00	26.93
		196771		10	1	0.00	34.35
		196769		15	1	0.00	43.07
R_2000	2000	786617	15	5	1	0.00	408.50
		786639		10	1	0.00	468.70
		786638		15	1	0.00	715.70
R_3000	3000	1770741	15	5	1	0	1446.00
		1770773		10	1	0.00	1698.00
		1770766		15	1	0.00	2650.00
R_4000	4015	3150301	15	5	1	0.00	1676.69
		3150332		10	1	0.00	2502.08
	4015	3150322		15	1	0.00	3007.54
R_5000	5000	4922695	15	5	1	0.00	4015.00
		4922710		10	1	0.00	4291.00
		4922714		15	1	0.00	5027.00

Table 2.6: Results from random instances with different degree of terminals.

Tables 2.6 represents the results for random graphs with different values of deg_T (We recall that the terminals are randomly connected to at least 2 vertices and at most deg_T vertices of V). We notice that the higher deg_T is, the higher CPU time is. Also, the Branch-and-Cut algorithm could solve all instances in the root node of the branching tree.

Instances	n	m	k	Tp	Nodes	Gap %	CPU_1
Myciel5	47	258	6	161	44	35.2	0.06
Queen8_8	64	1477	6	129	30	15.6	0.04
Huck	74	624	6	86	16	19.2	0.01
Jean	80	533	6	139	43	28.1	0.03
David	87	835	6	154	45	28.5	0.04
Myciel6	95	778	6	206	80	36.1	0.08
Queen8_12	96	2762	6	176	56	38.0	0.08
Queen10_10	100	2967	8	235	67	38.6	0.11
Games120	120	1307	8	304	338	35.4	0.57
DSJC125	125	764	8	311	204	39.1	0.31
Miles250	128	804	8	148	57	28.1	0.05
Miles500	128	2370	8	365	124	31.8	0.22
Miles750	128	4256	8	372	148	37.5	0.30
Miles1000	128	6462	8	327	86	37.5	0.21
Anna	138	1022	8	354	91	16.6	0.13
Queen12_12	144	5224	8	282	90	36.0	0.18
Mulsol.i.2	188	3920	8	341	164	37.5	0.26
Myciel7	191	2387	8	284	121	38.6	0.21
Queen14_14	196	8399	8	279	89	41.3	0.24
Mulsol.i.1	197	3952	8	183	56	38.8	0.11
Zeroin.i.3	206	3576	8	257	65	35.7	0.12
Queen16_16	256	12674	8	364	142	39.2	0.42
School1	385	19129	8	359	198	38.8	0.53
DSJR500	500	7140	8	697	2735	40.0	26.52

Table 2.7: Results from DIMACS instances without the valid inequalities.

Instances	n	m	k	Tp	Nodes	Gap %	CPU_1
R_50	50	511	7	179	48	38.8	0.05
R_70	70	993	7	178	42	39.4	0.05
R_100	100	1985	7	172	47	38.8	0.06
R_300	300	17792	7	163	53	39.4	0.43
R_600	600	70673	7	150	53	39.4	1.16
R_700	700	96432	7	150	35	38.8	1.31
R_800	800	125978	10	677	406	42.0	10.55
R_900	900	159331	10	821	813	43.1	23.00
R_1000	1000	196771	10	604	463	40.0	17.36
R_1200	1200	283386	10	670	461	42.0	35.34
R_1300	1300	332861	10	646	374	42.0	54.95
R_1500	1500	442444	10	689	657	42.5	90.38
R_1800	1800	637307	15	1922	3168	44.8	716.00
R_2000	2000	786639	15	1326	2504	42.6	653.90
R_2100	2100	867430	15	2172	3716	45.1	1441.00
R_2300	2300	1041158	15	1593	2190	44.2	1390.00
R_3000	3000	1770773	15	1583	3856	43.3	3570.00
R_3300	3300	2142370	15	2055	3498	45.5	5751.00
R_3800	3800	2841805	15	1651	2444	44.4	6035.00
R_4200	4200	3472117	15	1969	3494	45.0	9521.00
R_4500	4500	3987339	15	2049	3753	45.0	1230.000
R_4800	4800	4537794	15	1743	2525	44.5	9282.00
R_5000	5000	4922710	15	1451	1839	44.7	10037.00

Table 2.8: Results from random instances without the valid inequalities.

Tables 2.7 and 2.8 represent the results for random graphs using the Branch-and-Cut algorithm without the additional valid inequalities, *i.e.,* with only the terminal path inequalities. Observe that when the number of vertices is less than 1800, the CPU time is shorter than the one in Tables 2.2 and 2.3. This is explained by the good performance of Cplex. However, the number of nodes in the branching tree and the gap are higher. Indeed, the linear relaxation of the Natural formulation is week. For the instances with at least 1800 vertices, we notice that the CPU time, gap and number of nodes in the branching tree in Table 2.8 are higher than the CPU time in Table 2.3. Our inequalities have a great impact for large instances. The size of the branching tree has also been reduced.

Instances	n	m	k	Heur_val	Opt_val	CPU_Heur	Heur_Gap %
Myciel5	47	258	6	17	17	0.00	0.00
Queen8_8	64	1477	6	16	16	0.01	0.00
Huck	74	624	6	14	13	0.00	7.14
Jean	80	533	6	16	16	0.00	0.00
David	87	835	6	14	14	0.00	0.00
Myciel6	95	778	6	18	18	0.00	0.00
Queen8_12	96	2762	6	21	21	0.02	0.00
Queen10_10	100	2967	8	22	22	0.03	0.00
Games120	120	1307	8	24	24	0.01	0.00
DSJC125	125	764	8	23	23	0.01	0.00
Miles250	128	804	8	16	16	0.01	0.00
Miles500	128	2370	8	24	22	0.03	8.33
Miles750	128	4256	8	24	24	0.05	0.00
Miles1000	128	6462	8	24	24	0.07	0.00
Anna	138	1022	8	26	21	0.02	19.20
Queen12_12	144	5224	8	25	25	0.06	0.00
Mulsol.i.2	188	3920	8	28	28	0.04	0.00
Myciel7	191	2387	8	22	22	0.03	0.00
Queen14_14	196	8399	8	23	23	0.10	0.00
Mulsol.i.1	197	3952	8	18	18	0.04	0.00
Zeroin.i.3	206	3576	8	21	21	0.04	0.00
Queen16_16	256	12674	8	28	28	0.15	0.00
School1	385	19129	8	27	27	0.20	0.00
DSJR500	500	7140	8	25	25	0.10	0.00

Table 2.9: Results given by the improved isolated terminal heuristic.

Table 2.9 presents the results given by the improved isolated terminal heuristic for DIMACS instances. We can remark that the heuristic gives very good results in short time. In 93.6% of the instances, the solution given by the heuristic is optimal. For the other instances, the gap is very small. On the other hand, when the number of terminals is high, the CPU time of the heuristic becomes high. All instances were solved in 0.2 seconds or less.

2.7 Conclusion

In this chapter we have considered the multi-terminal vertex separator problem. We have first shown that the problem is NP-hard. Then, we have proposed two integer programming formulations for the problem. For one of these formulations we have identified some valid inequalities and discussed their facial structure. Using this, we have developed a Branch-and-Cut algorithm for the problem and presented extensive computational results. These show the effectiveness of the valid inequalities used in the algorithm. The computational results have also permitted to measure the importance of our separation heuristics. In particular the heuristic for separating star tree inequalities appeared more efficient than the exact polynomial algorithm. We have also proposed two heuristics for the problem, and analyzed their performance guarantee. The improved isolating terminal heuristic has appeared to be very efficient since it could find the optimal solution for most of the instances.

Chapter 3

The multi-terminal vertex separator problem : Composition of Polyhedra

Contents

In this chapter we are interested in characterizing the multi-terminal vertex separator polytope in graphs that are decomposable by one-node cutsets and terminal cutsets. When G decomposes into G_1 and G_2, we derive a procedure for composing facets and algorithms for the MTVSP. We also introduce the minimum rooted cycle cover problem

for which the MTVSP is a special case. We use Menger's theorem to provide a characterization of all rooted graphs in which the maximum number of vertex-disjoint rooted cycles equals the minimum size of a subset of non-root vertices intersecting all rooted cycles, for all subgraphs.

This chapter is organized as follows. In Section 1, we give two linear systems describing the multi-terminal vertex separator polytope for the star trees and the clique stars. Then, we show that these two linear systems are totally dual integral. In Section 2, we study a composition (decomposition) technique of the multi-terminal vertex separator polytope, for graphs that are decomposable by one-node cutsets. Then, we derive a composition procedure for facets and algorithms. In Section 3, we present a further composition procedure for the multi-terminal vertex separator polytope, by merging terminals. Section 4 is devoted to the more general minimum rooted cycle cover problem.

Preliminaries

In this chapter we suppose that Hypotheses 2.1-2.5, given in Chapter 2, are satisfied. Consider a graph $G = (V \cup T, E)$. Let $P(G,T) = conv(x \in [0,1]^V |\quad x \text{ satisfies (2.6)})$ be the polytope given by inequalities (2.6) − (2.8).

Menger's theorem [93] : *Let G be a graph and let s and t be two nonadjacent vertices of G. The minimum size of a st-vertex cut is equal to the maximum number of internally vertex-disjoint st-paths.*

Lemma 3.1 *When $|T| = 2$, the linear system (2.6)-(2.8) is totally dual integral.*

Proof. Given a graph $G = (V \cup T, E)$. From Hypothesis 2.5, for any weight vector $\omega : V \to \mathbb{Z}$, we can transform graph G in order to obtain all weights equal to 1. We notice that the dual D, of the linear program (2.5)-(2.8), consists in finding the maximum number of internally vertex disjoints terminal paths between two terminals in G. From Menger's theorem, we have that the value of the optimal integer solution of D is equal to the value of the optimal integer solution of (2.5)-(2.8). It follows that D has an integer solution for any weight vector ω. Consequently, the linear system (2.6)-(2.8) is totally dual integral. ■

3.1 Star trees and Clique stars

In this section we focus on two classes of graphs, the star trees and the clique stars, presented in Chapter 2. We give two linear systems describing the associated multi-terminal vertex separator polytopes and we show that these systems are totally dual integral.

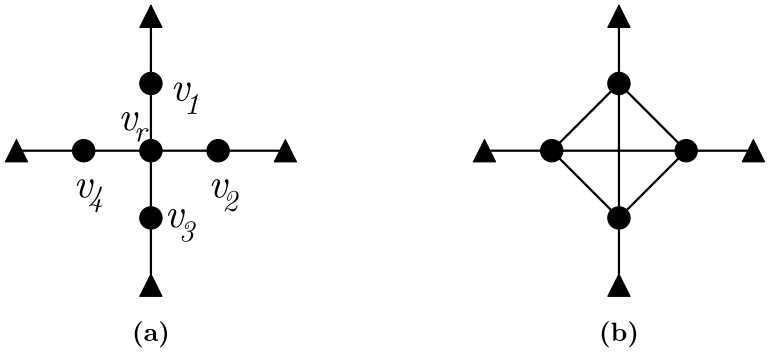


Figure 3.1: Star tree and clique star

Figure 3.1.(a) displays a star tree $H = (V(H) \cup T(H), E(H))$ with 4 branches and Figure 3.1.(b) shows a clique star $Q = (V(Q) \cup T(Q), E(Q))$ with 4 branches.

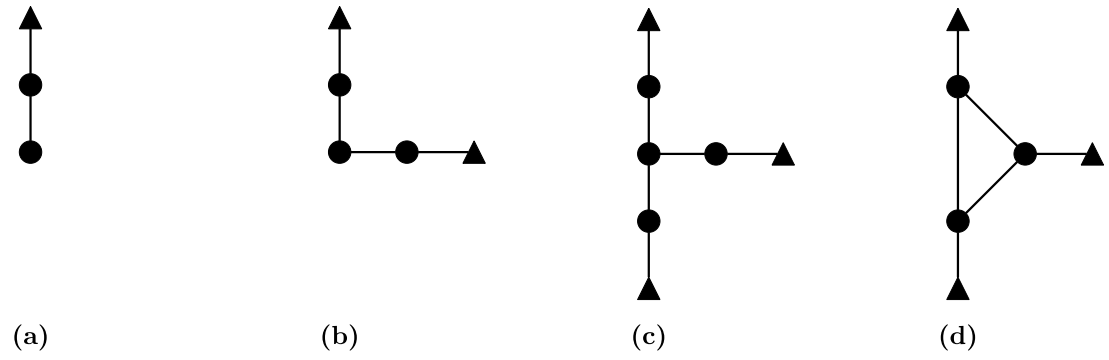


Figure 3.2: Star trees and clique stars

Consider a graph $G = (V \cup T, E)$ that is a star tree (resp. clique star) of k terminals. If $k = 1$, G is reduced to a single branch, see Figure 3.2.(a). If $k = 2$, G is reduced to a terminal path, see Figure 3.2.(b). If $k \geq 3$, G contains $\binom{k}{k'}$ star trees (resp. clique stars) as subgraphs with $k' \in \{1, \ldots, k\}$ terminals. Let Π (resp. Θ) be the set of all

star trees (resp. clique stars) subgraphs of G. Note that Π (resp. Θ) contains the star tree G (resp. the clique star G). For $v \in V$, let Π^v be the set of star trees of Π (resp. Θ^v the set of clique stars of Θ) containing v. For all $l \in \{1, \ldots, k\}$, let Π_l be the set of star trees of Π (resp. Θ_l the set of clique stars of Θ) with l branches.

In this Section, we assume that all star trees and clique stars satisfy the following Hypotheses

3.1- the number of terminals is at least three.

3.2- each branch of the star tree contains at least one internal vertex.

Otherwise, linear system (2.6), (2.8) is TDI, [95].

3.1.1 Polytope characterization

In what follows, we give the complete description of the MTVS polytope for the star trees and clique stars.

3.1.1.1 MTVS polytope for star trees

Proposition 3.2 *In star trees, the polytope given by (2.6) and the trivial inequalities is not integral.*

Proof. Consider a star tree $H = (V(H) \cup T(H), E(H))$ with at least one vertex in each branch. Let $\overline{x} \in [0,1]^{V(H)}$ be defined as follows

- $\overline{x}(v) = 0.5 \qquad \forall v \in N(v_r)$,
- $\overline{x}(v) = 0 \qquad \forall v \in V(H) \setminus N(v_r)$.

Vector $\overline{x}$ is a fractional extreme point of $P(H, T(H))$, since there are $t(H)$ terminal path inequalities and $|V(H)| - t(H)$ trivial inequalities, that are linearly independent and tight for $\overline{x}$. ■

Consider the following valid inequalities,

$$x(V(H) \setminus \{v_r\}) + (t(H) - 1)x(v_r) \geq \quad t(H) - 1 \quad \forall H \in \Pi \tag{3.1}$$

presented in Section 2.3. We recall that the terminal paths are star trees of two terminals. Thus, inequalities (2.6) are included in (3.1). Let us remark that all inequalities (3.1) associated with the star trees having one terminal are dominated by the trivial inequalities.

Theorem 3.3 *For star trees, the MTVS polytope is given by inequalities* (3.1) *and the trivial inequalities.*

Proof. Let us assume the contrary, and let x^* be a fractional extreme point of polytope $P(G,T)$ associated with star tree $G = (V \cup T, E)$, where $|V|$ is minimum (*i.e.,* for all star trees of n' vertices with $n' < |V|$, the associated polytope is integral). Thus, x^* satisfies a unique system of linear independent equalities $\mathcal{A}$

$$\begin{aligned} x(V(H) \setminus \{v_r\}) + (t(H) - 1)x(v_r) &= t(H) - 1 \quad \forall H \in \Pi_1, && (3.2)\\ x^*(v) &= 1 \quad \forall v \in V_1, && (3.3)\\ x^*(v) &= 0 \quad \forall v \in V_2 && (3.4) \end{aligned}$$

such that $|\Pi_1| + |V_1| + |V_2| = |V|$, $\Pi_1 \subseteq \Pi$, $V_1 \subseteq V$ and $V_2 \subseteq V$.

Claim 1. For all $v \in V \setminus \{v_r\}$, $x^*(v) > 0$.

Proof. Let us assume the contrary, and suppose there exists a vertex $v \in V \setminus \{v_r\}$ such that $x^*(v) = 0$. It is clear that v has exactly two neighbors, $v_1, v_2 \in V$. Let H be the star tree obtained from G by deleting v and adding an edge v_1v_2. Let $\overline{x} \in [0,1]^{V(H)}$ be the restriction of x^* on $V(H)$ and $\mathcal{A}'$ the system of equalities obtained from $\mathcal{A}$ by deleting equality $x^*(v) = 0$ and variable $x^*(v)$ from all the other equalities. Solution $\overline{x}$ satisfies all equalities of $\mathcal{A}'$. We suppose there are two solutions $\overline{x}_1, \overline{x}_2 \in [0,1]^{V(H)}$ such that $\overline{x} = \frac{1}{2}(\overline{x}_1 + \overline{x}_2)$. Let $x_1^*, x_2^* \in [0,1]^V$ be two solutions defined as follows

- $x_1^*(u) = \overline{x}_1(u) \qquad \forall u \in V(H)$.
- $x_2^*(u) = \overline{x}_2(u) \qquad \forall u \in V(H)$.
- $x_1^*(v) = x_2^*(v) = 0$.

Hence, $x^* = \frac{1}{2}(x_1^* + x_2^*)$, which is a contradiction with the extremality of x^*. Thus, $\overline{x}$ is a fractional extreme point of $P(H, T(H))$. But this yields a contradiction with the fact that $|V|$ is minimum. □

Claim 2. For each branch P_t of G, $x^*(P_t) \leq 1$.

Proof. Let us assume the contrary, and let P_t be a branch of G such that $x^*(P_t) > 1$. Thus, the variables associated with internal vertices of P_t must only belong to (3.3). Otherwise, there would exist a vertex $v \in P_t \setminus \{v_r\}$ such that $x^*(v)$ belongs either to (3.2) or (3.4). From Claim 1, $x^*(v)$ cannot belong to (3.4). Suppose that $x^*(v)$ belongs to an equality $ax^* = b$, of type (3.2). Let H be the star tree obtained from G by deleting the vertices of $P_t \setminus \{v_r\}$. Hence, the star tree inequality $ax - x(P_t) \geq b - 1$ associated with H is violated by x^*, a contradiction. Therefore, the variable of each vertex of P_t belongs to a unique equality, namely (3.3). Let $\overline{x} \in [0,1]^{V(H)}$ be the restriction of x^* on $V(H)$ and $\mathcal{A}$ the system of equalities obtained from $\mathcal{A}'$, by deleting equality $x^*(v) = 1$ for all $v \in P_t \setminus \{v_r\}$. Solution $\overline{x}$ satisfies all equalities of $\mathcal{A}'$ and then it is a fractional extreme point of $P(H, T(H))$. However, this contradicts the fact that $|V|$ is minimum. □

Claim 3. For root vertex v_r, $x^*(v_r) < 1$.

Proof. Let us assume the contrary, $x^*(v_r) = 1$. Since x^* is fractional, there must exist $v \in V \setminus \{v_r\}$ such that $x^*(v)$ is fractional. Thus $x^*(v)$ must belong to equalities (3.2). As $x^*(v_r) = 1$, this yields $x^*(v) = 0$, which contradicts the fact that x^* is fractional. □

Claim 4. Each branch P_t of G contains at most one internal vertex.

Proof. Suppose there exists a branch P_t of G that contains at least two vertices u and v. From Claims 1 and 2, the variables associated with the internal vertices of P_t are fractional and cannot appear separately in (3.2) (*i.e.,* if a variable of a vertex $v_i \in P_t$ appears in an equality (3.2), all the variables associated with the vertices of P_t appear in the same equality). There must exist a constant $\epsilon > 0$ such that solution $\overline{x} \in [0,1]^V$, obtained from x^* by subtracting ϵ from $x^*(u)$ and adding ϵ to $x^*(v)$, is feasible and satisfies system $\mathcal{A}$. This contradicts the extremality of x^*. □

Claim 5. If there exists a branch P_t such that $x^*(P_t) < 1$, then all support graphs of equalities (3.2), contain branch P_t.

Proof. Let $ax^* = b$ be an equality (3.2) not containing the variables associated with the vertices of $P_t \setminus \{v_r\}$. Thus, the star tree inequality $x(P_t) + ax \geq b + 1$ is violated by x^*, which contradicts the fact that x^* is feasible. □

Claim 6. For root vertex v_r, $x^*(v_r) > 0$.

Proof. Suppose $x^*(v_r) = 0$. Since x^* is fractional, there must exist a vertex $v \in V \setminus \{v_r\}$ such that $x^*(v)$ is fractional. Let us assume that $v \in P_t \setminus \{v_r\}$. Variable $x^*(v)$ has a coefficient 1 in all inequalities. As $x^*(v)$ must belong to equalities (3.2), there must exist another vertex $u \in V \setminus \{v, v_r\}$ such that $x^*(u)$ is fractional, and $u \in P_{t'}$ with $t' \in T \setminus \{t\}$. From Claims 4 and 5, it follows that P_t and $P_{t'}$ must belong to all support graphs of equalities (3.2). There must exist a constant $\epsilon > 0$ such that the solution $\overline{x} \in [0,1]^V$ obtained from x^* by subtracting ϵ from $x^*(v)$ and adding ϵ to $x^*(u)$, is feasible and satisfies system of equalities $\mathcal{A}$. This contradicts the extremality of x^*. □

From Claims 1, 2, 3 and 6 we obtain that for all $v \in V$, $0 < x^*(v) < 1$. Now, we distinguish two cases.

a. There exists a branch P_t such that $x^*(P_t) < 1$.
   Hence there exists $\epsilon > 0$ such that the vector $\overline{x} \in \mathbb{R}^V$ defined as follows

   - $\overline{x}(v) = x^*(v)$ for $v \in P_t \setminus \{v_r\}$,
   - $\overline{x}(v_r) = x^*(v_r) + \epsilon$,
   - $\overline{x}(v) = x^*(v) - \epsilon$, for all $v \in V \setminus (P_t \cup \{v_r\})$

   satisfies all equalities of $\mathcal{A}$. This contradicts the fact that x^* is a fractional extreme point.

b. For each branch P_t, $x^*(P_t) = 1$.
   We notice that $x^*(v) = 1 - x^*(v_r)$ for all $v \in V \setminus \{v_r\}$. Since all variables are fractional, there exists at least one equality (3.2) satisfied with equality by x^*,

$$x^*(V \setminus \{v_r\}) + (t(G) - 1)x^*(v_r) = t(G) - 1.$$

   By replacing $x^*(v)$ by $1 - x^*(v_r)$, for all $v \in V \setminus \{v_r\}$, we obtain

$$|V(G) \setminus \{v_r\}| - |V(G) \setminus \{v_r\}|x^*(v_r) + (t(G) - 1)x^*(v_r) = t(G) - 1.$$

By Claim 4, since $|V \setminus \{v_r\}| = t(G)$, we have

$$t(G) - t(G)x^*(v_r) + (t(G) - 1)x^*(v_r) = t(G) - 1.$$

It follows that $x^*(v_r) = 1$, and, consequently, x^* is integer, which yields again a contradiction with the fact that x^* is fractional. ■

3.1.1.2 MTVS polytope for clique stars

Proposition 3.4 *For clique stars, the polytope given by (2.6) and the trivial inequalities is not integral.*

Proof. Consider a clique star $Q = (V(Q) \cup T(Q, E(Q))$ defined by q branches and a clique K_q. Let $\overline{x}$ be the solution defined as follows

- $\overline{x}(v) = 0.5 \qquad \forall v \in K_q,$
- $\overline{x}(v) = 0 \qquad \forall v \in V(Q) \setminus K_q.$

Vector $\overline{x}$ represents a fractional extreme point of $P(Q, T(Q))$, since there is $t(Q)$ terminal path inequalities and $|V(Q)| - t(Q)$ trivial inequalities, that are linearly independent and tight for $\overline{x}$. ■

Consider the following valid inequalities,

$$x(Q) \geq \quad t(Q) - 1 \qquad \forall Q \in \Theta. \tag{3.5}$$

We recall that the terminal paths are clique stars of two terminals. Thus, inequalities (2.6) are included in (3.5). Let us remark that all inequalities (3.5) associated with the clique stars having one terminal are dominated by trivial inequalities.

Theorem 3.5 *For clique stars, the MTVS polytope is given by inequalities* (3.5) *and the trivial inequalities.*

Proof. Let us assume the contrary, let x^* be a fractional extreme point of the polytope associated with $G = (V \cup T, E)$, where $|V|$ is minimum (*i.e.,* for all clique stars of

$n' < |V|$ vertices, the associated polytope is integer). Thus, x^* satisfies a unique system of linear independent equalities $\mathcal{A}$

$$x(V(Q)) = t(Q) - 1 \qquad \forall Q \in \Theta_1, \tag{3.6}$$
$$x^*(v) = 1 \qquad \forall v \in V_1, \tag{3.7}$$
$$x^*(v) = 0 \qquad \forall v \in V_2, \tag{3.8}$$
$$\tag{3.9}$$

such that $|\Theta_1| + |V_1| + |V_2| = |V|$, $\Theta_1 \subseteq \Theta$, $V_1 \subseteq V$ and $V_2 \subseteq V$.

Claim 1. For all $v \in V \setminus K_q$, $x^*(v) > 0$.

Proof. We assume that there exists a vertex $v \in V \setminus K_q$ such that $x^*(v) = 0$. It is clear that v has exactly two neighbors, $v_1, v_2 \in V$. Let Q be the star tree obtained from G by deleting v and adding edge v_1v_2. Let $\overline{x} \in [0,1]^{V(Q)}$ be the restriction of x^* on $V(Q)$ and $\mathcal{A}'$ the system of equalities obtained from $\mathcal{A}$, by deleting equality $x^*(v) = 0$ and variable $x^*(v)$ from all other equalities. Solution $\overline{x}$ satisfies all equalities of $\mathcal{A}'$ and this implies that it is a fractional extreme point of $P(Q, T(Q))$. Otherwise, there would exist two solutions $\overline{x}_1, \overline{x}_2 \in [0,1]^{V(Q)}$ such that $\overline{x} = \frac{1}{2}(\overline{x}_1 + \overline{x}_2)$. Let $x_1^*, x_2^* \in [0,1]^V$ be the solutions defined as follows

- $x_1^*(u) = \overline{x}_1(u) \qquad \forall u \in V(Q)$.
- $x_2^*(u) = \overline{x}_2(u) \qquad \forall u \in V(Q)$.
- $x_1^*(v) = x_2^*(v) = 0$.

As $x^* = \frac{1}{2}(x_1^* + x_2^*)$, this contradicts the extremality of x^*. Thus, $\overline{x}$ is a fractional extreme point of $P(Q, T(Q))$. This yields a contradiction with the fact that $|V|$ is minimum. □

Claim 2. For all branch P_t, $x^*(P_t) \leq 1$.

Proof. Let us assume the contrary, and let P_t be a branch such that $x^*(P_t) > 1$. Thus, the variables associated with the internal vertices of P_t must only belong to (3.7). Otherwise, there would exist a vertex $v \in P_t \setminus K_q$ such that $x^*(v)$ belongs either to (3.6) or (3.8). From Claim 1, $x^*(v)$ cannot belong to (3.8). Suppose that $x^*(v)$ belongs to an equality $ax^* = b$, of type (3.6). Hence, inequality $ax - x(P_t) \geq b - 1$

associated with clique star Q obtained from G by deleting the branch P_t, is violated by x^*. Contradiction. Therefore, the variable of each vertex of P_t belongs to a unique equality, that is (3.7). Let $\overline{x} \in [0,1]^{V(Q)}$ be the restriction of x^* on $V(Q)$ and $\mathcal{A}$ the system of equalities obtained from $\mathcal{A}'$ by deleting equality $x^*(v) = 1$ for all $v \in P_t \setminus \{v_r\}$. Solution $\overline{x}$ satisfies all equalities of $\mathcal{A}'$ and then it is a fractional extreme point. This is a contradiction with the minimality of $|V|$. □

Claim 3. For all $v \in K_q$ not adjacent to a terminal, $x^*(v) > 0$.

Proof. Assume that on the contrary, there exists a vertex $v \in K_q$ adjacent to a vertex $u \in P_t$ such that $x^*(v) = 0$. Let Q be the clique star obtained from G by deleting vertex v and adding all edges between u and all the neighbors of v. Let $\overline{x} \in [0,1]^{V(Q)}$ be the restriction of x^* on $V(Q)$ and $\mathcal{A}'$ the system of equalities obtained from $\mathcal{A}$, by deleting equality $x^*(v) = 0$ and variable $x^*(v)$ from all other equalities. Solution $\overline{x}$ satisfies all equalities of $\mathcal{A}'$ which implies that it is a fractional extreme point of $P(Q, T(Q))$, contradicting the minimality of $|V|$. □

Claim 4. Each vertex of K_q is adjacent to a terminal.

Proof. Let us assume that there exists a branch P_t containing $v \in K_q$ and an internal vertex u. From Claims 1, 2 and 3, we have that $0 < x^*(u) < 1$ and $0 < x^*(v) < 1$. There must exist $\epsilon > 0$ such that solution $\overline{x} \in [0,1]^V$, obtained from x^* by adding ϵ to $x^*(v)$ and subtracting ϵ from $x^*(u)$, satisfies all equalities of $\mathcal{A}$. Which yields a contradiction with the extremality of x^*. □

As x^* is fractional there must exist one vertex $u \in K_q$ such that $0 < x^*(u) < 1$. Since all coefficients of the variables in $\mathcal{A}$ are 0 or 1, it follows that, there exists another vertex $v \in K_q$ such that $0 < x^*(v) < 1$. We will prove that $x(u)$ and $x(v)$ belong together to the same equalities in $\mathcal{A}$. Suppose there exists an equality (3.6), associated with a clique star $Q \in \Theta_1$, containing u but not v. This yields

$$x^*(Q) = t(Q) - 1.$$

Let Q' be the clique star obtained from Q by adding the branch P_t containing v. As $x^*(P_t) < 1$, the clique star inequality induced by Q' is violated by x^*. Thus, u and v belong together to the same equalities. Let $y^* \in \mathbb{R}^V$ defined as follows

- $y^*(w) = x^*(w)$ for all $w \in V \setminus \{u, v\}$,

- $y^*(u) = x^*(u) + \epsilon$,

- $y^*(v) = x^*(v) - \epsilon$.

Solution y^* is a solution for $\mathcal{A}$. As $x^* \neq y^*$, this yields a contradiction with the extremality of x^*. ■

3.1.2 TDI-ness

In what follows, we give TDI descriptions for the multi-terminal vertex separator problem in the star trees and the clique stars.

3.1.2.1 TDI system for star trees

Consider a graph $G = (V \cup T, E)$ that is a star tree. We first introduce some notations. Consider two star trees H_i and H_j subgraphs of G. We denote by $H_{i,j}^{\cap}$ the star tree subgraph of G whose branches are all those in common with H_i and H_j and by $H_{i,j}^{\cup}$ a star tree subgraph of G with $t(H_i) + t(H_j) - t(H_{i,j}^{\cap})$ terminals, whose branches belong either to H_i or to H_j. If $t(H_{i,j}^{\cap}) = 0$, $H_{i,j}^{\cup}$ is any star tree of $t(H_i) + t(H_j) - 1$ branches.

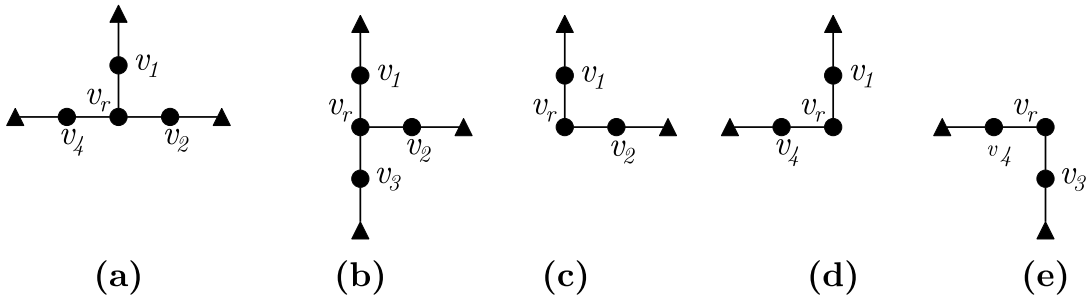


Figure 3.3: Star trees, subgraphs of the star tree in Figure 3.1.(a)

To illustrates these notations, if H_i is the graph in Figure 3.3.(c) and H_j the graph in Figure 3.3.(e), then $H_{i,j}^{\cup}$ is the graph in Figure 3.3.(a) or the graph in Figure 3.3.(b) and $H_{i,j}^{\cap}$ does not exist. If H_i is the graph in Figure 3.3.(a) and H_j the graph in Figure 3.3.(b), then $H_{i,j}^{\cup}$ the graph in Figure 3.4 and $H_{i,j}^{\cap}$ is the graph in Figure 3.3.(c).

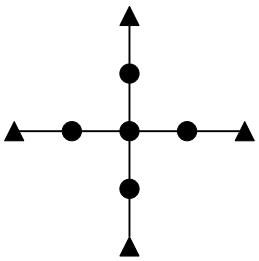


Figure 3.4: Star tree

Let P^G be the linear program defined by variable vector x, objective function (2.5), inequalities (3.1) and trivial inequalities (2.8), as follows

$$\begin{aligned} \min \quad & \sum_{v \in V} w(v)x(v) \\ x(V(H) \setminus \{v_r\}) + (t(H)-1)x(v_r) \geq \quad & t(H)-1 \quad \forall H \in \Pi, \qquad (3.10) \\ x(v) \leq \quad & 1 \quad \forall v \in V, \\ x(v) \geq \quad & 0 \quad \forall v \in V. \end{aligned}$$

The dual D^G of P^G, is given by

$$\begin{aligned} \max \quad & \sum_{H \in \Pi} (t(H)-1)y_H \\ \sum_{H \in \Pi^v} y_H \leq \quad & w(v) \quad \forall v \in V \setminus \{v_r\}, \qquad (3.11) \\ \sum_{H \in \Pi} (t(H)-1)y_H \leq \quad & w(v_r), \qquad (3.12) \\ y_H \geq \quad & 0 \quad \forall H \in \Pi. \qquad (3.13) \end{aligned}$$

where y_H is the dual variable associated with (3.10).

We notice that D^G consists in packing star trees of Π in G satisfying vertex capacity ω. Let $y^* \in \mathbb{R}_+^{\Pi}$ be an optimal solution of D^G.

Solution y^* is called *maximal optimal* if for each other optimal solution $\overline{y} \in \mathbb{R}_+^{\Pi}$ there exists $s \in \{1, \ldots, k\}$ satisfying the following.

1) $\sum_{H \in \Pi_l} \overline{y}_H = \sum_{H \in \Pi_l} y^*_H$, for all $l \in \{s+1, \ldots, k\}$,

2) $\sum_{H\in\Pi_s} \overline{y}_H < \sum_{H\in\Pi_s} y^*_H$.

Let y^* be a maximal optimal solution of D^G, such a solution exists.

Lemma 3.6 *For each pair of star trees $H_i, H_j \in \Pi$, such that $y^*_{H_i} > 0$ and $y^*_{H_j} > 0$, either $H_i \subseteq H_j$ or $H_j \subseteq H_i$.*

Proof. Suppose that H_i and H_j are such that $y^*_{H_j} \geq y^*_{H_i} > 0$ and no one is included in the other. Let $\beta = t(H^{\cup}_{i,j})$ and $\alpha = \max\{t(H^{\cap}_{i,j}), 1\}$. We will show that y^* cannot be maximal optimal by constructing an optimal solution $\overline{y} \in \mathbb{R}^{\Pi}_+$ of D^G such that

1) $\sum_{H\in\Pi_l} \overline{y}_H = \sum_{H\in\Pi_l} y^*_H$, for all $l \in \{\beta+1, \ldots, k\}$,

2) $\sum_{H\in\Pi_\beta} \overline{y}_H > \sum_{H\in\Pi_\beta} y^*_H$,

which will contradict the maximal optimality of y^*. Let $\overline{y}$ be defined as follows

- $\overline{y}_{H^{\cup}_{i,j}} = y^*_{H^{\cup}_{i,j}} + y^*_{H_i}$,
- $\overline{y}_{H_i} = 0$,
- $\overline{y}_{H_j} = y^*_{H_j} - y^*_{H_i}$,
- $\overline{y}_{H^{\cap}_{i,j}} = y^*_{H^{\cap}_{i,j}} + y^*_{H_i}$,
- $\overline{y}_H = y^*_H \qquad \forall H \in \Pi \setminus \{H_l, H_s, H^{\cap}_{i,j}, H^{\cup}_{i,j}\}$.

First, we show that all inequalities (3.11)-(3.13) remain satisfied by $\overline{y}$.

Each vertex $v \in V(H^{\cap}_{i,j}) \setminus \{v_r\}$ belongs to $V(H_i), V(H_j), V(H^{\cap}_{i,j})$ and $V(H^{\cup}_{i,j})$. Since the coefficient of each vertex of $V(H^{\cap}_{i,j}) \setminus \{v_r\}$ is 1 in all inequalities, by adding $y^*_{H_i}$ to the variable of $H^{\cup}_{i,j}$ and to the one of $H^{\cap}_{i,j}$ and by subtracting $y^*_{H_i}$ from the variable of H_i and from the one of $H^{\cap}_{i,j}$, it follows that all inequalities associated with the vertices of $V(H^{\cap}_{i,j}) \setminus \{v_r\}$ remain satisfied by $\overline{y}$.

Each vertex v of $V(H_i) \setminus V(H^{\cap}_{i,j})$ (resp. $V(H_j) \setminus V(H^{\cap}_{i,j})$) only belongs to $V(H_i)$ and $V(H^{\cup}_{i,j})$ (resp. $V(H_j)$ and $V(H^{\cup}_{i,j})$). Thus, since the coefficient of each vertex of

$V(H_i) \setminus V(H^{\cap}_{i,j})$ (resp. $V(H_j) \setminus V(H^{\cap}_{i,j})$ is 1, by adding $y^*_{H_i}$ to the variable of $H^{\cup}_{i,j}$ and by subtracting $y^*_{H_i}$ from the variable of H_i (resp. H_j), it follows that all inequalities associated with the vertices of $V(H_i) \setminus V(H^{i \cap j})$ (resp. $V(H_j) \setminus V(H^{\cap}_{i,j})$) remain satisfied by $\overline{y}$.

It remains to show that inequality (3.12) associated with v_r is also satisfied by $\overline{y}$. Let us remark that the left hand side of inequality (3.12) associated with v_r is precisely the function that we want to maximize. Consequently, proving the optimality of $\overline{y}$ reduces to proving its feasibility.

As $\overline{y}_H = y^*_H$ for all $H \in \Pi \setminus \{H_i, H_j, H^{\cap}_{i,j}, H^{\cup}_{i,j}\}$, to prove the optimality of $\overline{y}$ we just need to prove that

$$(t(H_i)-1)\overline{y}_{H_i} + (t(H_j)-1)\overline{y}_{H_j} + (\beta-1)\overline{y}_{H^{\cup}_{i,j}} + (\alpha-1)\overline{y}_{H^{\cap}_{i,j}} =$$
$$(t(H_i)-1)y^*_{H_i} + (t(H_j)-1)y^*_{H_j} + (\beta-1)y^*_{H^{i \cup j}} + (\alpha-1)y^*_{H^{\cap}_{i,j}}$$

Since $\beta = t(H_i) + t(H_j) - \alpha$, by replacing the values of $\overline{y}$ we obtain that

$$(l-1)\overline{y}_{H_i} + (s-1)\overline{y}_{H_j} + (\beta-1)\overline{y}_{H^{\cup}_{i,j}} + (\alpha-1)\overline{y}_{H^{\cap}_{i,j}} =$$

$$0 + (t(H_j)-1)(y^*_{H_j} - y^*_{H_i}) + (t(H_i)+t(H_j)-\alpha-1)(y^*_{H^{\cup}_{i,j}} + y^*_{H_i}) + (\alpha-1)(y^*_{H^{\cap}_{i,j}} + y^*_{H_i})$$
$$= (t(H_j)-1)y^*_{H_j} + (t(H_i)-1)y^*_{H_i} + (\beta-1)y^*_{H^{\cup}_{i,j}} + (\alpha-1)y^*_{H^{\cap}_{i,j}}.$$

Consequently, $\overline{y}$ is a feasible optimal solution. This contradicts the fact that y^* is maximal optimal. ■

Corollary 3.7 *For $l \in \{1, \ldots, k\}$, there is at most one star tree H over all star trees of Π_l, with $y^*_H > 0$.*

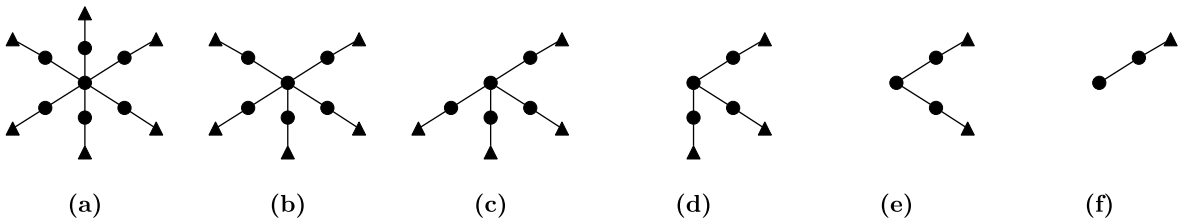


Figure 3.5: A maximal optimal solution structure

Figure 3.5 illustrates the structure of the maximal optimal solution y^* (each star tree is included in another, except G).

Theorem 3.8 *For star tree $G = (V \cup T, E)$, linear system of P^G is TDI.*

Proof. We should prove that D^G has an integer optimal solution. For this, we need to show the claims below.

Claim 1. If there exists a star tree $H \in \Pi$, such that $y^*_H > 0$, and $\sum_{l=t(H)}^{k} \sum_{H \in \Pi_l} y^*_H < w_v$, for all $v \in V(H)$, then $y^*_{H'} = 0$ for each star tree $H' \in \Pi$ with $t(H') < t(H)$.

Proof. Let us assume the contrary. Suppose there exists a star tree $H \in \Pi$, such that $y^*_H > 0$, and $\sum_{l=t(H)}^{k} \sum_{H \in \Pi_l} y^*_H < w_v$, for all $v \in V(H)$, and that there exists a star tree $H' \in \Pi$ with $t(H') < t(H)$, such that $y^*_{H'} > 0$. Assume that $t(H')$ is maximum (for all $H'' \in \Pi$ with $t(H'') \in \{t(H')+1, \ldots, t(H)-1\}$, $y^*_{H''} = 0$).

To prove the claim, we will show that y^* cannot be maximal optimal by constructing another solution $\overline{y} \in \mathbb{R}^{\Pi}_{+}$ from y^*. Let $\overline{y}$ be obtained from y^* by adding $\alpha > 0$ to y^*_H and subtracting $\beta > 0$ from $y^*_{H'}$. Indeed, proving that $\overline{y}$ is feasible and optimal, will contradict the fact that y^* is maximal optimal. To guarantee the optimality of $\overline{y}$, we need to satisfy the following equality

$$\alpha.(t(H)-1) = \beta.(t(H')-1)$$

Thus, if $\alpha = \min\{\frac{y^*_{H'}(t(H')-1)}{(t(H)-1)}, \min_{v \in V(H) \setminus \{v_r\}} \{\omega(v) - \sum_{l=t(H)}^{k} \sum_{H \in \Pi_l} y^*_H\}\}$, then $\alpha > 0$ and $\beta = \frac{\alpha(t(H)-1)}{t(H')-1}$.

As $t(H')$ is maximum, from Lemma 3.6 $\overline{y}$ must be feasible optimal solution for D^G. This contradicts the fact that y^* is maximal optimal. □

Claim 2. If y^* is fractional, then there exists exactly one star tree $H \in \Pi$ such that y^*_H is fractional.

Proof. Assume that there exist two different star trees $H_i, H_j \in \Pi$, such that $y^*_{H_i}$ and $y^*_{H_j}$ are fractional. Suppose that $t(H_j)$ is maximum (that is for all $H \in \Pi$ with $t(H) \geq t(H_j)$, $y^*_{H_j}$ is integer). From Corollary 3.7, $t(H_j) > t(H_i)$. We distinguish two cases.

1. There exists a vertex $v \in V(H_j) \setminus \{v_r\}$ such that $\sum_{l=t(H_j)}^{k} \sum_{H \in \Pi_l} y^*_H = \omega(v)$. As $t(H_j)$ is maximum, y^*_H is integer for any star tree H with $t(H) \in \{t(H_j)+1, \ldots, k\}$.

Moreover, since $\omega(v)$ is integer, it follows that $y^*_{H_j}$ is integer, contradicting that fact that $y^*_{H_j}$ fractional.

2. For all vertex $v \in V(H_j \setminus \{v_r\}$, we have $\sum_{l=t(H_j)}^{k} \sum_{H \in \Pi_l} y^*_H < \omega(v)$.

   From Claim 1, $y^*_H = 0$ for any star tree $H \in \Pi$ with $t(H) \leq t(H_j)-1$, a contradiction with $y^*_{H_i}$ fractional.

Thus, there exists at most one star tree $H_j \in \Pi$ such that $y^*_{H_j}$ is fractional. □

Claim 3. If y^* is fractional then there exists another optimal solution $\overline{y}$ that is integer.

Proof. From Claim 2, since y^* is fractional, there exists exactly one star tree $H \in \Pi$ such that y^*_H is fractional. We distinguish two cases

- If $t(H) = 1$, then let $\overline{y} \in \mathbb{R}^{\Pi}_+$ be the solution obtained from y^* by setting $y^*_H = 0$. Vector $\overline{y}$ represents an integer feasible optimal solution for D^G.

- If $t(H) \geq 2$, then by Claim 2 and from the fact that y^* is optimal and $\omega(v_r)$ is integer, we have that $(t(H)-1)y^*_H$ is integer. Let $\epsilon = y^*_H - \lfloor y^*_H \rfloor$. Thus, $(t(H)-1)\epsilon$ is integer. Let $\overline{y} \in \mathbb{R}^{\Pi}_+$ be another solution obtained from y^* by subtracting ϵ from y^*_H and by adding 1 to $y^*_{H'}$ for an arbitrary star tree $H' \in \Pi_{\epsilon.(t(H)-1)+1}$. Consequently, $\overline{y}$ is an integer optimal solution for D^G. □

Which ends the proof of the theorem. ■

As consequence, we obtain the following *min-max* relation:

Corollary 3.9 *In star trees, the minimum number of vertices covering all terminal paths is equal to the maximum packing of star trees.*

Then, we deduce the following polynomial algorithm to solve D^G. It is based on the fact that the maximal optimal solution $\overline{y}$ of D^G is integer and any star tree of $\overline{y}$ is included in another star tree of $\overline{y}$, except the maximal one. The algorithm starts by packing several copies of the maximal star tree of G until the capacity of at least one vertex of G is exhausted. We delete then all branches of G containing at least one

vertex with an exhausted capacity. The above operations are repeated until G becomes a branch.

Therefore, we deduce the following algorithm (Algorithm 11) for solving D^G.

Algorithm 11: An exact algorithm for solving D^G

Data: A star tree $G = (V \cup T, E)$, a vertex capacities vector ω
Result: Maximal optimal solution y^*
begin
  for $(t \in T)$ **do**
    $\omega'(P_t) = \min_{u \in P_t \setminus \{v_r\}} \{\omega(u)\}$;
    if $(\omega'(P_t) = 0)$ **then**
      Delete branch P_t from G;
    end
  end
  while $(t(G) > 1$ *and* $\omega(v_r) > 0)$ **do**
    $y^*_G = \min\{\lfloor \frac{\omega(v_r)}{t(G)-1} \rfloor, \min_{t \in T}\{\omega'(P_t)\}\}$;
    for $(t \in T)$ **do**
      $\omega'(P_t) = \omega'(P_t) - y^*_G$;
    end
    $\omega(v_r) = \omega(v_r) - y^*_G.(t(G) - 1)$;
    for $(t \in T)$ **do**
      if $(\omega'(P_t) = 0)$ **then**
        Delete branch P_t from G;
      end
    end
  end
end

Corollary 3.10 *For the star trees, the multi-terminal vertex separator problem is polynomial.*

3.1.2.2 TDI system for clique stars

Consider a graph $G = (V \cup T, E)$ that is a clique star. Given Q_i and Q_j two clique stars of G, we denote by $Q^{\cap}_{i,j}$ the clique star of G, whose branches are those in common

with Q_i and Q_j. We denote by $Q^{\cup}_{i,j}$ a clique star of G whose branches belong either to Q_i or to Q_j.

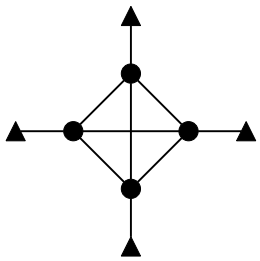


Figure 3.6: Clique star

Let P^G be the linear program whose objective function is (2.5), and the system of inequalities given by (2.8) and (3.5). Let D^G be the dual of P^G. Let us remark that D^G consists in a packing of clique stars of Θ in G satisfying the vertex capacities ω. Let $y \in \mathbb{R}^{\Theta}_{+}$ be vector of the dual variables associated with inequalities (3.5) and y^* an optimal solution of D^G. Solution y^* is called *maximal optimal* if for each optimal solution $\overline{y} \in \mathbb{R}^{\Theta}_{+}$ there exists $s \in \{1, \ldots, k\}$ satisfying the following conditions.

1) $\sum_{Q \in \Theta_l} \overline{y}_Q = \sum_{Q \in \Theta_l} y^*_Q, \qquad$ for all $l \in \{s+1, \ldots, k\}$,

2) $\sum_{Q \in \Theta_s} \overline{y}_Q < \sum_{Q \in \Theta_s} y^*_Q$.

In the following we assume that y^* is maximal optimal.

Lemma 3.11 *For two different clique stars $Q_i, Q_j \in \Theta$ of G, such that $y^*_{Q_i} > 0$ and $y^*_{Q_j} > 0$, either $Q_i \subseteq Q_j$ or $Q_j \subseteq Q_i$.*

Proof. Assume that there exist two clique stars $Q_i, Q_j \in \Theta$ of G, such that $y^*_{Q_i} > 0$ and $y^*_{Q_j} > 0$ and no one of them is included in the other. Then there exists $\epsilon > 0$ such that the solution $\overline{y} \in \mathbb{R}^{\Theta}_{+}$, obtained from y^* by subtracting ϵ from $y^*_{Q_i}$ and $y^*_{Q_j}$ and by adding ϵ to $y^*_{Q^{\cup}_{i,j}}$ and to $y^*_{Q^{\cap}_{i,j}}$, is feasible and optimal for D^*. This contradicts the fact that y^* is maximal optimal. ■

Corollary 3.12 *For $s \in \{1, \ldots, k\}$, there exists at most one clique star Q over all clique stars of Θ_s with $y^*_Q > 0$.*

Lemma 3.13 *For all $Q_j \in \Theta$, there exists a vertex $v \in V(Q_j)$ such that $\sum_{l=t(Q_j)}^{k} \sum_{Q \in \Theta_l} y^*_Q = w(v)$.*

Proof. Assume that there exists $Q_j \in \Theta$, such that for all $v \in V(Q_j)$, $\sum_{l=t(Q_j)}^{k} \sum_{Q \in \Theta_l} y^*_Q < w(v)$. Then, there must exist $Q' \in \Theta$ such that $t(Q') < t(Q_j)$ and $y^*_{Q'} > 0$, otherwise the solution would not be optimal. Assume that $t(Q')$ is maximum, that is, for all clique star $Q'' \in \Theta$ with $t(Q') < t(Q'') < t(Q_j)$, $y^*_{Q''} = 0$. Then there exists $0 < \epsilon \leq y^*_{Q'}$ such that $\overline{y} \in \mathbb{R}^{\Theta}_{+}$, obtained from y^* by subtracting ϵ from $y^*_{Q'}$ and adding ϵ to $y^*_{Q_j}$, is feasible and optimal for D^*. This contradicts the fact that y^* is maximal optimal. ■

Theorem 3.14 *For clique star $G = (V \cup T, E)$, the linear system of P^G is TDI.*

Proof. Using Corollary 3.12 and Lemma3.13, the maximal optimal solution of D^G can be obtained by packing several copies of the clique star with the highest number of terminals, until the capacity of one vertex is saturated. Then, we delete branches with at least one saturated vertex. We repeat the operations until the graph becomes a branch.

Therefore, we deduce the following algorithm (Algorithm 12) for solving D^G.

Algorithm 12: An exact algorithm for solving D^G

```
Data: Graph G = (V ∪ T, E) that is a clique star, a weight vector ω
Result: A maximal optimal solution y*
begin
    for (t ∈ T) do
        ω'(P_t) = min_{u∈P_t}{ω(u)} ;
        if (ω'(P_t) = 0) then
            Delete branch P_t from G;
        end
    end
    while (t(G) > 1) do
        y*_G = min_{∀t∈T}{ω'(P_t)};
        for (t ∈ T) do
            ω'(P_t) = ω'(P_t) − y*_G;
        end
        for (t ∈ T) do
            if (ω'(P_t) = 0) then
                Delete branch P_t from G;
            end
        end
    end
end
```

From Lemmas 3.11 and 3.13, Algorithm 12 gives an integer optimal solution y^* for D^*. Which ends the proof of the theorem. ■

As consequence, we obtain the following *min-max* relation:

Corollary 3.15 *In clique stars, the minimum number of vertices covering all terminal paths is equal to the maximum packing of clique stars.*

3.2 Composition of polyhedra by 1-sum

In what follows we study a composition (decomposition) technique for the multi-terminal vertex separator polytope in graphs that are decomposable by one-node cut-sets (1-sum). If G decomposes into G_1 and G_2, we show that the multi-terminal vertex

polytope of G can be described from two linear systems related to G_1 and G_2. As a consequence, we obtain a procedure to construct this polytope in graphs that are recursively decomposed.

Given a graph $G = (V \cup T, E)$ and two subgraphs of G, say $G_1 = (V_1 \cup T_1, E_1)$ and $G_2 = (V_2 \cup T_2, E_2)$, graph G is called a $k-sum$ of G_1 and G_2 if $V = V_1 \cup V_2$, $T = T_1 \cup T_2$, $|T_1 \cap T_2| = 0$, $|V_1 \cap V_2| = k$ and subgraph $(V_1 \cap V_2, E_1 \cap E_2)$ is complete. Set $V_1 \cap V_2$ is called a *k-node cutset* of G.

In what follows we investigate the polytope composition procedure for graphs that are 1-sums. Let $G = (V \cup T, E)$ be a 1-sum of $G_1 = (V_1 \cup T_1, E_1)$ and $G_2 = (V_2 \cup T_2, E_2)$. Let $V_1 \cap V_2 = \{u\}$, such that u is not adjacent to a terminal. Let $\widetilde{G}_i = (\widetilde{V}_i \cup \widetilde{T}_i, \widetilde{E}_i)$ be the graph obtained from G_i, for $i = 1, 2$, by adding a node w_i, a terminal q_i and edges $q_i w_i, w_i u$. Figure 3.7 illustrates graphs G, $\widetilde{G}_1$ and $\widetilde{G}_2$.

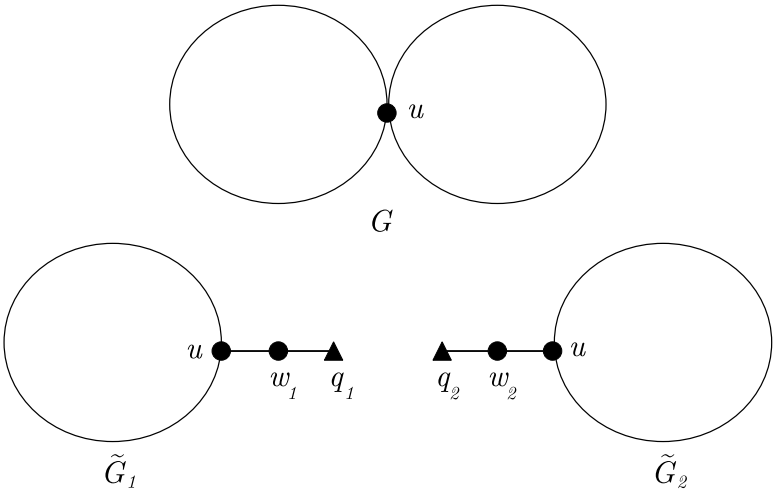


Figure 3.7: Composition (decomposition) of graphs

In $\widetilde{G}_i$, the multi-terminal vertex separator polytope $P(\widetilde{G}_i, \widetilde{T}_i)$ is completely described by a linear inequality system of the form

$$\sum_{v \in \widetilde{V}_i} a_l^i(v) x(v) \geq \alpha_l^i \quad \forall l \in L^i, \tag{3.14}$$

$$x(v) \leq 1 \quad \forall v \in \widetilde{V}^i,$$

$$x(v) \geq 0 \quad \forall v \in \widetilde{V}^i.$$

3.2.1 Structure properties

In what follows, we shall study some structural properties of the facets of $P(\widetilde{G}_i, \widetilde{T}_i)$. These properties will be used later for the composition of polyhedra.

Lemma 3.16 *For any inequality* (3.14), $a^i_l(v) \geq 0$ *for any vertex* $v \in \widetilde{V}_i$.

Proof. Let us assume the contrary, then there exists a vertex $v_0 \in \widetilde{V}_i$ such that $a^i_l(v_0) < 0$. As (3.14) is different from a bound inequality, there must exist a vertex separator S^i that does not contain v_0 and such that $\sum_{v \in \widetilde{V}_i \setminus \{v_0\}} a^i_l(v) x^{S^i}(v) = \alpha^i_l$. Let $S^{v_0} = S^i \cup \{v_0\}$. Set S^{v_0} is a separator for $\widetilde{G}_i$. It then follows that $\sum_{v \in \widetilde{V}_i \setminus \{v_0\}} a^i_l(v) x^{S^v}(v) + a^i_l(v_0) < \alpha^i_l$, a contradiction. ■

Lemma 3.17 *For any inequality* (3.14), $a^i_l(u) \geq a^i_l(w_i)$.

Proof. Since (3.14) defines a facet of $P(\widetilde{G}_i, \widetilde{T}_i)$ different from a non-negativity inequality and polytope $P(\widetilde{G}_i, \widetilde{T}_i)$ is full dimensional, there must exist a separator S^i such that x^{S^i} satisfies (3.14) with equality and $w_i \in S^i$. If $u \in S^i$, then $S^i \setminus \{w_i\}$ is a separator for $\widetilde{G}_i$, and this yields $a^i_l(w_i) = 0 \leq a^i_l(u)$. If $u \notin S^i$, then let $\overline{S}^i = (S^i \setminus \{w_i\}) \cup \{u\}$. As $\overline{S}^i$ is a separator for $\widetilde{G}_i$, it follows that $a^i_l(u) \geq a^i_l(w_i)$. ■

Lemma 3.18 *Consider a separator* S^i *for graph* $\widetilde{G}_i$. *If* x^{S^i} *satisfies an inequality* (3.14) *with equality, then* $|S^i \cap \{w_i, u\}| \leq 1$.

Proof. Let us assume the contrary, that is $\{w_i, u\} \subseteq S^i$ and x^{S^i} satisfies an inequality (3.14) with equality. Let $\overline{S}^i = S^i \setminus \{w_i\}$. It is clear that $\overline{S}^i$ is a separator for $\widetilde{G}_i$. It then follows that inequality (3.14) is violated by $\overline{S}^i$. ■

From Lemmas 3.16, 3.17, 3.18, the linear inequality system describing $P(\widetilde{G}_i, \widetilde{T}_i)$ can

be given as follows

$$\sum_{v \in V_i \setminus \{u\}} a^i_j(v)x(v) \geq \alpha^i_j \quad \forall j \in I^i, \tag{3.15}$$

$$\sum_{v \in V_i \setminus \{u\}} a'^i_j(v)x(v) + x(u) \geq \alpha'^i_j \quad \forall j \in I'^i, \tag{3.16}$$

$$\sum_{v \in V_i} b^i_j(v)x(v) + x(w_i) \geq \beta^i_j \quad \forall j \in J^i, \tag{3.17}$$

$$x(v) \leq 1 \quad \forall v \in \widetilde{V}^i, \tag{3.18}$$

$$x(v) \geq 0 \quad \forall v \in \widetilde{V}^i. \tag{3.19}$$

Where I^i is the set of inequalities whose support does not intersect $\{w_i, u\}$, I'^i the set of inequalities whose support does not intersect w_i but contains u and J^i the set of inequalities whose support contains u and w_i.

3.2.2 Composition of polyhedra

In what follows we derive a system of inequalities that describes $P(G,T)$. To this end, we first give the following lemmas.

Lemma 3.19 *For $j \in J^i$, inequality*

$$\sum_{v \in V_i} b^i_j(v)x(v) - x(u) \geq \beta^i_j - 1$$

is valid for $P(G,T)$.

Proof. Consider a separator S for graph G. We distinguish two cases

- $u \in S$: let $S_i = (S \cap V_i)$. It is clear that S_i is a separator of $\widetilde{G}_i$. It follows that $\sum_{v \in V_i} b^i_j(v)x^{S_i}(v) \geq \beta^i_j$, and since $-x^{S_i}(u) \geq -1$, it follows that $\sum_{v \in V_i} b^i_j(v)x^{S_i}(v) - x^{S_i}(u) \geq \beta^i_j - 1$.
- $u \notin S$: then either $(S \cap V_1)$ is a separator for $\widetilde{G}_1$ or $(S \cap V_2)$ is a separator for $\widetilde{G}_2$. Suppose for instance, that $(S \cap V_2)$ is a separator for $\widetilde{G}_2$, and let $S_1 = (S \cap V_1) \cup \{w_1\}$ and $S_2 = (S \cap V_2)$. Clearly, S_1 and S_2 are both separators for $\widetilde{G}_1$ and $\widetilde{G}_2$ respectively. Thus,

– for $\widetilde{G}_2$, we have $\sum_{v\in V_2} b_j^2(v)x^{S_2}(v) \geq \beta_j^2$ and since $-x^{S_2}(u) \geq -1$, this yields $\sum_{v\in V_2} b_j^2(v)x^{S_2}(v) - x^{S_2}(u) \geq \beta_j^2 - 1$.

– for $\widetilde{G}_1$, we have $\sum_{v\in V_1} b_j^1(v)x^{S_1}(v) + x^{S_1}(w_1) \geq \beta_j^1$ and $x^{S_1}(w_1) = 1$. Then, $\sum_{v\in V_1} b_j^1(v)x^{S_1}(v) \geq \beta_j^1 - 1$, and as $x^{S_1}(u) = 0$, we obtain that $\sum_{v\in V_1} b_j^1(v)x^{S_1}(v) - x^{S_1}(u) \geq \beta_j^1 - 1$.

Therefore we obtain that

$$\sum_{v\in V_1} b_j^1(v)x^S(v) - x^S(u) \geq \beta_j^1 - 1$$
$$\sum_{v\in V_2} b_j^2(v)x^S(v) - x^S(u) \geq \beta_j^2 - 1$$

and the lemma follows. ■

Lemma 3.20 *For $j \in J^i$, inequality*

$$\sum_{v\in V_i} b_j^i(v)x(v) - x(u) \geq \beta_j^i - 1$$

is valid for $P(\widetilde{G}_i, \widetilde{T}_i)$.

Proof. We assume w.l.o.g, that $i = 1$. The proof for $i = 2$ is similar. Let S^1 be a separator of $\widetilde{G}_1$. We distinguish two cases.

1- $u \in S^1$ or ($u \notin S^1$ and $w_1 \notin S^1$)
It can be easily seen that $S' = S^1 \cup (V_2 \setminus \{u\})$ is a separator for G. Therefore, $x^{S'}$ belongs to $P(G, T)$. From Lemma (3.19), $x^{S'}$ satisfies the following inequality

$$\sum_{v\in V_1} b_j^1(v)x^{S'}(v) - x^{S'}(u) \geq \beta_j^1 - 1.$$

Since $S^1 = S' \cap (V_1 \cup \{w_1\})$, we get

$$\sum_{v\in V_1} b_j^1(v)x^{S^1}(v) - x^{S^1}(u) \geq \beta_j^1 - 1.$$

2- $u \notin S^1$ and $w_1 \in S^1$
This implies, $x^{S^1}(u) = 0$ and $x^{S^1}(w_1) = 1$. As

$$\sum_{v \in V_1} b_j^1(v) x^{S^1}(v) + x^{S^1}(w_1) \geq \beta_j^1,$$

We obtain that

$$\sum_{v \in V_1} b_j^1(v) x^{S^1}(v) - x^{S^1}(u) \geq \beta_j^1 - 1.$$

And the lemma follows. ■

Theorem 3.21 *If $ax \geq \alpha$ is a valid inequality for $P(\widetilde{G}_i, \widetilde{T}_i)$ with $a(w_i) = 0$, then there exists an inequality $\overline{a}x \geq \overline{\alpha}$ valid for $P(\widetilde{G}_i, \widetilde{T}_i)$ that dominates $ax \geq \alpha$ and that is a linear combination of inequalities* (3.15), (3.16) *and inequalities* (3.18) *associated with the vertices of V_i.*

Proof. Let A' be the system given by inequalities $(3.15) - (3.17), (3.19)$ and inequalities (3.18) associated with the vertices of V_i. Let

$$P' = \min\{ \quad ax \quad | \quad x \text{is a solution of } A'\},$$

and

$$P'' = \min\{ \quad ax \quad | \quad x \in P(\widetilde{G}_i, \widetilde{T}_i)\}.$$

An optimal solution, say x_0, of P'' is a feasible solution of P'. Since $a(w_i) = 0$, we can assume that $x_0(w_i) = 1$.

Let $r \in \mathbb{R}^{I^i}, z \in \mathbb{R}^{I'^i}, \theta \in \mathbb{R}^{J^i}, \mu \in \mathbb{R}^{\widetilde{V}_i}, \lambda \in \mathbb{R}^{V_i}$ associated with inequalities (3.15), (3.16), (3.17),(3.19) and $x(v) \leq 1$, $\forall v \in V_i$, respectively. Let D' be the dual of P',

given by

$$\max \quad \sum_{j\in I^i} \alpha^i_j r_j + \sum_{j\in I'^i} \alpha'^i_j z_j + \sum_{j\in J^i} \beta^i_j \theta_j - \sum_{v\in V_i} \lambda(v)$$

$$\sum_{j\in I^i} a^i_j(v) r_j + \sum_{j\in I'^i} a'^i_j(v) z_j + \sum_{j\in J^i} b^i_j(v)\theta_j + \mu(v) - \lambda(v) \le a(v) \qquad \forall v \in \widetilde{V}_i \setminus \{u, w_i\} \tag{3.20}$$

$$\sum_{j\in I'^i} z_j + \sum_{j\in J^i} b^i_j(u)\theta_j + \mu(u) - \lambda(u) \le a(u) \tag{3.21}$$

$$\sum_{j\in J^i} b^i_j(w_i)\theta_j + \mu(w_i) \le a(w_i) \tag{3.22}$$

$$r^i_j \ge 0 \qquad \forall j \in I^i \tag{3.23}$$

$$z^i_j \ge 0 \qquad \forall j \in I'^i \tag{3.24}$$

$$\theta^i_j \ge 0 \qquad \forall j \in J^i \tag{3.25}$$

$$\mu(v) \ge 0 \qquad \forall v \in \widetilde{V}_i \tag{3.26}$$

$$\lambda(v) \ge 0 \qquad \forall v \in V_i \tag{3.27}$$

As for each vertex $v \in \widetilde{V}_i$, $\mu(v)$ is non-negative, the coefficient of $\mu(v)$ is 0 in the objective function and positive in all inequalities of D', we can set $\mu(v) = 0$ for all vertex $v \in \widetilde{V}_i$. As x_0 is a feasible solution of P', D' admits an optimal solution $(r^*, z^*, \theta^*, 0, \lambda^*)$. From Lemma 3.16, all coefficients in inequality (3.22) are positives. As $a(w_i) = 0$, we deduce that $\theta^*_j = 0$ for all $j \in J^i$. Let

- $\overline{a}(v) = \sum_{j\in I^i} a^i_j(v) r^*_j + \sum_{j\in I'^i} a'^i_j(v) z^*_j - \lambda^*(v)$ for all $v \in \widetilde{V}_i \setminus \{u, w_i\}$,
- $\overline{a}(u) = \sum_{j\in I'^i} z^*_j - \lambda^*(u)$,
- $\overline{a}(w_i) = 0$,
- $\overline{\alpha} = \sum_{j\in I^i} \alpha^i_j r^*_j + \sum_{j\in I'^i} \alpha'^i_j z^*_j - \sum_{v\in V_i} \lambda^*(v)$.

We can see that $\overline{a}x \ge \overline{\alpha}$ is a combination of inequalities (3.15), (3.16) and inequalities (3.18) using coefficients (r^*, z^*, λ^*). Then, $\overline{a}x \ge \overline{\alpha}$ is valid for $P(\widetilde{G}_i, \widetilde{T}_i)$.
From the inequalities of D', we notice that $\overline{a}(v) \le a(v)$ for all vertex $v \in \widetilde{V}_i$. Since $ax \ge \alpha$ and $\overline{\alpha} = \min\{ \quad ax \quad | \quad x \text{ solution of } A'\}$, it follows that $\overline{\alpha} \ge \alpha$. Consequently, inequality $\overline{a}x \ge \overline{\alpha}$ dominates $ax \ge \alpha$. ■

Consider the following system of inequalities

$$\sum_{v\in V_i\setminus\{u\}} a^i_j(v)x(v) \geq \alpha^i_j \qquad \forall i=1,2, \quad \forall j\in I^i \tag{3.28}$$

$$\sum_{v\in V_1\setminus\{u\}} a'^i_j(v)x(v) + x(u) \geq \alpha'^i_j \qquad \forall i=1,2, \quad \forall j\in I'^i \tag{3.29}$$

$$\sum_{p=1}^{2}\sum_{v\in V_i} b^i_{j_p}(v)x(v) - x(u) \geq \sum_{p=1}^{2}\beta^i_{j_p} - 1 \qquad \forall j_1\in J^1, \forall j_2\in J^2 \tag{3.30}$$

$$x(v) \leq 1 \qquad \forall v\in V \tag{3.31}$$

$$x(v) \geq 0 \qquad \forall v\in V \tag{3.32}$$

Inequalities (3.30) are called *mixed inequalities*. Let $Q(G,T)$ be the polytope given by inequalities (3.28) – (3.32).

In what follows will first show that $P(G,T) = Q(G,T)$.

Lemma 3.22 *The mixed inequalities are valid for $P(G,T)$.*

Proof. Consider a mixed inequality associated with $j_1 \in J^1$ and $j_2 \in J^2$. Consider a separator S for graph G. We distinguish two cases.

- $u \in S$: define $S_1 = (S\cap V_1)$ and $S_2 = (S\cap V_2)$. It is clear that S_1 and S_2 are two separators for $\widetilde{G}_1$ and $\widetilde{G}_2$ respectively. Thus, by (3.17) we have that

$$\sum_{v\in V_1} b^1_{j_1}(v)x^{S_1}(v) \geq \beta^1_{j_1},$$
$$\sum_{v\in V_2} b^2_{j_2}(v)x^{S_2}(v) \geq \beta^2_{j_2}.$$

  Thus $\sum_{v\in V_1} b^1_{j_1}(v)x^{S_1}(v) + \sum_{v\in V_2} b^2_{j_2}(v)x^{S_2}(v) - x^{S_1}(u) \geq \beta^1_{j_1} + \beta^2_{j_2} - 1$, and hence inequality (3.30) is satisfied for S.

- $u \notin S$: then either $(S\cap V_1)$ is a separator of $\widetilde{G}_1$ or $(S\cap V_2)$ is a separator of $\widetilde{G}_2$. Assume that $(S\cap V_2)$ is a separator of $\widetilde{G}_2$, and let $S_1 = (S\cap V_1)\cup\{w_1\}$ and $S_2 = (S\cap V_2)$. As, S_1 and S_2 are separators for $\widetilde{G}_1$ and $\widetilde{G}_2$ respectively, by (3.17) it follows that

$$\sum_{v\in V_1} b^1_i(v)x^{S_1}(v) + x^{S_1}(w_1) \geq \beta^1_{j_1},$$
$$\sum_{v\in V_2} b^2_i(v)x^{S_2}(v) \geq \beta^2_{j_2}.$$

Since $-x^{S_1}(w_1) \geq -1$ and $x(u) = 0$, this implies that

$$\sum_{v \in V_1} b_i^1(v) x^{S_1}(v) + \sum_{v \in V_2} b_i^2(v) x^{S_2}(v) \geq \beta_{j_1}^1 + \beta_{j_2}^2 - 1$$

■

Lemma 3.23 *All integer points of $Q(G,T)$ belong to $P(G,T)$.*

Proof. We need to prove that all integer points of $Q(G,T)$ satisfy all terminal path inequalities induced by the terminal paths of G. Let us assume that, on the contrary, there exists an integer point $x^S \in Q(G,T)$ that does not belong to $P(G,T)$. Clearly, x^S satisfies all terminal path inequalities of $P(\widetilde{G}_i, \widetilde{T}_i)$ in which $x(w_i)$ is not involved. As x^S does not belong to $P(G,T)$ there must exist a terminal path $P_{t_1t_2}$ in G with one end $t_1 \in T_1$ and the other $t_2 \in T_2$ not intersecting S. Let $P_{t_iq_i} = (P_{t_1t_2} \cap V_i) \cup \{w_i, q_i\}$ be the terminal path in $\widetilde{G}_i$ between t_i and q_i. From the mixed inequality associated with the terminal path inequalities associated with $P_{t_1q_1}$ and $P_{t_2q_2}$ we have that

$$\sum_{v \in P_{t_1q_1} \setminus \{w_1\}} x(v) + \sum_{v \in P_{t_2q_2} \setminus \{w_2\}} x(v) - x(u) \quad \geq 1$$

which is violated by x^S. This contradicts the fact that $x^S \in Q(G,T)$. ■

Lemma 3.24 *Inequality*

$$\sum_{v \in V_i} b_j^i(v) x(v) - x(u) \geq \beta_j^i - 1$$

is valid for the polytope given by inequalities (3.28) − (3.32) *and it is dominated by inequality $\overline{a}x \geq \overline{\alpha}$ that is obtained as a linear combination of inequalities* (3.28) − (3.29) *and inequalities* (3.31) *associated with vertices of V_i.*

Proof. From Lemma 3.20, inequality

$$\sum_{v \in V_i} b_j^i(v) x(v) - x(u) \geq \beta_j^i - 1$$

is valid for $P(\widetilde{G}_i, \widetilde{T}_i)$. As the coefficient of w_i is 0, from Theorem 3.21, this inequality is dominated by an inequality $\overline{a}x \geq \overline{\alpha}$ that is a combination of inequalities (3.15), (3.16) and inequalities (3.18) associated with vertices of V_i. These inequalities are in (3.28) − (3.32). Then, $\overline{a}x \geq \overline{\alpha}$ is also valid for the polytope given by inequalities (3.28) − (3.32). ■

Theorem 3.25 *For a graph $G = (V \cup T, E)$, polytope $P(G,T)$ is completely described by inequalities* (3.28) – (3.32).

Proof. Let us assume that on the contrary, $Q(G,T)$ has a fractional extreme point x. We will distinguish two cases.
Case 1: $x(u) = 1$. Let $x_1 \in [0,1]^{\widetilde{V}_1}$ and $x_2 \in [0,1]^{\widetilde{V}_2}$ be the two solutions defined as follows.

- $x_i(v) = x(v)$ for all $v \in V_i$,
- $x_i(w_i) = 0$.

As $x(u) = 1$, from Lemma 3.19, $x_1 \in P(\widetilde{G}_1, \widetilde{T}_1)$ and $x_2 \in P(\widetilde{G}_2, \widetilde{T}_2)$. Thus, there exist $\lambda \in [0,1]^s, \mu \in [0,1]^p$, extreme points y_i of $P(\widetilde{G}_1, \widetilde{T}_1)$ and z_j of $P(\widetilde{G}_2, \widetilde{T}_2)$ such that

- $x_1 = \sum_{i=0}^{s} \lambda_i y_i$,
- $x_2 = \sum_{j=0}^{p} \mu_j z_j$,
- $\lambda_i \geq 0$, for all $i \in \{1, \ldots, s\}$ and $\sum_{i=1}^{s} \lambda_i = 1$,
- $\mu_j \geq 0$, for all $j \in \{1, \ldots, p\}$ and $\sum_{j=1}^{p} \mu_j = 1$.

As $x_1 \neq 0 \neq x_2$, there must exist $i_0 \in \{1, \ldots, s\}$ and $j_0 \in \{1, \ldots, p\}$ such that, $\lambda_{i_0} > 0$ and $\mu_{j_0} > 0$. Also note that each non mixed inequality tight for x is tight for either x_1 or x_2. Since y_{i_0} and z_{j_0} are extreme points, each facet of $P(\widetilde{G}_1, \widetilde{T}_1)$ (resp. $P(\widetilde{G}_2, \widetilde{T}_2)$) that contains x_1 (resp. x_2) also contains y_{i_0} (resp. z_{j_0}). Then, it follows that each inequality tight for x_1 (x_2) is also tight for y_{i_0} (z_{j_0}). As $x(u) = 1$, for all y_i and z_j, $x(u) = y_i(u) = z_j(u)$. Let $x^* \in [0,1]^V$ be the solution defined as follows

$$x^*(v) = \begin{cases} y_{i_0}(v), & \text{for all } v \in V_1, \\ z_{j_0}(v), & \text{for all } v \in V_2 \setminus \{u\}. \end{cases}$$

Thus, each non mixed inequality tight for x is tight for x^*. Now, we need to prove that each mixed inequality tight for x is also tight for x^*.

For two inequalities corresponding to $j_1 \in J_1$ and $j_2 \in J_2$, consider the following mixed equality tight for x.

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) + \sum_{v \in V_2} b^2_{j_2}(v)x(v) - x(u) = \beta^1_{j_1} + \beta^2_{j_2} - 1 \tag{3.33}$$

As $x_i \in P(\widetilde{G}_i, \widetilde{T}_i)$, we have that

$$\sum_{v \in V_1} b^1_{j_1}(v)x_1(v) + x_1(w_1) \geq \beta^1_{j_1},$$
$$\sum_{v \in V_2} b^1_{j_2}(v)x_2(v) + x_2(w_2) \geq \beta^2_{j_2}.$$

Thus, from Lemma 3.20, inequalities

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) - x(u) \geq \beta^1_{j_1} - 1,$$
$$\sum_{v \in V_2} b^1_{j_2}(v)x(v) - x(u) \geq \beta^2_{j_2} - 1.$$

are valid for $P(G, T)$. Since $x(u) = 1$, it follows that

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) \geq \beta^1_{j_1},$$
$$\sum_{v \in V_2} b^1_{j_2}(v)x(v) \geq \beta^2_{j_2}.$$

By (3.33), we obtain that these inequalities are tight for x, that is

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) = \sum_{v \in V_1} b^1_{j_1}(v)x_1(v) = \beta^1_{j_1},$$
$$\sum_{v \in V_2} b^2_{j_2}(v)x(v) = \sum_{v \in V_2} b^2_{j_2}(v)x_2(v) = \beta^2_{j_2}.$$

Therefore

$$\sum_{v \in V_1} b^1_{j_1}(v)y_{i_0}(v) = \beta^1_{j_1},$$
$$\sum_{v \in V_2} b^2_{j_2}(v)z_{j_0}(v) = \beta^2_{j_2},$$

Implying that

$$\sum_{v \in V_1} b^1_{j_1}(v)y_{i_0}(v) + \sum_{v \in V_2} b^2_{j_2}(v)z_{j_0}(v) = \beta^1_{j_1} + \beta^2_{j_2}.$$

Hence

$$\sum_{v\in V_1} b^1_{j_1}(v)x^*(v) + \sum_{v\in V_2} b^2_{j_2}(v)x^*(v) - x^*(u) = \beta^1_{j_1} + \beta^2_{j_2} - 1.$$

Consequently, each inequality tight for x is also tight for x^*. Since $x \neq x^*$, this contradicts the extremality of x.

Case 2: $x(u) < 1$.
We distinguish two cases, when there is no mixed inequality tight for x and when there is at least one mixed inequality tight for x.

Case 2.1: There is no mixed inequality tight for x.
Let $x_1 \in [0,1]^{\widetilde{V}_1}$ and $x_2 \in [0,1]^{\widetilde{V}_2}$ be two vectors defined as follows.

$$x_i(v) = \begin{cases} x(v), & \text{for all } v \in V_i, \\ 1, & \text{for } v = w_i. \end{cases}$$

All terminal path inequalities associated with terminal paths between two terminals of T_i, are satisfied by x. Thus, they are satisfied by x_i. As $x_i(w_i) = 1$, all terminal path inequalities associated with terminal paths between q_i and terminals of T_i are also satisfied by x_i. Thus, $x_1 \in P(\widetilde{G}_1, \widetilde{T}_1)$ and $x_2 \in P(\widetilde{G}_2, \widetilde{T}_2)$. Consequently, there exist $\nu \in [0,1]^s, \mu \in [0,1]^p$ and extreme points $y_j \in P(\widetilde{G}_1, \widetilde{T}_1)$ and $z_j \in P(\widetilde{G}_2, \widetilde{T}_2)$ such that

- $x_1 = \sum_{j}^{s} \nu_j y_j$,
- $x_2 = \sum_{j}^{p} \mu_j z_j$,
- $\nu_j \geq 0$, for all $j \in \{1, \ldots, s\}$ and $\sum_{j=1}^{s} \nu_j = 1$,
- $\mu_j \geq 0$, for all $\forall j \in \{1, \ldots, p\}$ and $\sum_{j=1}^{p} \mu_j = 1$.

Since $x(u) < 1$, there must exist $i_0 \in \{1, \ldots, s\}$ and $j_0 \in \{1, \ldots, p\}$ such that, $y_{i_0}(u) = z_{j_0}(u) = 0$ and $\nu_{i_0} > 0$ and $\mu_{j_0} > 0$. Each non-mixed inequality tight for x is tight for x_1 or x_2. As y_{i_0} and z_{j_0} are extreme points of $P(\widetilde{G}_1, \widetilde{T}_1)$ and $P(\widetilde{G}_2, \widetilde{T}_2)$, respectively, this implies that each facet of $P(\widetilde{G}_1, \widetilde{T}_1)$ (resp. $P(\widetilde{G}_2, \widetilde{T}_2)$) that contains x_1 (resp. x_2) also contains y_1 (resp. z_1). Let $x^* \in [0,1]^V$ be the solution given by

$$x^*(v) = \begin{cases} y_{i_0}(v), & \text{for all } v \in V_1, \\ z_{j_0}(v), & \text{for all } v \in V_2 \setminus \{u\}. \end{cases}$$

It follows that each inequality tight for x_1 and x_2 is also tight for y_{i_0} and z_{j_0}. Thus, each non-mixed inequality tight for x is tight for x^*. As $x \neq x^*$, we have a contradiction with the fact that x is an extreme point of the polytope given by (3.28) – (3.32).

Case 2.2: There is at least one mixed inequality tight for x.
Consider a mixed inequality for some $j_1 \in J_1$ and $j_2 \in J_2$, tight for x,

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) + \sum_{v \in V_2} b^2_{j_2}(v)x(v) - x(u) = \beta^1_{j_1} + \beta^2_{j_2} - 1. \tag{3.34}$$

Lemma 3.26 *There exist* $0 \leq \lambda \leq 1$ *and* $0 < \epsilon \leq 1$ *such that*

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) = \beta^1_{j_1} - 1 + \lambda, \tag{3.35}$$

$$\sum_{v \in V_2} b^2_{j_2}(v)x(v) = \beta^2_{j_2} - \lambda + \epsilon. \tag{3.36}$$

Proof. From Lemma 3.24 the following inequality

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) - x(u) \geq \beta^1_{j_1} - 1$$

is satisfied by x. Then by (3.34) we obtain

$$\sum_{v \in V_2} b^2_{j_2}(v)x(v) \leq \beta^2_{j_2}.$$

From Lemma 3.24, we also have

$$\beta^2_{j_2} - 1 \leq \sum_{v \in V_2} b^2_{j_2}(v)x(v) - x(u) \leq \beta^2_{j_2}.$$

Therefore, there exists $1 \geq \lambda \geq 0$ such that

$$\sum_{v \in V_2} b^2_{j_2}(v)x(v) - x(u) + \lambda = \beta^2_{j_2}.$$

Let $\epsilon = x(u)$, thus

$$\sum_{v \in V_2} b^2_{j_2}(v)x(v) = \beta^2_{j_2} - \lambda + \epsilon.$$

From equality (3.34) we then obtain

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) = \beta^1_{j_1} - 1 + \lambda,$$

and the lemma follows. ■

Now consider solutions $x_1 \in [0,1]^{\widetilde{V}_1}$ and $x_2 \in [0,1]^{\widetilde{V}_2}$ such that

$$x_1(v) = \begin{cases} x(v), & \text{for all } v \in V_1, \\ 1 - \lambda, & \text{for } v = w_1. \end{cases}$$

and

$$x_2(v) = \begin{cases} x(v), & \text{for all } v \in V_2, \\ \lambda - \epsilon, & \text{for } v = w_2. \end{cases}$$

we have the following lemma.

Lemma 3.27 *$x_i \in P(\widetilde{G}_i, \widetilde{T}_i)$, for $i = 1, 2$.*

Proof. We consider two cases

- $x_1 \in P(\widetilde{G}_1, \widetilde{T}_1)$.
  It is clear that x_1 satisfies inequalities (3.15)-(3.16). We will prove that it also satisfies inequalities (3.17). Assume, on the contrary, that for some $j \in J^1$,

  $$\sum_{v \in V_1} b^1_{j_3}(v)x_1(v) + x_1(w_1) < \beta^1_{j_3}.$$

  From Lemma 3.26, it then follows that

  $$\sum_{v \in V_1} b^1_{j_3}(v)x_1(v) + x_1(w_1) + \sum_{v \in V_2} b^2_{j_2}(v)x(v) < \beta^1_{j_3} + \beta^2_{j_2} - \lambda + \epsilon.$$

  Since $x_1(w_1) = 1 - \lambda$ and $x(u) = \epsilon$, this yields

  $$\sum_{v \in V_1} b^1_{j_3}(v)x(v) + \sum_{v \in V_2} b^2_{j_2}(v)x(v) - x(u) < \beta^1_{j_3} + \beta^2_{j_2} - 1,$$

  a contradiction with $x \in Q(G, T)$.

- $x_2 \in P(\widetilde{G}_2, \widetilde{T}_2)$.
  It is clear that x_2 satisfies inequalities (3.15)-(3.16). We will prove that it also satisfies inequalities (3.17). To this end we will show that all inequalities of J^2 are satisfied by x_2. Consider an inequality $j_4 \in J^2$

$$\sum_{v \in V_2} b^2_{j_4}(v) x_2(v) + x_2(w_2) \geq \beta^2_{j_4}.$$

  And let us assume that it is violated by x_2, that is

$$\sum_{v \in V_2} b^2_{j_4}(v) x_2(v) + x_2(w_2) < \beta^2_{j_4}.$$

  From Lemma (3.26)

$$\sum_{v \in V_2} b^2_{j_4}(v) x_2(v) + x_2(w_2) + \sum_{v \in V_1} b^1_{j_1}(v) x(v) < \beta^1_{j_4} + \beta^1_{j_1} - 1 + \lambda.$$

  As $x_2(w_2) = \lambda - \epsilon$, this yields

$$\sum_{v \in V_2} b^2_{j_4}(v) x(v) + \sum_{v \in V_1} b^1_{j_1}(v) x(v) - \epsilon < \beta^1_{j_4} + \beta^1_{j_1} - 1.$$

  Therefore, the mixed inequality associated with j_4 and j_1 is violated by x, a contradiction.

Thus, our lemma holds. ■

Since $x_1 \in P(\widetilde{G}_1, \widetilde{T}_1)$ and $x_2 \in P(\widetilde{G}_2, \widetilde{T}_2)$, there exist $\lambda \in [0,1]^s, \mu \in [0,1]^p$ and extreme points $y_j \in P(\widetilde{G}_1, \widetilde{T}_1)$ and $z_j \in P(\widetilde{G}_2, \widetilde{T}_2)$ such that

- $x_1 = \sum_j^s \lambda_j y_j$
- $x_2 = \sum_j^p \mu_j z_j$
- $\lambda_j > 0$, for all $i \in \{1, \ldots, s\}$
- $\mu_j > 0$, for all $\forall j \in \{1, \ldots, p\}$

Since $x_1(u) < 1$ and $x_2(u) < 1$, there must exist y_{i_0} and z_{j_0} such that $y_{i_0}(u) = z_{j_0}(u) = 0$.
Let $x^* \in [0,1]^V$ defined as

$$x^*(v) = \begin{cases} y_{i_0}(v), & \text{for all } v \in V_1, \\ z_{j_0}(v), & \text{for all } v \in V_2 \setminus \{u\}. \end{cases}$$

Each non mixed inequality tight for x is tight for x_i. Also each tight inequality tight for x_i is tight for x^*. It then follows that each non-mixed inequality tight for x is tight for x^*.

Lemma 3.28 *Each mixed inequality tight for x is tight for x^*.*

Proof. Consider a mixed inequality tight for x, corresponding to some $j' \in J_1$ and $j'' \in J_2$,

$$\sum_{v \in V_1} b^1_{j'}(v)x(v) + \sum_{v \in V_2} b^2_{j''}(v)x(v) - x(u) = \beta^1_{j'} + \beta^2_{j''} - 1. \tag{3.37}$$

By Lemma (3.26) we have that

$$\sum_{v \in V_1} b^1_{j'}(v)x(v) = \beta^1_{j'} - 1 + \lambda'', \tag{3.38}$$

$$\sum_{v \in V_2} b^2_{j''}(v)x(v) = \beta^2_{j''} - \lambda'' + \epsilon. \tag{3.39}$$

It is easy to see that $\lambda'' \geq \lambda$. Otherwise, by summing inequality(3.38) and (3.36) we obtain an inequality violated by x. In the following claim we will prove that indeed $\lambda'' = \lambda$.

Claim 1.

$$\sum_{v \in V_1} b^1_{j'}(v)x(v) = \beta^1_{j'} - 1 + \lambda,$$

$$\sum_{v \in V_2} b^2_{j''}(v)x(v) = \beta^2_{j''} - \lambda + \epsilon.$$

Proof. If $j' = j_1$ and $j'' = j_2$, then the claim holds. Therefore, let us assume that $j' \neq j_1$ or $j'' \neq j_2$ and $\lambda'' > \lambda$. From equality (3.38), it then follows that

$$-\sum_{v \in V_1} b^1_{j'}(v)x(v) < -\beta^1_{j'} + 1 - \lambda.$$

By summing the above inequality together with equality (3.37), we get

$$\sum_{v \in V_2} b^2_{j''}(v)x(v) < \beta^2_{j''} + \epsilon - \lambda.$$

And by summing the above inequality together with (3.35), we obtain

$$\sum_{v \in V_1} b^1_{j_1}(v)x(v) + \sum_{v \in V_2} b^2_{j''}(v)x(v) - x(u) < \beta^1_{j_1} + \beta^2_{j''} - 1.$$

a contradiction. It then follows that $\lambda'' = \lambda$. □

Claim 2. Inequalities j' and j'' are tight for x_1 and x_2, respectively.

Proof. We have that

$$\sum_{v \in V_1} b^1_{j'}(v)x(v) = \sum_{v \in V_1} b^1_{j'}(v)x_1(v) = \beta^1_{j'} + \lambda - 1,$$
$$\sum_{v \in V_2} b^2_{j''}(v)x(v) = \sum_{v \in V_2} b^2_{j''}(v)x_2(v) = \beta^2_{j''} - \lambda + \epsilon.$$

As $x_1(w_1) = 1 - \lambda$ and $x_2(w_2) = \lambda - \epsilon$, it then follows that

$$\sum_{v \in V_1} b^1_{j'}(v)x_1(v) + x_1(w_1) = \beta^1_{j'} \tag{3.40}$$
$$\sum_{v \in V_2} b^2_{j''}(v)x_2(v) + x_2(w_2) = \beta^2_{j''} \tag{3.41}$$

Therefore, the statement holds. □

Now summing equalities (3.40) and (3.41), yields

$$\sum_{v \in V_1} b^1_{j'}(v)x_1(v) + \sum_{v \in V_2} b^2_{j''}(v)x_2(v) + x_2(w_2) + x_1(w_1) = \beta^1_{j'} + \beta^2_{j''}.$$

Since $x_1(w_1) = 1 - \lambda$ and $x_2(w_2) = \lambda - \epsilon$, we obtain

$$\sum_{v \in V_1} b^1_{j'}(v)x^*(v) + \sum_{v \in V_2} b^2_{j''}(v)x^*(v) - x^*(u) = \beta^1_{j'} + \beta^2_{j''} - 1.$$

Hence, any inequality tight for x is also tight for x^*. This ends the proof of the lemma.

From Lemma 3.28, any inequality tight for x is also tight for x^*. This contradicts the fact that x an extreme point of $Q(G,T)$, and the proof of our theorem is complete. ■

Corollary 3.29 *For terminal trees, the multi-terminal vertex separator polytope is given by terminal tree inequalities (2.18) and the trivial inequalities.*

Proof. Each vertex in a terminal tree is a vertex cut. It follows that a terminal tree G can be decomposed into $\widetilde{G}_1$ and $\widetilde{G}_2$. Moreover, $\widetilde{G}_1$ and $\widetilde{G}_2$ are also terminal trees and can then be decomposed. Thus a terminal tree can be decomposed into several star trees. Moreover, a star tree is a terminal tree with at most one vertex of degree greater or equal to 3 and the terminal tree inequality is the general form of the star tree inequality. Thus the mixed inequality associated with two terminal tree inequalities, is a terminal tree inequality. By Theorem 3.3 for star trees, the MTVS polytope is given by inequalities (3.1) and the trivial inequalities. Consequently, by Theorem 3.25, for terminal trees the multi-terminal vertex separator polytope is given by terminal tree inequalities (2.18) and the trivial inequalities. ■

3.2.3 Facet composition

In this section we will discuss composition of facets for the multi-terminal vertex separator polytope. In particular we will show that a mixed inequalities (3.30) defines a facet of $P(G,T)$ if the two inequalities to which it is related define facets in the corresponding graphs.

Let $G = (V \cup T, E)$ be a graph, and H a subgraph of G.

Lemma 3.30 *If $ax \geq \alpha$ is valid for $P(H, T(H))$, then it is valid for $P(G,T)$.*

Proof. Let us assume the contrary. Suppose that $ax \geq \alpha$ is valid for $P(H, T(H))$ and that there exists a graph G with a subgraph H such that $ax \geq \alpha$ is not valid for $P(G,T)$. It follows that there exists a solution $x_0 \in P(G,T)$ that violates $ax \geq \alpha$. Let $\overline{x}$ be the restriction of x_0 on $V(H)$. Clearly, $\overline{x} \in P(H, T(H))$ and $\overline{x}$ violates $ax \geq \alpha$. This is a contradiction with the fact that $ax \geq \alpha$ is valid for $P(H, T(H))$. ■

Theorem 3.31 *If an inequality (3.17) defines a facet for $P(\widetilde{G}_1, \widetilde{T}_1)$ and an inequality (3.17) defines a facet for $P(\widetilde{G}_2, \widetilde{T}_2)$, then the associated mixed inequality (3.30) defines a facet for $P(G,T)$.*

Proof. Since inequality (3.17) defines a facet for $P(\widetilde{G}_i, \widetilde{T}_i)$, let $\mathcal{S}_i$ a set of $|\widetilde{V}_i|$ affinely independent solutions satisfying inequality (3.14) with equality, $i = 1, 2$. From Lemma 3.18, there is no solution in $\mathcal{S}_i$ containing w_i and u together, for $i = 1, 2$. Let $n_1^i, n_2^i, n_3^i \in \mathbb{N}$ defined as follows

- n_1^i : the number of solutions in $\mathcal{S}_i$ containing u.
- n_2^i : the number of solutions in $\mathcal{S}_i$ containing w_i.
- n_3^i : the number of solutions in $\mathcal{S}_i$ not containing neither u nor w_i.

Clearly, $n_1^i + n_2^i + n_3^i = |\widetilde{V}_i|$, for $i = 1, 2$.

We need to construct $|V|$ solutions affinely independent satisfying (3.30) with equality. Since w_1 and w_2 do no appear in G and u appears once, we only need $|\widetilde{V}_1| + |\widetilde{V}_2| - 3$ such solutions. Consider a solution $\overline{S}_1 \in \mathcal{S}_1$ containing u. For each solution $S_2 \in \mathcal{S}_2$ containing u, we construct a new solution $(\overline{S}_1 \cup S_2)$. Let A_1 be the set of all these solutions. Now consider a solution $\overline{S}_2 \in \mathcal{S}_2$ containing u. For each solution $S_1 \in \mathcal{S}_1$ containing u, we construct a new solution $(S_1 \cup \overline{S}_2)$. Let A_2 be the set of all these solutions. It can be easily seen that there is one solution in A_1 that is also in A_2. Thus, $A = A_1 \cup A_2$ contains $n_1^1 + n_1^2 - 1$ affinely independent solutions. Consider a solution $\widehat{S}_1 \in \mathcal{S}_1$ not neither u, nor w_1. For each solution $S_2 \in \mathcal{S}_2$ containing w_2, we construct a new solution $(\widehat{S}_1 \cup S_2)$. Let B_1 be the set of all these solutions. Consider a solution $\widehat{S}_2 \in \mathcal{S}_2$ not containing neither u, nor w_1. For each solution $S_1 \in \mathcal{S}_1$ containing w_1, we construct a new solution $(S_1 \cup \widehat{S}_2)$. Let B_2 be the set of all these solutions. Thus, $B = B_1 \cup B_2$ contains $n_2^1 + n_2^2$ affinely independent solutions. Consider a solution $\overline{\overline{S}}_1 \in \mathcal{S}_1$ containing w_1. For each solution $S_2 \in \mathcal{S}_2$ not containing neither u, nor w_2, we construct a new solution $(\overline{\overline{S}}_1 \cup S_2)$. Let C_1 be the set of all these solutions. Consider a solution $\overline{\overline{S}}_2 \in \mathcal{S}_2$ containing w_2. For each solution $S_1 \in \mathcal{S}_1$ not containing neither u, nor w_1, we construct a new solution $(S_1 \cup \overline{\overline{S}}_2)$. Let C_2 be the set of all these solutions. Clearly, there is one solution in C_2 that is also in B_1 and one solution in C_2 that is in B_2. Thus, $C = (C_1 \cup C_2) \setminus (B_1 \cup B_2)$ contains $n_3^1 + n_3^2 - 2$ affinely independent solutions.

Therefore, $A \cup B \cup C$ represents a set of $|\widetilde{V}_1| + |\widetilde{V}_2| - 3$ affinely independent solutions satisfying inequality (3.30) with equality. ■

Now we present two valid inequalities, that are the generalization of the clique star and the terminal cycle inequalities (see Section 2.3).

3.2.3.0.1 General clique star inequality

A general clique star $H = (V' \cup T', E)$ is obtained by the composition of one clique star and several star trees. Thus, H contains a clique $K_f \subseteq V'$ on f vertices and has q terminals. Figure 3.8 displays a general clique star of 14 terminals.

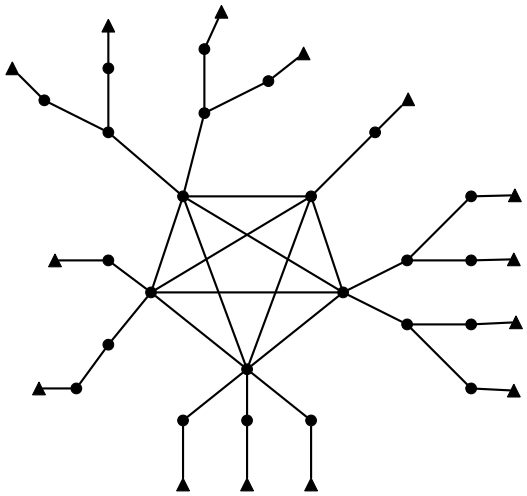


Figure 3.8: General clique star

Proposition 3.32 *For a general clique star $H = (V' \cup T', E)$ subgraph of $G = (V \cup T, E)$, the following inequality*

$$\sum_{v \in K_f} (d_H(v) - (f-1))x(v) + \sum_{v \in V' \setminus K_f} (d_H(v) - 1)x(v) \geq q - 1$$

is valid for $P(G,T)$, where $d_H(v)$ is the degree of v in H.

3.2.3.0.2 General terminal cycle inequality

A general terminal cycle $H = (V' \cup T', E)$ is obtained by the composition of one terminal cycle and several star trees. Thus H contains one cycle $C_f \subseteq V'$ on f vertices and has q terminals. Let $C' \subset C_f$ be the subset of vertices in C_f of degree greater than or equal to 3 in H. Figure 3.9 shows a general terminal cycle of 14 terminals.

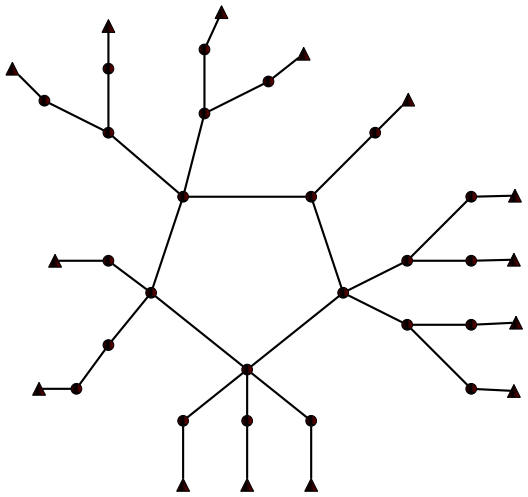


Figure 3.9: General terminal cycle

Proposition 3.33 *For a general terminal cycle* $H = (V' \cup T', E)$ *subgraph of* $G = (V \cup T, E)$*, the following inequality*

$$\sum_{v \in C'} (d_H(v) - 2)x(v) + \sum_{v \in V' \setminus C'} (d_H(v) - 1)x(v) \geq q - \lfloor \frac{|C'|}{2} \rfloor$$

is valid for $P(G, T)$*, where* $d_H(v)$ *is the degree of* v *in* H*.*

3.2.4 Algorithmic aspect

The polyhedral and the algorithmic aspects for combinatorial optimization problems are very related. In this section we will analyze algorithmic consequences of the decomposition studied before. We will show that if G is the 1-sum of two graphs G_1 and G_2, then solving the multi-terminal vertex separator problem in G can reduce to solving it in graph $\widetilde{G}_2$ with an appropriate weight system.

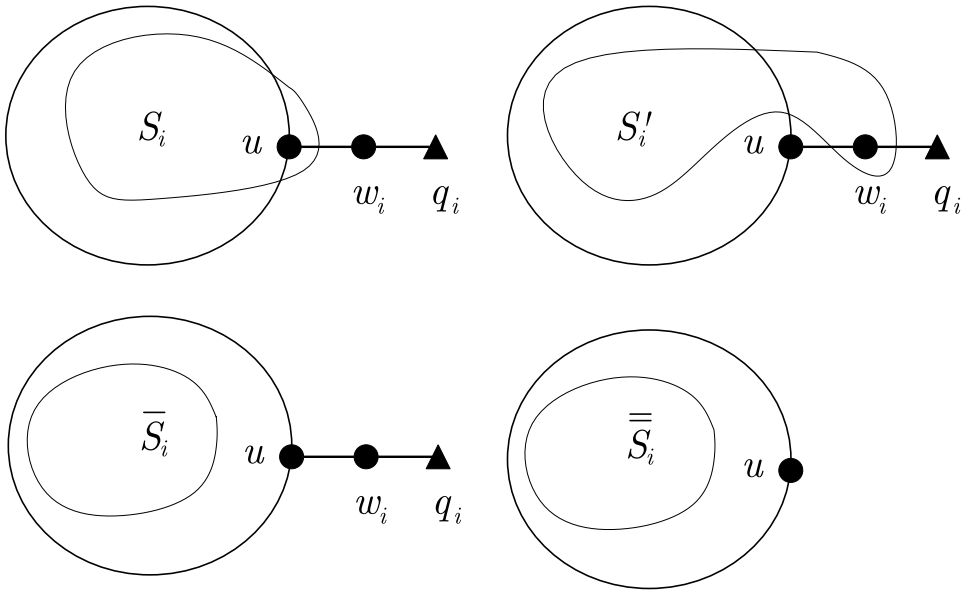


Figure 3.10: Illustration of separators S_i, S'_i, $\overline{S}_i$ and $\overline{\overline{S}}_i$.

Given a weight vector $c : V \to \mathbb{R}$ and let $c_i \in \mathbb{R}^{V_i}$ be the restriction of c on V_i, for $i = 1, 2$. Let $c'_i \in \mathbb{R}^{\widetilde{V}_i}$ be a weight vector obtained from c_i by setting $c'_i(w_i) = 0$.

Let $S_i, S'_i, \overline{S}_i$ and $\overline{\overline{S}}_i$ be four subsets of vertices, such that

- S_i is the MTVS separator of $\widetilde{G}_i$ containing u but not w_i.
- S'_i is the MTVS separator of $\widetilde{G}_i$ containing w_i but not u.
- $\overline{S}_i$ is the MTVS separator of $\widetilde{G}_i$ not containing neither u nor w_i.
- $\overline{\overline{S}}_i$ is the MTVS separator of G_i not containing u.

Let $s_i, s'_i, \overline{s}_i, \overline{\overline{s}}_i \in \mathbb{N}$ (resp. $\omega_i, \omega'_i, \overline{\omega}_i, \overline{\overline{\omega}}_i \in \mathbb{N}$) be the size (resp. the weight) of $S_i, S'_i, \overline{S}_i, \overline{\overline{S}}_i$, respectively.

These separators are illustrated in Figure 3.10. It is important to precise that the same separators computed before are used later. To force a vertex to be in the separator we can delete the vertex from the graph. To force a vertex v to be out of the separator we set its weight to $+\infty$.

Let $\overline{c}_2 \in \mathbb{R}^{\widetilde{V}_2}$ be a vector associated with $\widetilde{V}_2$ such that

- $\overline{c}_2(u) = \omega_1 - s_2\frac{\overline{\overline{\omega}}_1}{\overline{s}_2}$,
- $\overline{c}_2(w_2) = \overline{\omega}_1 - s'_2\frac{\overline{\overline{\omega}}_1}{\overline{s}_2}$,
- $\overline{c}_2(v) = c(v) + \frac{\overline{\overline{\omega}}_1}{\overline{s}_2} \quad \forall v \in \widetilde{V}_2 \setminus \{u, w_i\}$,

In $\widetilde{G}_2$ w.r.t $\overline{c}_2$, let $S_2^* \in \{S_2, S'_2, \overline{S}_2\}$, already computed, having a minimum weight ω_2^*. Clearly, since we add the same value $\frac{\overline{\overline{\omega}}_1}{\overline{s}_2}$ to the weight of each vertex $v \in V_2 \setminus \{u\}$, then the minimum separator in $\widetilde{G}_2$ under vertex weight vector $\overline{c}_2$ is either S_2, S'_2 or $\overline{S}_2$.

Let $S^* \subseteq V$ be the set giving by

$$S^* = \begin{cases} S_1 \cup S_2^* & \text{If } u \in S_2^* \text{ and } w_2 \notin S_2^* \quad \textit{i.e., } S_2^* = S_2 \\ (\overline{S}_1 \cup S_2^*) \setminus w_2 & \text{If } w_2 \in S_2^* \text{ and } u \notin S_2^* \quad \textit{i.e., } S_2^* = S'_2 \\ \overline{\overline{S}}_1 \cup S_2^* & \text{If } u, w_2 \notin S_2^* \quad \textit{i.e., } S_2^* = \overline{S}_2 \end{cases}$$

Theorem 3.34 *S^* is the minimum multi-terminal vertex separator in G, its weight is $\omega^* = \omega_2^*$.*

Proof. We distinguish three cases for the state of separator S^*

- S^* contains u. Thus $(S^* \cap V_i)$ is a separator in $\widetilde{G}_i$, for $i = 1, 2$.
- S^* does not contain u and $(S^* \cap V_1)$ intersects all terminal paths between q_1 and all terminal of T_1 in graph $\widetilde{G}_1$.
- S^* does not contain u and it may exist a terminal path between q_1 and a terminal of T_1 in graph $\widetilde{G}_1$ that does not intersect $(S^* \cap V_1)$.

Thus, the weight of S^* is

$$\omega^* = \min(\omega_1 + \omega_2 - c(u), \overline{\omega}_1 + \omega'_2 - c(w_2), \overline{\overline{\omega}}_1 + \overline{\omega}_2).$$

As

$$\omega^* = \min((\overline{c}_2(u) + s_2\frac{\overline{\overline{\omega}}_1}{\overline{s}_2}) + \omega_2 - c(u), (\overline{c}_2(w_2) + s'_2\frac{\overline{\overline{\omega}}_1}{\overline{s}_2}) + \omega'_2 - c(w_2), \overline{\overline{\omega}}_1 + \overline{\omega}_2).$$

We know that

- $s_2 \frac{\overline{\overline{\omega}}_1}{\overline{s}_2} + \omega_2 - c(u) = \omega_2^* - \overline{c}_2(u)$
- $s_2' \frac{\overline{\overline{\omega}}_1}{\overline{s}_2} + \omega_2' - c(w_2) = \omega_2^* - \overline{c}_2(w_2)$
- $\overline{c}_2(\overline{S}_2) = w_2^* = \overline{s}_2 \frac{\overline{\overline{\omega}}_1}{\overline{s}_2} + c(\overline{S}_2) = \overline{\overline{\omega}}_1 + c(\overline{S}_2) = \overline{\overline{\omega}}_1 + \overline{\omega}_2$

this implies that $\omega^* = \omega_2^*$. ■

Let us remark that if we can solve the MTVSP in polynomial time in $\widetilde{G}_1$ and $\widetilde{G}_2$, then we can solve the MTVSP in polynomial time in G. Indeed, as we can solve the MTVSP in polynomial time in $\widetilde{G}_1$, then we can compute all parameters $s_i, s_i', \overline{s}_i, \overline{\overline{s}}_i, \omega_i, \omega_i', \overline{\omega}_i, \overline{\overline{\omega}}_i$ and we can obtain sets $S_i, S_i', \overline{S}_i, \overline{\overline{S}}_i$ in polynomial time.

3.3 Composition of polyhedra by terminal-sum

Consider two graphs $G_1 = (V_1 \cup T_1, E_1)$ and $G_2 = (V_2 \cup T_2, E_2)$ and let $T_1' \subseteq T_1$ and $T_1' \subseteq T_2$ be two subsets of q terminals $T_1' = \{t_1^1, \ldots, t_q^1\}$ and $T_2' = \{t_1^2, \ldots, t_q^2\}$. $G = (V \cup T, E)$ is called a *terminal-sum* of $G_1 = (V_1 \cup T_1, E_1)$ and $G_2 = (V_2 \cup T_2, E_2)$ if it is obtained by merging each terminal $t_i^1 \in T_1$ with a terminal $t_i^2 \in T_2$, for all $i \in \{1, \ldots, q\}$. Figure 3.11 illustrates the graphs G, G_1 and G_2.

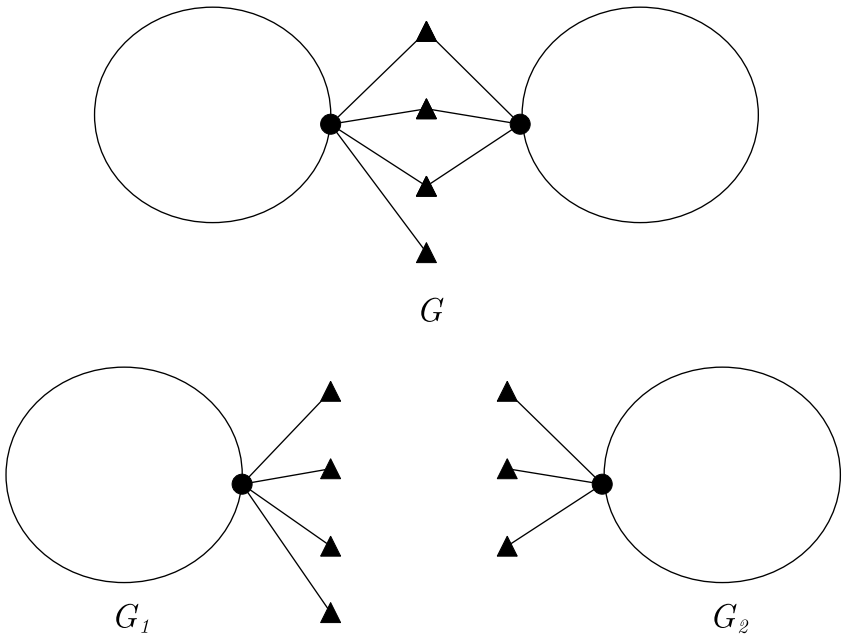


Figure 3.11: Terminal-sum where $q = 3$.

The multi-terminal vertex separator polytope for graph G_i can be expressed by the a system of inequalities A_i of the form

$$\begin{aligned} \sum_{v \in V_i} a_l^i(v) x(v) \geq \alpha_l^i \quad & \forall l \in L^i, \\ x(v) \leq 1 \quad & \forall v \in \widetilde{V}_i, \\ x(v) \geq 0 \quad & \forall v \in \widetilde{V}_i, \end{aligned} \tag{3.42}$$

where L^i is a set of inequalities, for $i = 1, 2$.

Theorem 3.35 *The multi-terminal vertex separator polytope for G is completely described by system of inequalities A obtained by the juxtaposition of systems A_1 and A_2.*

Proof. Let us assume the contrary, and let x^* be a fractional extreme point of the polytope given by the inequalities of A. Note that vector x^* satisfies a subset of inequalities $A^* \subseteq A$ with equality, where $|A^*| = |V_1| + |V_2|$. There is no equality in A^* containing two variables associated with two vertices, one in G_1 and the other in G_2. Therefore A^* can be partitioned into 2 subsets A_1^* and A_2^*, one associated with G_1 and the other one associated with G_2. Since A^* is linearly independent, it follows

that $|A_1^*| = |V_1|$ and $|A_2^*| = |V_2|$. Let x_i^* be the restriction of x^* on V_i. Hen x^* is an extreme point of $P(G_i, T_i)$. Moreover, since x^* is fractional, it follows that either x_1^* is a fractional or x_2^* is fractional, which is impossible. ■

Theorem 3.36 *If two linear systems A_1 and A_2 with no variable in common, are TDI, then the linear system given by the juxtaposition of A_1 and A_2 is also TDI.*

Proof. Let y_i^* be the integer optimal solution of the dual problem associated with A_i. Since the dual problem associated with G_i is TDI, y_i^* should exist. Thus, $y^* = [y_1^*, y_2^*]$ represents a feasible integer solution for the dual problem of the juxtaposition of A_1 and A_2. Moreover, it should be optimal. Otherwise, let $\overline{y}$ be the optimal solution of the dual problem of the juxtaposition of A_1 and A_2. Let $\overline{y}_i$ be the restriction of $\overline{y}$ on V_i. Clearly, $\overline{y}_i$ is a feasible solution for the dual problem of A_i. Thus, y_1^* or y_2^* is not optimal. Contradiction. ■

Let S_i^* be the optimal solution in G_i, for $i = 1, 2$. Then, we have the following corollary

Corollary 3.37 *Vertex separator $S^* = S_1^* \cup S_2^*$ is a minimum multi-terminal vertex separator in G.*

3.4 The minimum rooted-cycle cover problem

In this section, we introduce the *minimum rooted cycle cover problem* (MRCCP) which is a generalization of the multi-terminal vertex separator problem. This problem consists in, given a simple undirected graph $G = (V, E)$ and a root vertex r of G, deleting a node subset of $V \setminus \{r\}$ of a minimum weight so that root r is not contained in any cycle. In the case where r is adjacent to all vertices of $V \setminus \{r\}$, the minimum rooted cycle cover problem is equivalent to the minimum vertex-cover problem. Moreover, the minimum multi-terminal vertex separator problem is a particular case of the MRCC problem. Indeed, if for each vertex $v \in N(r)$ we add a terminal t adjacent to v, and by deleting r from G, then we obtain an instance of the minimum multi-terminal vertex separator problem. Any MTV separator in the new graph, intersects all cycles of G containing r.

Our main motivation to introduce the minimum rooted cycle cover problem is that it allows us to give short proofs for some min-max theorems, such results are fundamental

in combinatorial optimization and linear programming [107]. Jost and Naves gave such results for the minimum multi-terminal vertex separator problem in [95] (actually we independently obtained one result, namely Corollary 3.43, that is a consequence of one of their results).

Take a graph G and fix distinct vertices s, t. A *st-path of* G is a subset $P \subseteq V$ of vertices of G which can be ordered into a sequence $s = v_0, v_1, \ldots, v_k = t$ where $v_i v_{i+1}$ is an edge of G. The vertices v_0, v_k are the *extremities of* P, the other vertices are the *internal vertices of* P. Two st-paths P, Q are *internally vertex-disjoint* if $P \cap Q = \{s, t\}$. A subset D of vertices of G is a *st-vertex cut* if neither s nor t belongs to D, and D intersects every st-path. Let r be a vertex of G. A subset $C \subseteq V$ containing r and such that $C \setminus \{r\}$ is a path the extremities of which are adjacent with r is called a *rooted cycle of* (G, r). Two rooted cycles are *internally* vertex-disjoint if r is their only common vertex. A *rooted-cycle cover* of (G, r) is a subset of $V' \subseteq V \setminus \{r\}$ of non-root vertices so that $C \cap V' \neq \emptyset$ for all rooted cycle C. A rooted-cycle cover is minimum if $|V'|$ is minimum. A *rooted-cycle packing* of (G, r) is a collection $C_1, \ldots, C_k$ of rooted cycles so that $C_i \cap C_j = \{r\}$ for all distinct $i, j = 1, \ldots, k$. A rooted-cycle packing is maximum if k is maximum. Clearly the minimum of the cover is at least the maximum of the packing.

Let us recall two fundamental min-max theorems.

Given a graph, a *matching* is a subset of pairwise vertex-disjoint edges.

Kőnig's theorem. ([82]) *Let G be a bipartite graph. The minimum size of a vertex-cover of G is equal to the maximum size of a matching of G.*

Menger's theorem. ([93]) *Let G be a graph and let s and t be two terminals of G. A st-vertex cut of minimum size in G is equal to the maximum number of internally vertex-disjoint paths between s and t.*

3.4.1 Packing and covering rooted cycles

The following consequence of Menger's theorem is useful to characterize rooted graphs for which the minimum size of a subset of non-root vertices intersecting all rooted cycles is equal to the maximum number of internally vertex-disjoint rooted cycles.

Corollary 3.38 *Let G be a graph with a vertex t and a subset S, of at least k vertices, not containing t. If there are k internally vertex-disjoint vt-paths for every $v \in S$, then there are k distinct vertices $s_1, \ldots, s_k$ of S, with $s_i t$-paths P_i for each $i = 1, \ldots, k$, so that t is the only vertex belonging to all paths $P_1, \ldots, P_k$.*

Proof. Add a new vertex s to G and link it to every vertex in S, See Figure 3.12.

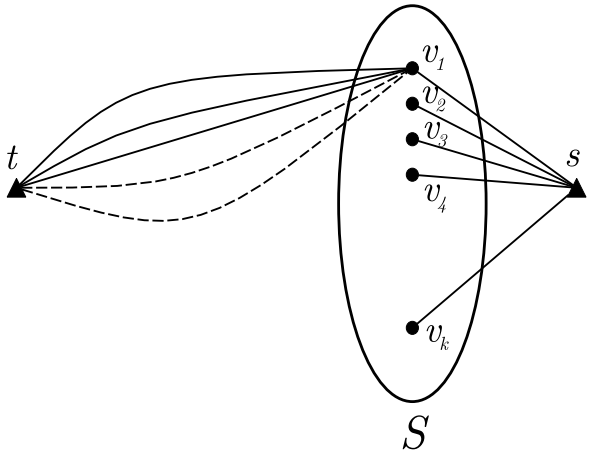


Figure 3.12: k disjoint st-paths

We only need to prove that there are k internally vertex-disjoint st-paths. If this is not the case, then, by Menger's theorem, there is a st-vertex cut D of size $|D| < k$. Let $v \in S \setminus D$. Clearly v is not adjacent to t. Thus D is a vt-vertex cut. But this is impossible since (again by Menger's theorem) there are k internally vertex-disjoint vt-paths. ■

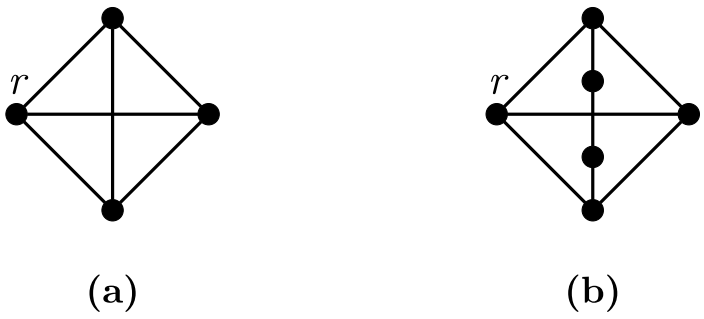


Figure 3.13: Rooted graph (K_4, r) and a subdivision of (K_4, r)

Let K_4 be the complete graph on four vertices and r one of its vertices. A rooted graph $(\hat{G}, r)$ is a subdivision of (K_4, r) if it is obtained from (K_4, r) by inserting vertices on the edges. Note that in any such subdivision, the vertex r has degree three. A *rooted partial subgraph of* (G, r) is a rooted graph (G', r) where G' is a partial subgraph of G. Figure 3.13.(a) displays a rooted graph (K_4, r) and Figure 3.13.(a) shows a subdivision of (K_4, r).

Theorem 3.39 *The minimum size of a subset of non-root vertices intersecting all rooted cycles is equal to the maximum number of internally vertex-disjoint rooted cycles, for all partial rooted subgraphs of* (G, r)*, if and only if no partial rooted subgraph of* (G, r) *is a subdivision of* (K_4, r)*.*

Proof. ($\Rightarrow$) It suffices to see that, for any subdivision of (K_4, r), any two rooted cycles must have a non-root vertex in common while any rooted cycle cover needs at least two non-root vertices.
($\Leftarrow$) Let (G, r) be a minimum graph, that is, a graph with a minimum number of edges, such that the minimum rooted cycle cover is strictly greater than the maximum packing of rooted cycles. Minimality implies that G has no vertices of degree < 3. It then follows that the graph $G \setminus \{r\}$ obtained from G by removing r has a cycle C with at least three distinct vertices s_1, s_2, s_3 (since G is a simple graph). Hence, by Corollary 3.38, it suffices to prove Claim 1 below, since this implies that there are three internally vertex-disjoint $s_i r$-paths which form with C a subdivision of (K_4, r).

Claim 1. There are three internally vertex-disjoint vr*-paths for every vertex* $v \in V \setminus \{r\}$*.*

Proof. Assume that the claim is not true, and let v be a vertex for which there is no three internally vertex-disjoint vr-paths. Remark that, for any vertex s nonadjacent to r, the minimality of G implies that no sr-vertex cut is a clique. In particular, any sr-vertex cut has at least two vertices (indeed otherwise v belongs to no rooted cycle). Furthermore, no vertex-cut has only one vertex. We build a graph $\hat{G}$ from G as follows:

(a): If v is nonadjacent with r, then (by Menger's theorem) there is a vr-vertex cut with size 2, say $D = \{u, w\}$. Let $V' \ni v$ be the subset of vertices in the component, containing v, of the graph obtained from G by removing D. We let $\hat{G}$ be the graph obtained by removing V' and adding the edge $e = uw$.

(b): If v is adjacent with r, then, in the graph $G \setminus vr$ (obtained from G by removing the edge vr), there is a vr-vertex cut $D = \{u\}$. Let $V' \ni v$ be the subset of vertices in

the component, containing v, of the graph obtained by removing u from $G \setminus vr$. We let $\hat{G}$ be the graph obtained by removing V' and adding the edge $e = ur$.

Now observe, that if there are ν vertex-disjoint rooted cycles in $\hat{G}$, then there are also ν vertex-disjoint rooted cycles in G. Indeed, if some rooted cycle of $\hat{G}$ contains the additional edge e, then e can be replaced by a path of G with all internal vertices in V'. Moreover, if there are τ vertices intersecting every rooted cycles of $\hat{G}$, then these τ vertices intersect also every rooted cycles of G. We have a contradiction, since the minimality of G implies $\tau = \nu$. □

And the proof of the theorem is ended. ■

Finally, since series-parallel graphs are those with no minor K_4, one has:

Remark 3.40 *Given a graph G, the minimum rooted cycle cover equals the maximum rooted cycle packing, for every choice of a root r, if and only if G is series-parallel.*

3.4.2 Pseudo-bipartite rooted graphs

Given a graph $G = (V, E)$ and a node r, the *closed neighborhood of* r is the set $N[r] = N(r) \cup \{r\}$.

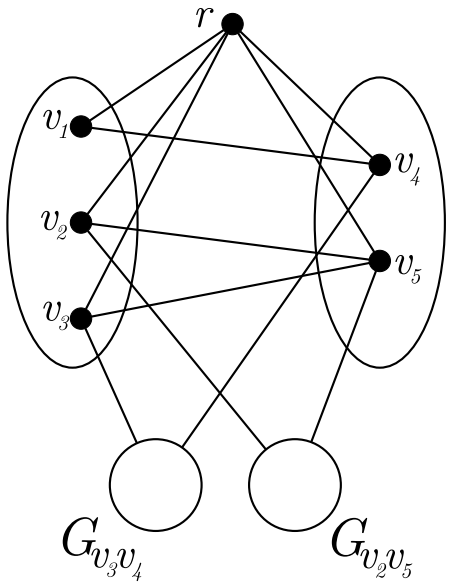


Figure 3.14: Pseudo-bipartite rooted graph

A rooted graph is *pseudo-bipartite* if it is obtained from a bipartite graph $(V_1, V_2; E)$ by creating a root vertex linked to every vertex of $V_1 \cup V_2$, and then, by replacing some original edges $uv \in E$ by any graph G_{uv} (on new vertices) with edges between u, or v, and some vertices of G_{uv}. Figure 3.14 illustrates a pseudo-bipartite rooted graph. More precisely:

Definition 1 *A rooted graph (G, r) is* pseudo-bipartite *if*

(a) The subgraph $G[N(r)]$ induced by the neighbors of r is a bipartite graph G_B,

(b) There is a bipartition of G_B, so that every component of the graph $G \setminus N[r]$, that we obtain if we remove the closed neighborhood of r, has at most one neighbor in each side of G_B.

Contracting a vertex v is to delete v, and to add edges so that its neighborhood $N(v)$ forms a clique. A *rooted minor of* (G, r) is a rooted graph $(\hat{G}, r)$ obtained from (G, r) by deleting vertices different from r, or by *contracting* vertices $v \in V \setminus N[r]$ outside the closed neighborhood of the root.

An *odd wheel* is a graph composed of an odd cycle together with one vertex, called the *center* of the wheel, to which all the vertices of the odd cycle are linked. A *rooted odd wheel* is a rooted graph (G, r) so that G is an odd wheel, the center of which is the root r.

Theorem 3.41 *A rooted graph (G, r) is pseudo-bipartite if and only if it has no rooted minor which is a rooted odd wheel.*

Proof. Necessity holds since $G[N(r)] = G_B$ has no odd cycle and since every edge which appears in $G[N(r)]$ by contracting some vertex outside $N[r]$ necessarily links two vertices in different side of the bipartition of G_B.

To see sufficiency, suppose that no rooted-minor of (G, r) is a rooted odd-wheel. Condition (a) of Definition 1 is indeed satisfied since deleting all vertices but those of an odd cycle in the neighborhood of r leaves a rooted odd wheel. Assume that condition (b) is not satisfied. Let $U_1, \ldots, U_p$ be the components of the graph obtained by removing r and all its neighbors. If U_i has at least three neighbors x, y, z (in G_B), then contracting all the vertices in U_i and deleting all vertices but x, y, z (and r) leaves (K_4, r) which is a rooted odd wheel. It follows that U_i has at most two neighbors. Chose a bipartition

V_1, V_2 of G_B so that the number of components $U_1, \ldots, U_p$ having two neighbors in the same side is minimum. There is a component U_i with two neighbors x, y in the same side, say $x, y \in V_1$ (otherwise we are done). Let V_1^x (resp. V_2^x) be the set of vertices in V_1 (resp. in V_2) reachable, from x, by a path of G_B. Similarly, define V_1^y and V_2^y. If either $V_1^x \cap V_1^y \neq \emptyset$ or $V_2^x \cap V_2^y \neq \emptyset$, then there exists a xy-path P in G_B. Yet contracting U_i and deleting all vertices but those of P (and r) leaves a rooted odd wheel. It follows that $V_1^x, V_2^x, V_1^y, V_2^y$ are pairwise disjoint, and hence $(V_1 \setminus V_1^x) \cup V_2^x, (V_2 \setminus V_2^x) \cup V_1^x$ is a possible bipartition for G_B. The way the bipartition V_1, V_2 was chosen implies that there is another component, say U_j, with either one neighbor in V_1^x and the other in V_2^y, or one neighbor in V_1^y and the other in V_2^x. Anyway, contracting both U_i and U_j creates an odd cycle in the neighborhood of r. Now deleting the vertices outside this odd cycle leaves a rooted odd wheel; contradiction. ■

Given a pseudo-bipartite rooted graph (G, r) with bipartite graph $G_B = G[N(v)]$, we let $\hat{G}$ be the graph obtained by removing r from (G, r), and then by creating two new vertices s and t, so that s is linked to every vertex of one side of G_B, and t is linked similarly to every vertex in the other side of G_B.

Remark 3.42 *A subset $P \subseteq V$ is a st-path of $\hat{G}$ if and only if $P \setminus \{s, t\} \cup \{r\}$ is a rooted cycle of (G, r).*

Since Remark 3.42 holds, Menger's theorem implies that, for pseudo-bipartite rooted graphs, the minimum rooted cycle cover equals the maximum rooted cycle packing.

Observe, moreover, that for any rooted odd wheel with an odd cycle having $2k + 1$ vertices, the minimum size of a cover is $k + 1$ while the maximum packing is k. It follows that Theorem 3.41 implies Corollary 3.43 below, which is also a consequence of a result in [95] (namely Lemma 9).

Corollary 3.43 ([95]) *The minimum size of a subset of non-root vertices intersecting all rooted cycles is equal to the maximum number of internally vertex-disjoint rooted cycles, for all rooted minor of (G, r), if and only if no rooted minor of (G, r) is a rooted odd wheel.*

Observe that, if every vertex is linked to r, then (G, r) is pseudo-bipartite if and only if the graph induced by $V \setminus \{r\}$ is bipartite. Hence by Theorem 3.41:

Remark 3.44 *Corollary 3.43 contains Kőnig's theorem as particular case.*

Note also that for recognizing if (G, r) is pseudo-bipartite or not, it suffices to contract all vertices outside the closed neighborhood of r, to remove r, and to check if the remaining graph is bipartite or not. So one has:

Remark 3.45 *It can be checked in polynomial time if (G, r) is pseudo-bipartite or not.*

3.5 Conclusion

In this chapter we have considered the multi-terminal vertex separator problem. We have characterized the MTVS polytope for two classes of graphs the star trees and the clique stars. We have given a linear system for each class of graphs, that is total dual integral. We have studied a composition (decomposition) technique for the multi-terminal vertex separator polytope in graphs that are decomposable by one-node cutsets. As a consequence, we have obtained a procedure to construct this polytope in graphs that are recursively decomposed. We have also studied polytope compositions by merging terminals, and have presented the algorithmic aspect resulting of these compositions. Finally we have introduced the minimum rooted cycle cover problem that is a generalization of the MTVSP. We have used Menger's theorem to provide a characterization of all rooted graphs such that the maximum number of vertex-disjoint rooted cycles equals the minimum size of a subset of non-root vertices intersecting all rooted cycles, for all subgraphs.

Chapter 4

The multi-terminal vertex separator problem : Branch-and-Cut-and-Price

Contents

In this chapter we present three extended linear integer programming formulations for the multi-terminal vertex separator problem. We develop Branch-and-Price algorithms for all these formulations and Branch-and-Cut-and-Price algorithms for two of them. For each formulation we present the pricing problem, the branching scheme and the computation of the dual bound used during the column generation phase. Computational results are reported comparing the performance of the formulations on a set of instances.

This chapter is organized as follows. In Sections 1,2 and 3, we present the three extended linear integer programming formulations for the MTVSP. We give a column generation scheme to solve the linear relaxation, the way to compute the lagrangian bound. We then present the branching scheme used in the Branch-and-Price algorithms. Section 4, is devoted to the Branch-and-Cut-and-Price algorithms and Section 5 to the numerical results obtained from the above algorithms.

Throughout this chapter, we suppose that Hypotheses 2.1-2.5, defined in Chapter 2, are satisfied.

4.1 The terminal-set formulation

A *terminal-set* $W \subseteq V \cup T$ is a set of vertices containing exactly one terminal and such that the neighbors of W do not contain terminals, *i.e.*,$|W \cap T| = 1$ and $N(W) \cap T = \emptyset$. Let $\mathcal{W}$ be the set of all terminal-sets of G.

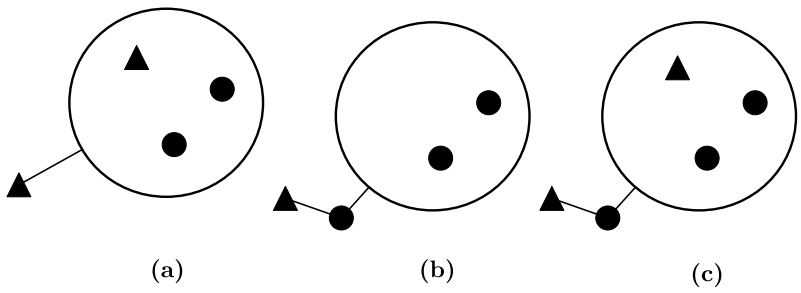


Figure 4.1: Three different subsets

Figure 4.1 shows three graphs with different configurations. The terminals are represented by triangles. The subset of vertices in Figure 4.1.(a) is not a terminal-set, since there is one terminal adjacent to at least one vertex of this subset. The subset of vertices in Figure 4.1.(b) is not a terminal-set since it contains no terminal. The subset of vertices in Figure 4.1.(c) represents a terminal-set.

In this subsection we will introduce a formulation for the MTVSP based on the terminal-sets. Indeed, any solution of the MTVSP can be seen as a partition of the vertex set into $k+1$ subsets such that k of them are disjoint terminal-sets. Thus, the variables of this formulation are on the terminal-sets. Then, the multi-terminal vertex separator problem reduces to finding k disjoint terminal-sets such that the total of their cardinality is maximum. For $W \in \mathcal{W}$, let x^W be a $0-1$ variable which takes 1 if W is selected and 0 if not. We notice that these variables are exponential in number.

Given a terminal-set $W \in \mathcal{W}$, let $a^W \in \{0,1\}^E$ and $b^W \in \{0,1\}^{V \cup T}$ be the vectors defined as follows

$$a^W_{uv} = \begin{cases} 1 & \text{if } \{u,v\} \cap W \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } uv \in E,$$

$$b^W_v = \begin{cases} 1 & \text{if vertex } v \in W, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } v \in V \cup T.$$

The MTVSP is equivalent to the following integer linear formulation

$$\max \sum_{W \in \mathcal{W}} |W| x^W \tag{4.1}$$

$$\sum_{W \in \mathcal{W}} a_e^W x^W \leq 1 \qquad \forall e \in E, \tag{4.2}$$

$$\sum_{W \in \mathcal{W}} b_v^W x^W \leq 1 \qquad \forall v \in V \cup T, \tag{4.3}$$

$$\sum_{W \in \mathcal{W}} -b_t^W x^W \leq -1 \qquad \forall t \in T, \tag{4.4}$$

$$x^W \in \{0,1\} \qquad \forall\, W \in \mathcal{W}. \tag{4.5}$$

Inequalities (4.2) permit to select disjoint terminal-sets in the solution, inequalities (4.3) bound the number of terminal-sets to which each vertex belongs. Finally, inequalities (4.4) guarantee that each terminal belongs to a terminal-set.

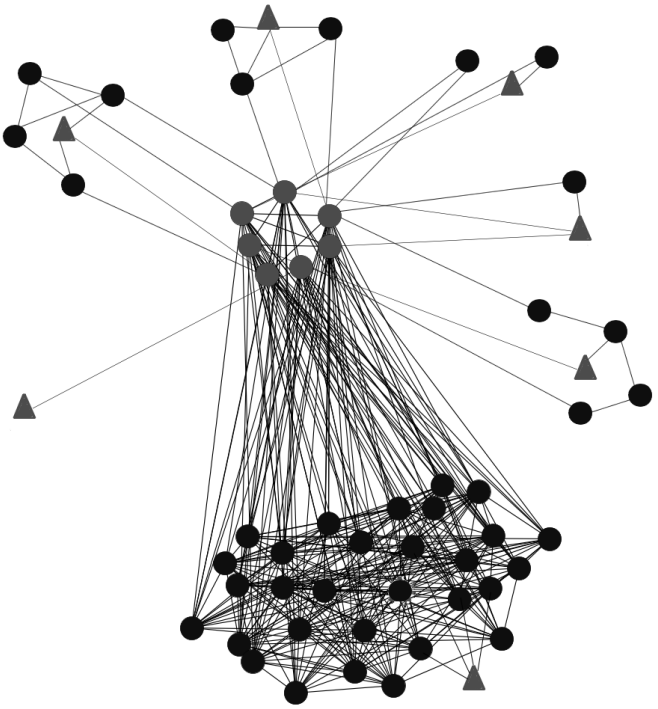


Figure 4.2: Example of terminal-sets

Figure 4.2 displays a multi-terminal vertex separator and seven terminal-sets.

Model (4.1)-(4.5) is called the *master problem*. It has an exponential size, thus we need a column generation procedure to solve its continuous relaxation.

Let $u \in \mathbb{R}_+^E$, $\eta \in \mathbb{R}_+^{V \cup T}$ and $\lambda \in \mathbb{R}_+^T$ be the dual variables associated with inequalities (4.2) – (4.4). The dual of the linear relaxation of (4.1)-(4.5), denoted by DLMP, is defined as follows

$$\begin{aligned} \min \sum_{e \in E} u^e + \sum_{v \in V \cup T} \eta_v - \sum_{t \in T} \lambda_t & & \\ \sum_{e \in E} a_e^W u^e + \sum_{v \in V \cup T} \eta_v b_v^W - \sum_{t \in T} \lambda_t b_t^W \geq |W| & \qquad \forall\, W \in \mathcal{W}, & (4.6) \\ u_e \geq 0 & \qquad \forall e \in E, & \\ \eta_v \geq 0 & \qquad \forall v \in V \cup T, & \\ \lambda_t \geq 0 & \qquad \forall t \in T. & \end{aligned}$$

The master problem is initialized with a subset of variables, and then the additional variables necessary to solve its linear relaxation, are generated by separating the associated dual constraints (4.6) given below. This constitutes the *pricing problem*. DLMP will be used for defining the pricing problem and computing the lagrangian bound.

4.1.1 Pricing problem

Given a dual solution $\pi = (u, \eta, \lambda)$, the pricing problem is equivalent to the separation of the dual constraints (4.6), *i.e.,* it consists in finding a subset $W \in \mathcal{W}$ such that

$$\sum_{e \in E} a_e^W u^e + \sum_{v \in V \cup T} \eta_v b_v^W - \sum_{t \in T} \lambda_t b_t^W < |W|.$$

This can be tackled as an optimization problem. It can be modeled as a Binary Linear Program using variable $y(v)$ which models coefficient b_v^W, *i.e.,* the fact that vertex v belongs or not to W, and variables $z(e)$, which models coefficient a_e^W, *i.e.,* the fact that at least one vertices of e is in subset W or not, that is

$$y(v) = \begin{cases} 1 & \text{if vertex } v \in W, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } v \in V \cup T,$$

$$z(uv) = \begin{cases} 1 & \text{if } \{u, v\} \cap W \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } uv \in E.$$

We notice that y represents a terminal-set W. Thus, $|W| = \sum_{v \in V \cup T} y(v)$ and then inequality (4.6) associated with W is equivalent to the following

$$\sum_{e \in E} z(e)u^e + \sum_{v \in V \cup T} \eta_v y(v) - \sum_{t \in T} \lambda_t y(t) \geq \sum_{v \in V \cup T} y(v),$$

which can be written as

$$\sum_{e \in E} z(e)u^e + \sum_{v \in V} \eta_v y(v) - \sum_{t \in T} (\lambda_t - \eta_t) y(t) \geq \sum_{v \in V \cup T} y(v),$$

then

$$\sum_{e \in E} -z(e)u^e + \sum_{v \in V} (1 - \eta_v) y(v) + \sum_{t \in T} (1 - \eta_t + \lambda_t) y(t) \leq 0.$$

Therefore, the pricing problem is equivalent to the following mixed integer linear program P',

$$\max \sum_{e \in E} -u^e z(e) + \sum_{v \in V} (1 - \eta_v) y(v) + \sum_{t \in T} (1 - \eta_t + \lambda_t) y(t)$$

$$z(uv) \geq y(u) \qquad \forall uv \in E, \tag{4.7}$$

$$z(uv) \geq y(v) \qquad \forall uv \in E, \tag{4.8}$$

$$z(uv) \leq y(u) + y(v) \qquad \forall uv \in E, \tag{4.9}$$

$$z(ut') \leq 1 - y(t) \qquad \forall t \in T, \forall t' \in T \setminus \{t\}, \forall ut' \in E, \tag{4.10}$$

$$\sum_{t \in T} y(t) \geq 1, \tag{4.11}$$

$$z(e) \geq 0 \qquad \forall e \in E, \tag{4.12}$$

$$y(v) \geq 0 \qquad \forall v \in V, \tag{4.13}$$

$$y(v) \leq 1 \qquad \forall v \in V, \tag{4.14}$$

$$y(v) \in \{0, 1\} \qquad \forall v \in V \cup T, \tag{4.15}$$

$$z(e) \in \{0, 1\} \qquad \forall e \in E. \tag{4.16}$$

Inequalities (4.7) – (4.9) link variables z and y. If a terminal $t \in T$ is in the solution, inequalities (4.10) ensure that all nodes adjacent to the terminals of $T \setminus \{t\}$ are not selected in the solution. Finally, inequalities (4.11) guarantee that at least one terminal is selected.

If the optimal objective value of P' is greater than 0, then we add the column corresponding to W, the optimal solution of P', to the restricted master problem (RMP).

Otherwise, the current solution of the RMP is optimal.

Lemma 4.1 *Inequalities (4.9) can be deleted.*

Proof. Let P'' be the linear program obtained from P' by deleting inequalities (4.9). Consider an optimal solution (y, z) of P''. Let (y', z') be the solution obtained from (y, z) by setting $z'_{uv} = 0$ for all $uv \in E$, with $y(u) = y(v) = 0$. Clearly, (y', z') satisfies the inequalities of P'. Moreover, since the weight of z is less or equal to 0, (y', z') is an optimal solution of P'. ■

Lemma 4.2 *Constraints (4.16) can be deleted.*

Proof. Let P'' be the linear program obtained from P' by deleting constraints (4.16). Let (y, z) be an optimal fractional solution of P''. Hence, there exists $e \in E$ such that $z(e)$ is fractional. As a consequence, inequalities (4.7) − (4.10) associated with e are not tight for (y, z). Let (y', z') be the solution obtained from (y, z) by setting $z'_e = 0$ for all fractional variable $z(e)$. Clearly, (y', z') satisfies the inequalities of P'. Moreover, since the weight of z is less or equal to 0, (y', z') is an optimal solution of P'. ■

In the following, let P'' be the linear program obtained from P' by deleting inequalities (4.9) and constraints (4.16).

Corollary 4.3 *Solving the pricing problem reduces to solving P''.*

A matrix A is *totally unimodular* if each sub-determinant of A is $0, +1$, or -1. Obviously, each entry of an unimodular matrix is 0, $+1$, or -1. If A is *totally unimodular* then for an integer vector b, the polytope given by $Ax \leq b$ is integral.

Hence, the following holds.

Theorem 4.4 *The pricing problem can be solved in polynomial time.*

Proof. As a terminal-set contains exactly one terminal, looking for a terminal-set can be decomposed as a sequence of subproblems, each subproblem consists in finding

a terminal-set with a specific terminal t. From Corollary 4.3, this is equivalent to solve P'' by fixing one variable $y(t)$ to 1 and all other variables $y(t')$, $t' \in T \setminus \{t\}$ to 0. Consider the linear relaxation of P''. By fixing the values of y, inequality (4.11) becomes redundant and can then be deleted. Now it is easy to see that the matrix of the remaining linear system is totally unimodular. It follows that the polyhedron given by (4.7) and (4.8) together with (4.10) and (4.12) – (4.14) is integer. Consequently, the pricing problem reduces to solving k linear programs with polynomial size. Which can be done in a polynomial time. ■

4.1.2 Heuristic algorithm for the pricer

Since P' is a mixed integer linear program, its resolution may take time. It would be then interesting to use a heuristic in order to speed up the resolution. In this subsection we propose a heuristic for the pricing problem. The first step of the algorithm consists in selecting a terminal which may be an appropriate terminal for a terminal-set violating (4.6). We have remarked from the numerical tests that terminal t^* maximizing $(1 - \eta_t + \lambda_t) + \sum_{tv \in E} -u^{tv}$ gives good results. The second step consists in adding one by one each vertex of V, not adjacent to terminals of $T \setminus \{t^*\}$, to W and to check whether the value of W increases or not. If it decreases, we do not add it to W. This heuristic is given in Algorithm 13

Algorithm 13: Heuristic for the pricing problem

Data: Graph $G = (V \cup T, E)$ and $\pi = (u, \eta, \lambda)$
Result: Inequality (4.6) violated by π
begin
  $t^* \leftarrow \underset{t \in T}{\text{argmax}}(1 - \eta_t + \lambda_t + \sum_{tv \in E} -u^{tv})$;
  $W \leftarrow W \cup \{t^*\}$;
  $Z^W \leftarrow 1 - \eta_{t^*} + \lambda_{t^*} + \sum_{(t^*v) \in E} -u^{t^*v}$;
  for $(v \in V \setminus N(T \setminus \{t^*\}))$ **do**
    $tmp \leftarrow Z^W + (1 - \eta_v)$;
    for $(vu \in E)$ **do**
      if $(u \notin W)$ **then**
        $tmp \leftarrow tmp - u^{vu}$;
      end
    end
    if $(tmp > Z^W)$ **then**
      $Z^W = tmp$;
      $W \leftarrow W \cup \{v\}$;
    end
  end
  Check if inequality (4.6) associated with W is violated by π ;
end

The above algorithm runs in $O(nm)$-time.

4.1.3 Basic columns

The first restricted master problem is associated to a collection of terminal-sets $\mathcal{W}^1 \subset \mathcal{W}$. It is important to have a good collection $\mathcal{W}^1$, this can improve the resolution time of the problem. For our algorithm, $\mathcal{W}^1$ contains k terminal-sets of cardinality 1, each consists of only one terminal, See Figure 4.3.(a). Set $\mathcal{W}^1$ also contains k maximal terminal-sets $W_{t_1}, \ldots, W_{t_k}$, with terminals $t_1, \ldots, t_k$ of T, respectively. *i.e.,* $W_{t_i} = \{t_i\} \cup (V \setminus N(T \setminus \{t_i\}))$, See Figure 4.3.(b).

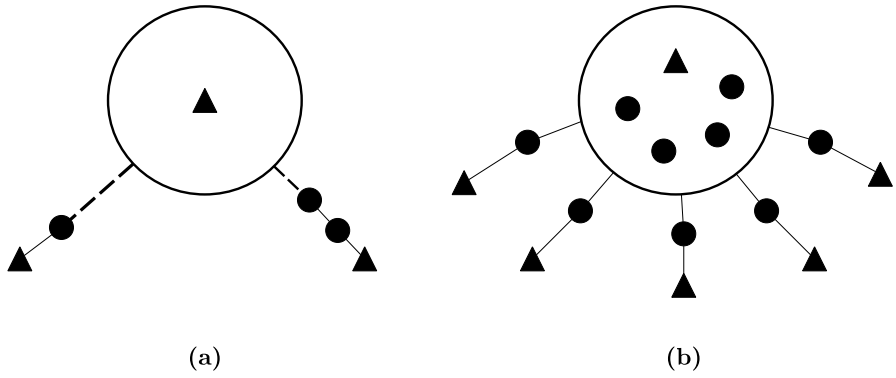


Figure 4.3: Basic Columns

4.1.4 Column generation strategy

The strategy for finding new columns is to use the heuristic, presented in Algorithm 13, in the beginning of the column generation process. Once the heuristic generates a terminal-set W with non-positive value (that is the value of the objective function is nonpositive), we start using the exact method by solving P' to generate columns.

4.1.5 Branching scheme

Consider a restricted master problem associated with $\mathcal{W}' \subset \mathcal{W}$. For two vertices $u, v \in V \cup T$, let $\mathcal{W}'_{u,v} \subseteq \mathcal{W}'$ be the set of all terminal-sets in $\mathcal{W}'$ containing u and v. The branching scheme that we will use is as follows. If for $u, v \in V \cup T$, $0 < \sum_{W \in \mathcal{W}'_{u,v}} x^W < 1$, then the branching generates two nodes by imposing either

$$\sum_{W \in \mathcal{W}'_{u,v}} x^W = 1,$$

or

$$\sum_{W \in \mathcal{W}'_{u,v}} x^W = 0.$$

Lemma 4.5 *For any fractional solution, there exists a pair of terminals $u, v \in V \cup T$ such that $0 < \sum_{W \in \mathcal{W}'_{u,v}} x^W < 1$.*

Proof. Consider a restricted master problem (RMP) associated with a set of terminal-sets $\mathcal{W}' \subseteq \mathcal{W}$. Suppose that the solution, x, of this RMP is fractional. Consider a terminal-set $W \in \mathcal{W}$ such that $0 < x^W < 1$ and $|W|$ is maximum. Since $W \neq \emptyset$, let $v \in W$. We distinguish two cases.

- Case 1: $|W| = 1$. It then follows that v is a terminal and by $(4.3) - (4.4)$, there exists another terminal-set $W' \in \mathcal{W}$ such that $v \in W'$ and $0 < x^{W'} < 1$. As $W \neq W'$ and $v \in W \cap W'$, it follows that $|W'| > |W|$. This is a contradiction with the fact that $|W|$ is maximum.
- Case 2: $|W| \geq 2$. By (4.3), $\sum_{W \in \mathcal{W}'} b_v^W x^W \leq 1$.

  If $\sum_{W \in \mathcal{W}'} b_v^W x^W < 1$, then it is clear that for all $u \in W \backslash \{v\}, 0 < \sum_{W \in \mathcal{W}'_{u,v}} x^W < 1$ and the lemma follows.

  If $\sum_{W \in \mathcal{W}'} b_v^W x^W = 1$, then again the results follows. For otherwise, for all vertex $u \in W \setminus \{v\}$, $\sum_{W \in \mathcal{W}'_{u,v}} x^W = 1$ and hence there would exist a column corresponding to a terminal-set W' that contains W. This contradicts the fact that $|W|$ is maximum.

■

A branching scheme is said to be *complete*, if it can generate any feasible solution.

Corollary 4.6 *The branching scheme is complete.*

4.1.6 Lagrangian bound

The lagrangian bound is a value that represents the dual bound of the linear relaxation of the master problem (LMP). It is the value of the objective function of DLMP (dual of the LMP).

To compute the lagrangian bound, we need a feasible solution of DLMP. In what follows we will show how to construct a feasible solution of DLMP during the column generation phase and then how to compute the lagrangian bound for our problem. Let $\pi = (u, \eta, \lambda)$ be a dual vector obtained from a restricted master problem. And let Z

be the optimal value of the pricer with respect to π.

Remark that for all $W \in \mathcal{W}$,

$$|W| \leq (\sum_{e \in E} a_e^W u^e + \sum_{v \in V \cup T} \eta_v b_v^W - \sum_{t \in T} \lambda_t b_t^W) + Z.$$

Each terminal-set $W \in \mathcal{W}'$ contains exactly one terminal. It follows that $\sum_{t \in T} b_t^W = 1$. Moreover, for all terminal $t \in T$, b_t^W is integer. Let $\overline{\eta} \in \mathbb{R}_+^{V \cup T}$ be the vector defined as follows

- $\overline{\eta}_v = \eta_v$ for all vertex $v \in V$,
- $\overline{\eta}_t = \eta_t + Z$ for all terminal $t \in T$.

It follows that for all $W \in \mathcal{W}$

$$|W| \leq \sum_{e \in E} a_e^S u^e + \sum_{v \in V \cup T} \overline{\eta}_v b_v^W - \sum_{t \in T} \lambda_t b_t^W.$$

Thus, $(u, \overline{\eta}, \lambda)$ is a feasible solution for DLMP and $LB = \sum_{e \in E} u^e + \sum_{v \in V \cup T} \overline{\eta}_v - \sum_{t \in T} \lambda_t$ is a lower bound for LMP.

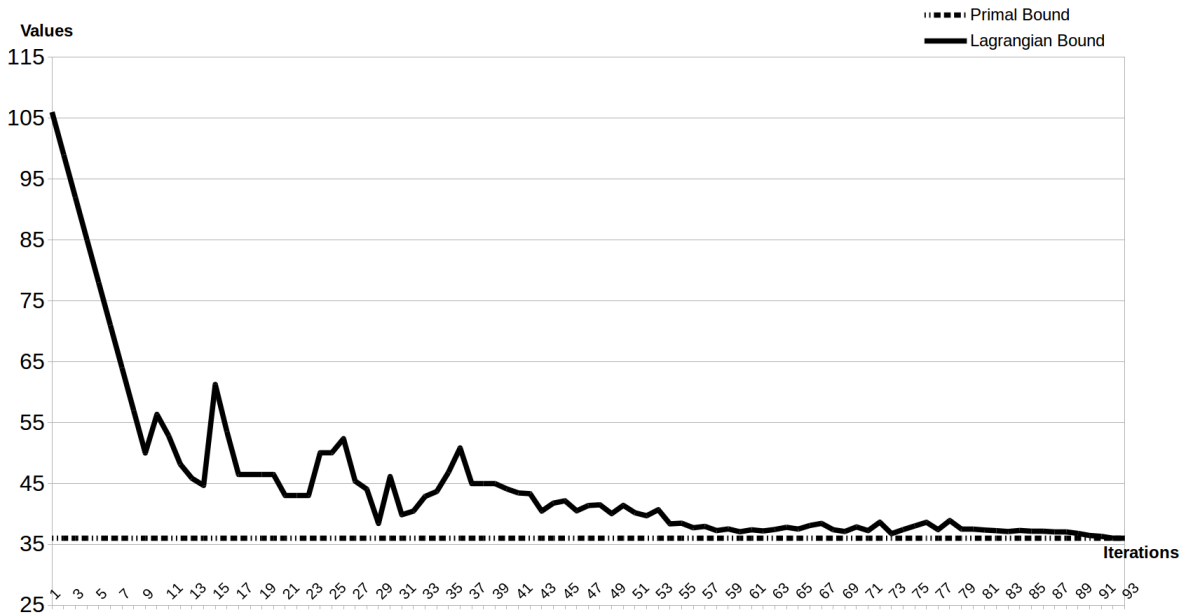


Figure 4.4: Example of the lagrangian bound during column generation solving

Figure 4.4 displays the state of the lagrangian bound and the primal bound during the column generation phase. We notice that the lagrangian bound decreases until it equals the primal bound.

4.2 The isolating-separator formulation

For a terminal $t \in T$, an *isolating-separator* $S^t \subseteq V$ is a set of vertices that intersects all paths between t and the terminals of $T \setminus \{t\}$. For a terminal $t \in T$, let $\mathcal{S}^t$ be the set of all isolating-separators in G associated with t. Let $\mathcal{S} = \bigcup_{t \in T} \mathcal{S}^t$ be the set of all isolating-separators in G.

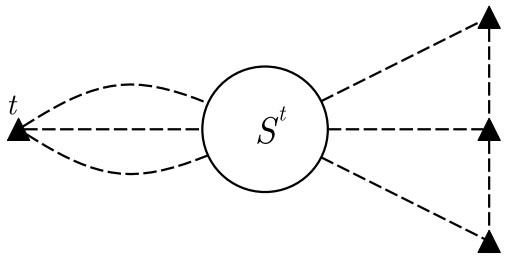


Figure 4.5: Example of an isolating-separator

In Figure 4.5, set S^t represents an isolating-separator since it intersects all paths (represented by dashed lines) between t and all other terminals.

Lemma 4.7 *A MTV separator is an isolating-separator that can be associated to any terminal.*

Proof. For all $t \in T$, a MTV separator intersects all terminal paths between t and all terminals of $T \setminus \{s\}$, thus it is an isolating-separator of t. ■

Lemma 4.8 *A set $S \subseteq V$ is a MTV separator, if and only if, each terminal of T can be associated to an isolating-separator included in S.*

Proof. ($\Rightarrow$) Suppose that there is at least one terminal $t \in T$ that cannot be associated to any isolating-separator included in S. It follows that S is not an isolating-separator for t. Thus there exists a terminal path between t and another terminal that does not intersect S. Therefore S is not a MTV separator.

($\Leftarrow$) Suppose that S is not a MTV separator. It follows that there exists a terminal path $P_{tt'}$ between t and t' that does not intersect S. Thus, no subset of S intersects $P_{tt'}$. Therefore, no subset of S is an isolating-separator of t. ■

Lemma 4.9 *An isolating-separator may be related to several terminals of T.*

Proof. Let $S \subset V$ be a subset of vertices that intersects all terminal paths between a terminal $t_0 \in T$ and all terminals $t \in T \setminus \{t_0\}$ and intersects all terminal paths between $t_1 \in T \setminus \{t_0\}$ and all terminals of $T \setminus \{t_1\}$. Clearly, S is an isolating-separator of t_0 and t_1. ■

In this subsection we will introduce a formulation for the MTVSP based on the isolating-separators. Indeed, any solution of the MTVSP can be seen as a union of several isolating-separators, each one intersects all terminal paths between one specific terminal and all the other terminals. Then, the multi-terminal vertex separator problem reduces to finding a set of isolating-separators such that the cardinality of their union is minimum. Thus, two families of variables are needed for this formulation, one family is on the isolating-separator and the other one is on the vertices. Given an isolating-separator $S \in \mathcal{S}$, let x^S be a variable which takes 1 if S is chosen and 0 if not. And for $v \in V$, let y_v be a variable such that $y(v)$ takes 1 if v belongs to at least one selected isolating-separator and 0 if not. We notice that the variables x are exponential in number.

Given a isolating-separator $S \in \mathcal{S}$, let $a^S \in \{0,1\}^{V \times T}$, $\overline{a}^S \in \{0,1\}^V$ and $b^S \in \{0,1\}^T$ be the vectors defined as follows

$$a^S_{v,t} = \begin{cases} 1 & \text{if } v \text{ belongs to } S \text{ and } S \in \mathcal{S}^t, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } v \in V,\ t \in T,$$

$$\overline{a}^S_v = \begin{cases} 1 & \text{if } v \text{ belongs to } S, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } v \in V,$$

$$b^S_t = \begin{cases} 1 & \text{if } S \in \mathcal{S}^t, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } t \in T.$$

The MTVSP is equivalent to the following integer linear program

$$\min \quad y(V) \tag{4.17}$$

$$y(v) - \sum_{S \in \mathcal{S}} a_{v,t}^S x^S \geq 0 \qquad \forall t \in T, \quad \forall v \in V, \tag{4.18}$$

$$-y(v) + \sum_{S \in \mathcal{S}} \overline{a}_v^S x^S \geq 0 \qquad \forall v \in V, \tag{4.19}$$

$$\sum_{S \in \mathcal{S}^t} b_t^S x^S = 1 \qquad \forall t \in T, \tag{4.20}$$

$$x^S \geq 0 \qquad \forall\ S \in \mathcal{S}, \tag{4.21}$$

$$y(v) \in \{0,1\} \qquad \forall\ v \in V. \tag{4.22}$$

Inequalities (4.18) ensure that a vertex belonging to a selected isolating-separator, belongs also to the separator. Inequalities (4.19) ensure that a vertex belonging to the separator, belongs to at least one isolating-separator. Inequalities (4.20) guarantee that at least one isolating-separator, associated with each terminal, is selected.

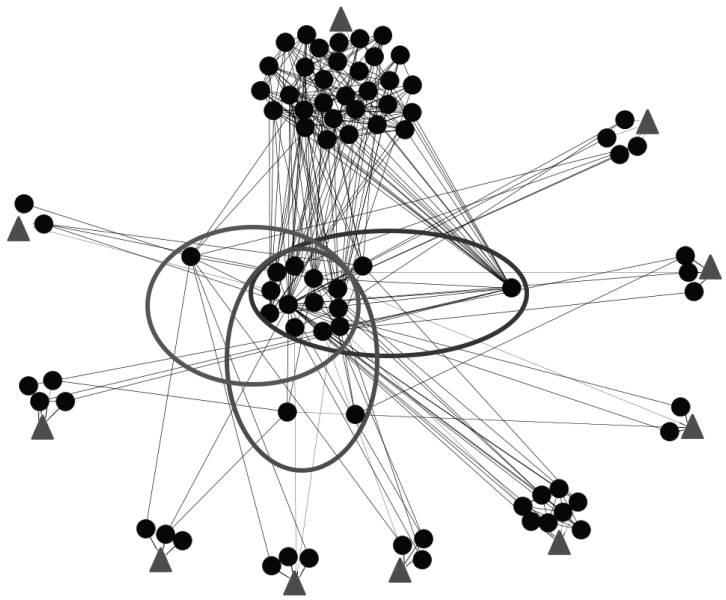


Figure 4.6: Example of a solution of the isolating-separator formulation

Figure 4.6 illustrates a solution of the MTVSP. The three subsets in the Figure represent selected isolating-separators. The union of the three isolating-separators represents a multi-terminal vertex separator for G. We note that one isolating-separator can be associated to several terminals.

Model (4.17)-(4.22) is the *master problem*. It has an exponential size, thus we need a column generation procedure to solve its continuous relaxation.

Let $u \in \mathbb{R}_+^{T \times V}$, $\lambda \in \mathbb{R}_+^V$ and $\eta \in \mathbb{R}_+^T$ be the dual variables associated with inequalities $(4.18) - (4.20)$. The dual of the linear relaxation of the integer linear program (4.17)-(4.22), denoted by DLMP, is given by the following program

$$\begin{aligned}
\max \sum_{t \in T} \eta_t & & & \\
\sum_{t \in T} u_v^t - \lambda_v \leq & \quad 1 & \forall v \in V, & \\
\sum_{v \in V} \sum_{t \in T} -a_{v,t}^S u_v^t + \sum_{v \in V} \overline{a}_v^S \lambda_v + \sum_{t \in T} b_t^S \eta_t \leq & \quad 0 & \forall S \in \mathcal{S}, & \qquad (4.23) \\
u_v^t \geq & \quad 0 & \forall t \in T, v \in V, & \\
\lambda_v \geq & \quad 0 & \forall v \in V, & \\
\eta_t \in \mathbb{R} & & \forall t \in T. &
\end{aligned}$$

The master problem is initialized with a subset of variables, and the additional variables necessary to solve its linear relaxation are generated by separating the associated dual constraints (4.23). This constitutes the *pricing problem*.

4.2.1 Pricing problem

Given a dual solution $\pi = (u, \lambda, \eta)$, the pricing problem is equivalent to the separation of the dual constraints (4.23), *i.e.,* it consists in finding a subset $S \in \mathcal{S}$ such that

$$\sum_{v \in V} \sum_{t \in T} -a_{v,t}^S u_v^t + \sum_{v \in V} \overline{a}_v^S \lambda_v + \sum_{t \in T} b_t^S \eta_t > 0.$$

Then the pricing problem reduces to generating an isolating-separator S^t such that $-\sum_{v \in V} u_v^t a_{vt}^S + \sum_{v \in V} \lambda_v \overline{a}_v^S + \eta_t b_t$ is maximum. If this value is greater than 0, then we add the corresponding column to the RMP. Otherwise, the current solution of the RMP is optimal.

Theorem 4.10 *The pricing problem can be solved in polynomial time.*

Proof. For a terminal $t \in T$, let $G^t = (V \cup \{t, t_s\}, E)$ be the graph obtained from G by merging all terminals of $T \setminus \{t\}$ into one terminal t_s. Let $c \in \mathbb{R}^V$ such that $c(v) = u_v^t - \lambda_v$ for all $v \in V$. To solve the pricing problem, we need to iterate over each terminal $t \in T$ and to solve the minimum two-terminal vertex separator problem in graph G^t. The latter problem can be solved in polynomial time. Let Z_t^* be the value of the optimal two-terminal vertex separator in graph G^t. The optimal solution for the pricer is the two-terminal separator with maximum value $Z_t^* - \eta_t$. It follows that the pricer problem can be solved in polynomial time. ■

4.2.2 Basic columns

Let us remark that set V represents an isolating-separator for any terminal of T. Thus, $V \in \mathcal{S}^t$ for all $t \in T$. So, for the basic columns we can add a variable associated with set V for each terminal of T. Moreover, any multi-terminal vertex separator is also an isolating-separator for any terminal of T. Thus, for each terminal $t \in T$, we add a variable associated with an isolating-separator S^t that is a multi-terminal vertex separator. This is obtained using the improved isolating terminal heuristic (Algorithm 10 presented in Chapter 2).

4.2.3 Column generation strategy

The column generation strategy aims at generating several columns at each iteration of the column generation phase. This may speed up the resolution time. It is based on the fact that an isolating-separator may be related to several terminals of T. Therefore, each time we generate a new column, we check if the associated isolating-separator can also correspond to other terminals. Then, we add several columns by iteration.

4.2.4 Branching scheme

The branching scheme is reduced to the classical branching on y. Then, the branching generates two nodes by imposing either $y(v) = 1$ or $y(v) = 0$ for some fractional variable $y(v)$.

4.2.5 Lagrangian bound

Let $\pi = (u, \lambda, \eta)$ be a dual vector obtained from the linear relaxation of a restricted master problem. Let Z be the optimal value of the pricer. Remark that

$$\sum_{v \in V} \sum_{t \in T} -a_{v,t}^S u_v^t + \sum_{v \in V} \overline{a}_v^S \lambda_v + \sum_{t \in T} b_t^S \eta_t \leq Z \qquad \forall S \in \mathcal{S}.$$

We know that Let $\overline{\eta} \in \mathbb{R}_+^T$ be a vector defines as follows

- $\overline{\eta}_t = \eta_t - Z$ for all terminal $t \in T$.

Since $\sum_{t \in T} b_t^S = 1$, it follows that

$$\sum_{v \in V} \sum_{t \in T} -a_{v,t}^S u_v^t + \sum_{v \in V} \overline{a}_v^S \lambda_v + \sum_{t \in T} b_t^S \overline{\eta}_t \leq 0 \qquad \forall S \in \mathcal{S}$$

Consequently, $\pi = (u, \lambda, \overline{\eta})$ is a feasible solution for the DLMP and $LB = \sum_{t,t' \in T} \overline{\eta}_t$ is a lower bound for LMP.

4.3 The terminal-pair-separator formulation

A *terminal-pair-separator* S is a set of vertices whose removal disconnects a pair of terminals. We say that S *separates* this pair of terminals. For $t, t' \in T$, let $\mathcal{S}_{t'}^t \subseteq \mathcal{S}$ be the set of all terminal-pair-separators of G that separate t and t'. Let $\mathcal{S} = \bigcup_{t,t' \in T} \mathcal{S}_{t'}^t$ be the set of all terminal-pair-separators of G.

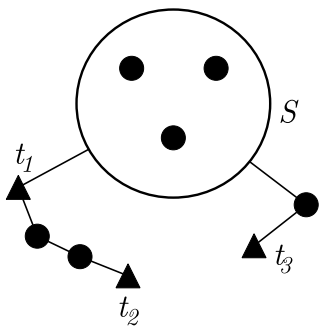


Figure 4.7: Example of a terminal-pair-separator

Figure 4.7 gives an example of a terminal-pair-separator. We can see that S is a terminal-pair-separator for terminals t_1 and t_3 but not a terminal-pair-separator for terminals t_1 and t_2.

Lemma 4.11 *A MTV separator is a terminal-pair-separator that can be associated to any terminal.*

Proof. For all $t, t' \in T$, a MTV separator intersects all terminal paths between t and t', thus it is a terminal-pair-separator for t and t'. ■

Lemma 4.12 *A set $S \subseteq V$ is a MTV separator, if and only if, each pair of terminals in T can be associated to a terminal-pair-separator included in S.*

Proof. ($\Rightarrow$) Suppose that there is at least one pair of terminals $t, t' \in T$ that cannot be associated to any terminal-pair-separator included in S. It follows that S is not a terminal-pair-separator for t and t'. Thus there exists a terminal path between t and t' that does not intersect S. Therefore S is not a MTV separator.

($\Leftarrow$) Suppose that S is not a MTV separator. It follows that there exists a terminal path $P_{tt'}$ between t and t' that does not intersect S. Thus, no subset of S intersects $P_{tt'}$. Therefore, no subset of S is a terminal-pair-separator for t and t'. ■

Lemma 4.13 *A terminal-pair-separator may be related to several terminals of T.*

Proof. Let $S \subset V$ be a subset of vertices that intersects all terminal paths between a terminal $t_0 \in T$ and $t_1 \in T \setminus \{t_0\}$ and intersects all terminal paths between $t_2 \in T \setminus \{t_0, t_1\}$ and $t_3 \in T \setminus \{t_2\}$. Clearly, S is a terminal-pair-separator for t_0 and t_1 and also for t_2 and t_3. ■

In this subsection we will introduce a formulation for the MTVSP based on the terminal-pair-separators. Indeed, any MTV separator of G can be seen as a union of several terminal-pair-separators. Then, the multi-terminal vertex separator problem reduces to finding a set of terminal-pair-separators such that the cardinality of their union is minimum. For this formulation we consider two families of variables, one is associated to the terminal-pair-separators and the other is related to the vertices. For

$S \in \mathcal{S}$, let x^S be a variable which takes 1 if S is chosen and 0 if not. And for $v \in V$, let y_v be a variable which takes 1 if v is chosen and 0, if not. Note that the variables x are exponential in number.

Given a terminal-pair-separator $S \in \mathcal{S}$, let $a^S \in \{0,1\}^{V \times T \times T}$ and $b^S \in \{0,1\}^{T \times T}$ be the vectors defined as follows

$$a^S_{tt'v} = \begin{cases} 1 & \text{if } v \in S \text{ and } S \in \mathcal{S}^t_{t'}, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } t, t' \in T \text{ and } v \in V,$$

$$b^S_{tt'} = \begin{cases} 1 & \text{if } S \text{ separates } t \text{ and } t', \\ 0 & \text{otherwise.} \end{cases} \quad \text{for all } t, t' \in T,$$

The MTVSP is equivalent to the following integer linear formulation

$$\min \quad y(V) \tag{4.24}$$

$$-\sum_{S \in \mathcal{S}} a^S_{tt'v} x^S + y(v) \geq 0 \qquad \forall v \in V, \forall t, t' \in T, \tag{4.25}$$

$$-\sum_{S \in \mathcal{S}} b^S_{tt'} x^S \geq -1 \qquad \forall t, t' \in T, \tag{4.26}$$

$$\sum_{S \in \mathcal{S}} b^S_{tt'} x^S \geq 1 \qquad \forall t, t' \in T, \tag{4.27}$$

$$y(v) \in \{0,1\} \qquad \forall v \in V, \tag{4.28}$$

$$x^S \geq 0 \qquad \forall S \in \mathcal{S}. \tag{4.29}$$

Inequalities (4.25) ensure that each vertex belonging to at least one selected terminal-pair-separator, also belongs to the separator. Inequalities (4.25) – (4.26) allow to select exactly one terminal-pair-separator per pair of terminals.

Theorem 4.14 *For each feasible solution (x, y) of (4.25) – (4.29), y is an incidence vector of a multi-terminal vertex separator.*

Proof. By (4.26) – (4.27), for each pair of terminals $t, t' \in T$, there is at least one $S \in \mathcal{S}^t_{t'}$ such that $x^S > 0$. Denote this separator by $S^t_{t'}$. It is clear that $\widehat{S} = \bigcup_{t,t'} S^t_{t'}$ is a multi-terminal vertex separator of G. As y is integer, by (4.25), it follows that for all $t, t' \in T$ and $v \in S^t_{t'}$, $y_v = 1$. ■

Model (4.24)-(4.29) is the *master problem*. It has an exponential size, thus we need a column generation procedure to solve its continuous relaxation.

Let $u \in \mathbb{R}_+^{V\times T\times T}$, $\eta \in \mathbb{R}_+^{T\cup T}$ and $\mu \in \mathbb{R}_+^{T\cup T}$ be the dual variables associated with inequalities $(4.25) - (4.27)$. The dual of the linear relaxation of (4.24)-(4.29), denoted by DLMP, is given by

$$\max \sum_{t,t'\in T} \mu_{tt'} - \eta_{tt'}$$

$$\sum_{t,t'\in T} (\sum_{v\in V} -a^S_{tt'v}u^v_{tt'} + (\mu_{tt'} - \eta_{tt'})b^S_{tt'}) \leq 0 \qquad \forall S \in \mathcal{S} \tag{4.30}$$

$$\sum_{t,t'\in T} u^v_{tt'} \leq 1 \qquad \forall v \in V \tag{4.31}$$

$$u^v_{tt'} \geq 0 \qquad \forall v \in V, \forall t,t' \in T \tag{4.32}$$

$$\eta_{tt'} \geq 0 \qquad \forall t,t' \in T \tag{4.33}$$

$$\mu_{tt'} \geq 0 \qquad \forall t,t' \in T \tag{4.34}$$

The master problem is initialized with a subset of variables, and then the additional variables necessary to solve its linear relaxation are generated by separating the associated dual constraints (4.30). This constitutes the pricing problem.

4.3.1 Pricing problem

Given a dual solution $\pi = (u, \eta, \mu)$, the pricing problem consists in finding a subset $S \in \mathcal{S}$ such that

$$\sum_{t,t'\in T} (\sum_{v\in V} -a^S_{tt'v}u^v_{tt'} + (\mu_{tt'} - \eta_{tt'})b^S_{tt'}) > 0.$$

The pricing problem reduces to generating a terminal-pair-separator S associated with two terminals $t, t' \in T$ such that $\sum_{t,t'\in T}\sum_{v\in V} -u^v_{tt'}a^S_{tt'v} + (\mu_{tt'} - \eta_{tt'})b^S_{tt'}$ is maximum.

Theorem 4.15 *The pricing problem can be solved in polynomial time.*

Proof. For all $t, t' \in T$, let $c^t_{t'} : V \cup (T \setminus \{t, t'\}) \to \mathbb{R}$ such that for all $v \in V$, $c^t_{t'}(v) = u^v_{tt'}$ and for all $t'' \in T \setminus \{t, t'\}$, $c^t_{t'}(t'') = +\infty$. Solving the pricing problem consists in iterating over each pair of terminals $t, t' \in T$, and finding a terminal-pair-separator S of minimum weight in G w.r.t $c^t_{t'}$. This is equivalent to finding a minimum cut between t and t'. It follows that the pricing problem reduces to solving a polynomial number of maximum flow problems.

4.3.2 Basic columns

It is clear that any multi-terminal vertex separator is also a terminal-pair-separator. Thus, for each pair of terminals $t, t' \in T$, we add a variable associated with a multi-terminal vertex separator obtained using the improved isolating terminal heuristic (Algorithm 10 presented in Chapter 2). We also use this heuristic to obtain a good primal bound.

4.3.3 Column generation strategy

This column generation strategy is based on the fact that the minimum two-terminal vertex separator problem can be solved in a polynomial time. The idea of this strategy is to iterate over all pairs of terminals $t, t' \in T$ and to look for the minimum two-terminal vertex separator intersecting all paths between t and t' in G w.r.t $u_{tt'}$. We stop when a two-terminal vertex separator S is found with a positive reduced cost, *i.e.,* $\sum_{t,t' \in T} \sum_{v \in V} -u^v_{tt'} a^S_{tt'v} + (\mu_{tt'} - \eta_{tt'}) b^S_{tt'} > 0$. We then add the associated column.

We remarked from the numerical tests, that when we iterate over all pairs of terminals, it would be more interesting to start with a pair of terminals $t, t' \in T$ having a maximum value $\mu_{tt'} - \eta_{tt'}$. Each time we add a new column to the restricted master problem, we check if the associated terminal-pair-separator separates other terminals. As a consequence, we may add several columns in the same iteration.

4.3.4 Branching scheme

The branching scheme is reduced to the classical branching on y. Then, the branching generates two nodes by imposing either $y(v) = 1$ or $y(v) = 0$ for some fractional variable $y(v)$.

4.3.5 The Lagrangian bound

Let $\pi = (u, \eta, \mu)$ be a dual vector obtained from the linear relaxation of a restricted master problem. Let Z be the optimal value of the pricer. Hence, w.r.t π

$$\sum_{t,t' \in T} (\sum_{v \in V} -a^S_{tt'v} u^v_{tt'} + (\mu_{tt'} - \eta_{tt'}) b^S_{tt'}) \leq Z \qquad \forall S \in \mathcal{S}.$$

Let $\overline{\eta} \in \mathbb{R}_+^{T\times T}$ be vector such that $\overline{\eta}_{tt'} = \eta_{tt'} + Z$ for all terminal $t, t' \in T$. It follows that

$$\sum_{t,t'\in T}(\sum_{v\in V} -a^S_{tt'v}u^v_{tt'} + (\mu_{tt'} - \overline{\eta}_{tt'})b^S_{tt'}) \leq 0 \qquad \forall S \in \mathcal{S}$$

Consequently, since $\sum_{t,t'\in T} b^S_{tt'} = 1$, $(u, \overline{\eta}, \mu)$ is a feasible solution for the DLMP, and $LB = \sum_{t,t'\in T} \mu_{tt'} - \overline{\eta}_{tt'}$ is a lower bound for LMP.

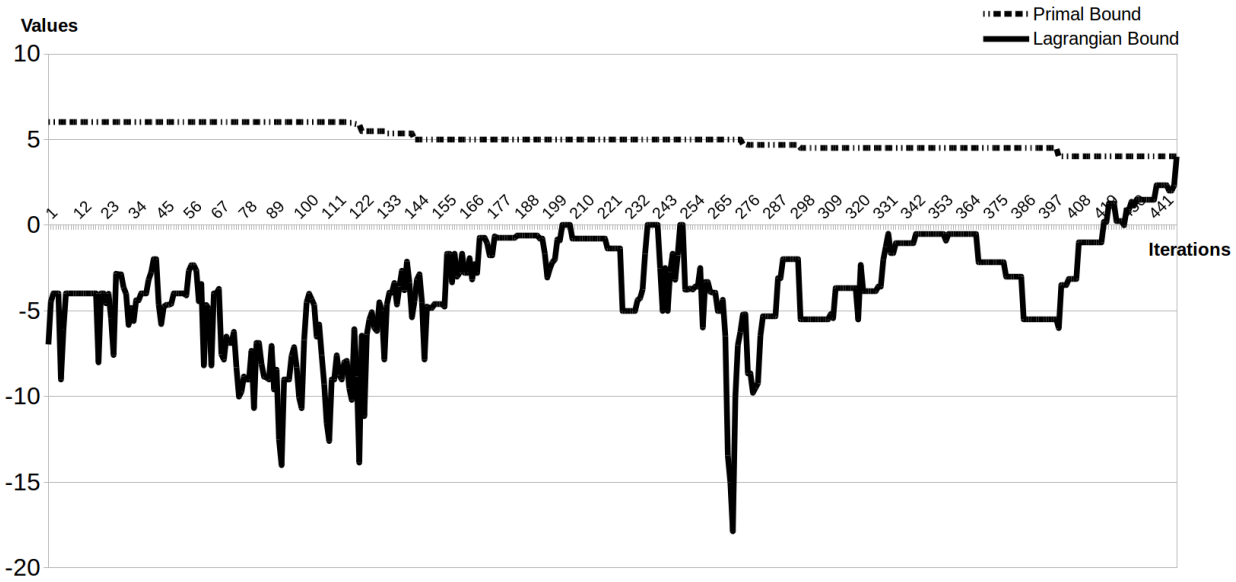


Figure 4.8: Example of lagrangian bound during column generation solving

Figure 4.8 displays the state of the lagrangian bound and the primal bound during the column generation phase. we notice that the lagrangian bound increases until it equals the primal bound.

4.4 Branch-and-Cut-and-Price

The isolating-separator and the terminal-pair-separator formulations use variable vector $y \in \{0, 1\}^V$ such that $y(v) = 1$ if v belongs to the MTV separator and 0 if not. This vector is not used in the terminal-set formulation. It follows that the valid inequalities presented in Chapter 2 can be added to the isolating-separator and the terminal-pair-separator formulations but not to the terminal-set formulation.

In this section we present the valid inequalities that we add to the isolating-separator and the terminal-pair-separator formulations in order to cut the non feasible solutions and to strengthen the linear relaxation. We add three classes of valid inequalities (2.10), (2.20) and (2.18), introduced in Chapter 2. This permits to derive a Branch-and-Cut-and-Price algorithm. We note that adding valid inequalities on y variables will not affect the pricing problems.

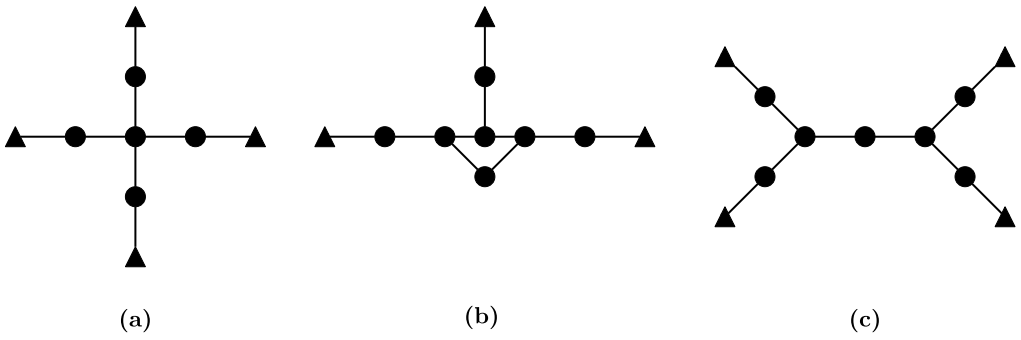


Figure 4.9: Examples of star tree, terminal cycle and terminal tree graphs

Figure 4.9 displays a star tree, a terminal cycle and a terminal tree.

The valid inequalities used in the Branch-and-Cut-and-Price algorithms are presented below.

4.4.1 Star tree inequalities

A *star tree* $H = (V(H) \cup T(H), E(H))$ of G is a tree such that the terminals of $T(H)$ are the leaves of H and all the other (non-terminal) nodes, different from the root node, are of degree two. Figure 4.9.(a) displays a star tree with 4 terminals.

Proposition 4.16 *If $H = (V(H) \cup T(H), E(H))$ is a star tree subgraph of G with root v_r, then the following star tree inequality*

$$y(V(H) \setminus \{v_r\}) + (q-1)y_{v_r} \geq q - 1$$

is valid for the polytopes associated with the isolating-separator and the terminal-pair-separator formulations.

4.4.2 Terminal cycle inequalities

A *terminal cycle* $J = (V(J) \cup T(J), E(J))$ is given by a cycle C and q disjoint paths between a vertex in C and a terminal in T. Figure 4.9.(b) displays a terminal cycle with three terminals.

Proposition 4.17 *If $J = (V(J) \cup T(J), E(J))$ is a terminal cycle subgraph of G, then the following terminal cycle inequality*

$$y(V(J)) \geq \lceil \frac{|q|}{2} \rceil$$

is valid for the polytopes associated with the isolating-separator and the terminal-pair-separator formulations.

4.4.3 Terminal tree inequalities

A *terminal tree* $H = (V(H) \cup T(H), E(H))$, is a tree such that the terminals of $T(H)$ are the leaves of H. Let $d_H(v)$ be the degree of v in H. Figure 4.9.(c) displays a terminal tree with four terminals.

Proposition 4.18 *If $H = (V(H) \cup T(H), E(H))$ is a terminal tree subgraph of G, then the following terminal tree inequality*

$$\sum_{v \in V(H)} (d_H(v) - 1) y_v \geq q - 1.$$

is valid for the polytopes associated with the isolating-separator and the terminal-pair-separator formulations.

The separation of the above inequalities is discussed in Section 2.5 of Chapter 2.

4.5 Computational Results

Based upon the previous theoretical results, we have developed a Branch-and-Price algorithm for each extended formulation presented in Sections 1, 2 and 3. Since the

isolating-separator and the terminal-pair-separator formulations use y variables, several valid inequalities presented in Chapter 2 can then be added. Thus, Branch-and-Cut-and-Price algorithms is developed for these formulations to solve efficiently the multi-terminal vertex separator problem. We use two kinds of instances, the DIMACS graph coloring instances [1] and random graphs generated using boost graph library [3]. The terminals are new nodes added to the graphs. Each terminal is randomly connected to at least 2 vertices and to at most a given $deg\_T \in \mathbb{N}$ vertices. In the Tables below, deg_T is fixed in relation with the size of the graph, *i.e.,* the higher the size of graph is, the higher deg_T is. The edges incident to the terminals are added in such a way that they respect Hypotheses 2.3-2.4. The primal bound is the value associated with any separator of G. We use the improved isolating terminal heuristic to solve the MTVSP for generating a good primal bound. This heuristic is presented in Chapter 2.
In the column generation phases, we stop generating variables, either when there is no column to add, or when the lagrangian bound is equal to the objective value.

For the Branch-and-Cut-and-Price algorithms, we add valid inequalities (2.10), (2.20) and (2.18) at the end of each column generation phase, if the linear relaxation value, in a specific node of the branching tree, is not equal to the best primal bound. We perform the separation of all cuts at each iteration. We use the separation algorithms presented in Chapter 2. The Branching tree is managed by our algorithms using a depth first search. All flow problems are solved using Lemon library [2] and the linear programs are solved using Cplex. The numerical experiments were done on an Intel Xeon E312xx machine at 2.39 GHz $\times 1$ with 48GB RAM, running under Linux 64 bits. The maximum CPU run time has been fixed to 4 hours.

In the following Tables, we have the following entries.

n	:	the number of vertices in V.
m	:	the number of edges in E.
k	:	the number of terminal in T.
Cols	:	the number of columns added during the Branch-and-Price or the Branch-and-Cut-and-Price algorithms.
ST	:	the number of the star tree inequalities generated.
TC	:	the number of the terminal cycle inequalities generated.
TT	:	the number of the terminal tree inequalities generated.
Gap %	:	the relative error between the best upper and lower bound obtained at the root node of the branching tree.
No	:	the number of node in the branching tree.
Opt	:	the optimal objective value.
Heur	:	the cardinality of the MTV separator given by the improved isolating terminal heuristic presented in Chapter 2.
CPU	:	the CPU time in seconds, given by the Branch-and-Price or the Branch-and-Cut-and-Price algorithm.

Instances	n	m	k	Cols	Opt	Heur	No	Gap %	CPU
DIMACS	47	258	6	1156	36	36	1	0.00	4.85
DIMACS	64	1477	6	1148	54	54	1	0.00	109.85
DIMACS	74	624	6	40059	-	63	1	-	-
DIMACS	80	533	6	290279	-	70	1	-	-
DIMACS	87	835	6	500703	-	79	1	-	-
DIMACS	95	778	6	13900	83	83	1	0.00	11326.50
DIMACS	96	2762	6	3159	81	81	1	0.00	11327.50
DIMACS	100	2967	8	3783	-	86	1	-	-
DIMACS	120	1307	8	14484	-	104	1	-	-
DIMACS	125	764	8	11176	-	110	1	-	-
DIMACS	128	804	8	208015	-	118	1	-	-
DIMACS	128	2370	8	4637	-	112	1	-	-
DIMACS	128	4256	8	3928	-	112	1	-	-
DIMACS	128	6462	8	2682	-	112	1	-	-
DIMACS	128	10426	8	2090	-	112	1	-	-
DIMACS	138	1022	8	242749	-	114	1	-	-
DIMACS	144	5224	8	2427	-	127	1	-	-
DIMACS	188	3920	8	2682	-	168	1	-	-
DIMACS	191	2387	8	3115	-	177	1	-	-
DIMACS	196	8399	8	1780	-	181	1	-	-
DIMACS	197	3952	8	3093	-	187	1	-	-
DIMACS	206	3576	8	3954	-	193	1	-	-
DIMACS	211	3573	8	2547	-	199	1	-	-
DIMACS	211	4132	8	2566	-	201	1	-	-
DIMACS	256	12674	8	1590	-	236	1	-	-
DIMACS	385	19129	8	1302	-	360	1	-	-
DIMACS	500	7140	8	1525	-	483	1	-	-

Table 4.1: Results with the Branch-and-Price algorithm of the terminal-subset formulation on DIMACS instances.

Instances	n	m	k	Cols	Opt	Heur	No	Gap %	CPU
Random	50	511	7	1774	39	39	1	0.00	31.17
Random	70	993	7	3548	58	58	1	0.00	3629.78
Random	100	1985	7	4159	-	89	1	-	-
Random	300	17792	7	1408	-	288	1	-	-
Random	600	70673	7	782	-	588	1	-	-
Random	700	96432	7	782	-	689	1	-	-
Random	800	125978	10	763	-	766	1	-	-
Random	900	159331	10	677	-	859	1	-	-
Random	1000	196771	10	600	-	965	1	-	-
Random	1200	283386	10	484	-	1166	1	-	-
Random	1300	332861	10	444	-	1266	1	-	-
Random	1500	442444	10	-	-	1463	-	-	-
Random	1800	637307	15	-	-	1737	-	-	-
Random	2000	786639	15	-	-	1947	-	-	-
Random	2100	867411	15	-	-	2033	-	-	-
Random	2300	1041144	15	-	-	2245	-	-	-
Random	3000	1770773	15	-	-	2940	-	-	-
Random	3300	2142370	15	-	-	3235	-	-	-
Random	3800	2841805	15	-	-	3743	-	-	-
Random	4200	3472117	15	-	-	4135	-	-	-
Random	4500	3987339	15	-	-	4434	-	-	-
Random	4800	4537794	15	-	-	4748	-	-	-
Random	5000	4922710	15	-	-	4945	-	-	-

Table 4.2: Results with the Branch-and-Price algorithm of the terminal-subset formulation on random instances.

Instances	n	m	k	Cols	Opt	Heur	No	Gap %	CPU
DIMACS	47	258	6	442	17	17	23	0.55	**0.24**
DIMACS	64	1477	6	258	16	16	21	0.52	**0.15**
DIMACS	74	624	6	308	13	14	34	0.24	**0.22**
DIMACS	80	533	6	1596	16	16	97	0.39	**2.25**
DIMACS	87	835	6	419	14	14	31	0.40	**0.31**
DIMACS	95	778	6	475	18	18	25	0.57	**0.24**
DIMACS	96	2762	6	622	21	21	31	0.62	**0.53**
DIMACS	100	2967	8	871	22	22	33	0.63	1.00
DIMACS	120	1307	8	16775	-	24	60	-	-
DIMACS	125	764	8	14887	23	23	117	0.64	13396.50
DIMACS	128	804	8	4791	16	16	63	0.39	23.24
DIMACS	128	2370	8	11337	22	24	245	0.47	1386.72
DIMACS	128	4256	8	7865	24	24	131	0.60	342.64
DIMACS	128	6462	8	1327	24	24	45	0.60	4.27
DIMACS	128	10426	8	863	24	24	35	0.60	3.38
DIMACS	138	1022	8	44835	-	26	1891	-	-
DIMACS	144	5224	8	3995	25	25	35	0.56	25.41
DIMACS	188	3920	8	904	28	28	41	0.60	**1.53**
DIMACS	191	2387	8	2069	22	22	33	0.63	5.19
DIMACS	196	8399	8	1801	23	23	37	0.70	5.39
DIMACS	197	3952	8	606	18	18	27	0.64	0.72
DIMACS	206	3576	8	961	21	21	29	0.56	**1.39**
DIMACS	211	3573	8	927	20	20	31	0.67	1.10
DIMACS	211	4132	8	765	18	18	27	0.64	1.36
DIMACS	256	12674	8	4888	28	28	43	0.65	43.69
DIMACS	385	19129	8	15810	-	27	94	-	-
DIMACS	500	7140	8	15074	-	25	1	-	-

Table 4.3: Results with the Branch-and-Price algorithm of the isolating-separator formulation on DIMACS instances.

Instances	n	m	k	Cols	Opt	Heur	No	Gap %	CPU
Random	50	511	7	552	18	18	27	0.64	**0.46**
Random	70	993	7	678	19	19	29	0.65	0.80
Random	100	1985	7	561	18	18	27	0.64	**0.61**
Random	300	17792	7	816	19	19	29	0.65	4.42
Random	600	70673	7	6624	19	19	29	0.65	285.23
Random	700	96432	7	1381	18	18	27	0.64	48.41
Random	800	125978	10	31073	-	44	41	-	-
Random	900	159331	10	28127	-	51	45	-	-
Random	1000	196771	10	24214	-	45	42	-	-
Random	1200	283386	10	15536	-	44	38	-	-
Random	1300	332861	10	18995	-	44	38	-	-
Random	1500	442444	10	9814	-	47	50	-	-
Random	1800	637307	15	7879	-	78	71	-	-
Random	2000	786639	15	7030	-	68	59	-	-
Random	2100	867411	15	6116	-	82	76	-	-
Random	2300	1041144	15	6720	-	70	65	-	-
Random	3000	1770773	15	4162	-	75	66	-	-
Random	3300	2142370	15	5828	-	80	73	-	-
Random	3800	2841805	15	4455	-	72	65	-	-
Random	4200	3472117	15	5045	-	81	74	-	-
Random	4500	3987339	15	5123	-	81	74	-	-
Random	4800	4537794	15	5237	-	74	67	-	-
Random	5000	4922710	15	5406	-	67	70	-	-

Table 4.4: Results with the Branch-and-Price algorithm of the isolating-separator formulation on random instances.

Instances	n	m	k	Cols	ST	TC	TT	Opt	Heur	No	Gap %	CPU
DIMACS	47	258	6	527	0	0	19	17	17	21	0.44	**0.33**
DIMACS	64	1477	6	288	0	0	18	16	16	1	0.06	**0.22**
DIMACS	74	624	6	257	49	0	49	13	14	6	0.06	**0.23**
DIMACS	80	533	6	708	99	0	118	16	16	13	0.10	**1.47**
DIMACS	87	835	6	417	0	0	52	14	14	5	0.13	**0.41**
DIMACS	95	778	6	733	0	2	34	18	18	1	0.05	1.56
DIMACS	96	2762	6	484	0	0	26	21	21	1	0.04	**0.84**
DIMACS	100	2967	8	387	0	0	14	22	22	1	0.04	**0.53**
DIMACS	120	1307	8	2687	0	5	43	24	24	1	0.04	48.29
DIMACS	125	764	8	1754	0	0	20	23	23	1	0.04	6.97
DIMACS	128	804	8	784	8	0	14	16	16	1	0.06	1.12
DIMACS	128	2370	8	934	0	0	85	22	24	2	0.00	3.21
DIMACS	128	4256	8	1240	0	2	25	24	24	1	0.03	3.96
DIMACS	128	6462	8	634	0	1	16	24	24	1	0.04	**1.69**
DIMACS	128	10426	8	568	0	0	21	24	24	1	0.04	1.60
DIMACS	138	1022	8	5665	360	0	381	21	26	147	0.05	125.09
DIMACS	144	5224	8	723	0	1	24	25	25	1	0.02	1.56
DIMACS	188	3920	8	653	0	2	39	28	28	1	0.03	**1.86**
DIMACS	191	2387	8	848	0	0	29	22	22	1	0.03	**1.20**
DIMACS	196	8399	8	362	0	10	19	23	23	1	0.04	**0.84**
DIMACS	197	3952	8	626	0	0	16	18	18	1	0.05	**1.28**
DIMACS	206	3576	8	749	0	0	19	21	21	1	0.04	2.28
DIMACS	211	3573	8	1423	0	0	15	20	20	3	0.09	5.19
DIMACS	211	4132	8	668	0	0	10	18	18	1	0.04	1.09
DIMACS	256	12674	8	1647	0	11	37	28	28	1	0.03	8.11
DIMACS	385	19129	8	910	0	4	20	27	27	1	0.03	5.13
DIMACS	500	7140	8	9102	0	4	13	-	25	1	-	-

Table 4.5: Results with the Branch-and-Cut-and-Price algorithm of the isolating-separator formulation on DIMACS instances.

Instances	n	m	k	Cols	ST	TC	TT	Opt	Heur	No	Gap %	CPU
Random	50	511	7	210	0	0	25	18	18	1	0.05	**0.28**
Random	70	993	7	510	0	2	18	19	19	1	0.04	0.93
Random	100	1985	7	341	0	3	13	18	18	1	0.05	**0.44**
Random	300	17792	7	695	0	10	16	19	19	3	0.08	3.74
Random	600	70673	7	1047	0	4	18	19	19	1	0.05	26.58
Random	700	96432	7	1580	0	8	20	18	18	1	0.04	23.71
Random	800	125978	10	1812	0	9	36	44	44	1	0.01	110.99
Random	900	159331	10	3106	0	11	83	51	51	1	0.01	382.08
Random	1000	196771	10	2856	0	9	40	45	45	1	0.02	417.03
Random	1200	283386	10	2314	0	10	36	44	44	1	0.01	395.60
Random	1300	332861	10	2207	0	12	46	44	44	1	0.02	494.92
Random	1500	442444	10	1916	0	18	68	47	47	1	0.01	535.64
Random	1800	637307	15	2211	0	20	73	78	78	1	0.01	1083.80
Random	2000	786639	15	2283	0	7	57	68	68	1	0.01	1105.77
Random	2100	867411	15	2423	0	12	75	82	82	1	0.01	1470.35
Random	2300	1041144	15	2681	0	27	81	70	70	1	0.01	1949.58
Random	3000	1770773	15	3393	0	29	76	75	75	1	0.01	3445.06
Random	3300	2142370	15	3775	0	26	90	80	80	1	0.01	4962.08
Random	3800	2841805	15	4074	0	27	71	72	72	1	0.01	5556.17
Random	4200	3472117	15	4640	0	12	121	81	81	1	0.01	11047.50
Random	4500	3987339	15	4928	0	26	90	81	81	1	0.01	10305.40
Random	4800	4537794	15	5007	0	17	66	74	74	1	0.01	8110.66
Random	5000	4922710	15	5268	0	18	69	67	67	1	0.01	9372.34

Table 4.6: Results with the Branch-and-Cut-and-Price algorithm of the isolating-separator formulation on random instances.

Instances	n	m	k	Cols	Opt	Heur	No	Gap %	CPU
DIMACS	47	258	6	291	17	17	23	0.55	**0.49**
DIMACS	64	1477	6	213	16	16	21	0.52	**0.22**
DIMACS	74	624	6	334	13	14	39	0.24	**0.26**
DIMACS	80	533	6	1014	16	16	91	0.39	**1.32**
DIMACS	87	835	6	459	14	14	27	0.40	**0.47**
DIMACS	95	778	6	332	18	18	25	0.57	**0.23**
DIMACS	96	2762	6	292	21	21	31	0.62	**0.37**
DIMACS	100	2967	8	500	22	22	33	0.63	**1.18**
DIMACS	120	1307	8	27991	24	24	125	0.55	8382.35
DIMACS	125	764	8	13013	23	23	117	0.64	161.05
DIMACS	128	804	8	3023	16	16	63	0.39	8.04
DIMACS	128	2370	8	3414	22	24	211	0.47	**15.81**
DIMACS	128	4256	8	1674	24	24	133	0.60	10.22
DIMACS	128	6462	8	702	24	24	45	0.60	5.01
DIMACS	128	10426	8	390	24	24	35	0.60	4.08
DIMACS	138	1022	8	32571	21	26	2077	0.20	937.50
DIMACS	144	5224	8	565	25	25	35	0.56	2.65
DIMACS	188	3920	8	550	28	28	41	0.60	**2.10**
DIMACS	191	2387	8	660	22	22	33	0.63	2.14
DIMACS	196	8399	8	719	23	23	37	0.70	5.78
DIMACS	197	3952	8	408	18	18	27	0.64	0.93
DIMACS	206	3576	8	457	21	21	29	0.56	**1.08**
DIMACS	211	3573	8	417	20	20	31	0.67	0.90
DIMACS	211	4132	8	368	18	18	27	0.64	0.89
DIMACS	256	12674	8	883	28	28	43	0.65	10.47
DIMACS	385	19129	8	1301	27	27	75	0.64	34.35
DIMACS	500	7140	8	22501	-	25	33	-	-

Table 4.7: Results with the Branch-and-Price algorithm of the Terminal-pair-separator formulation on DIMACS instances.

Instances	n	m	k	Cols	Opt	Heur	No	Gap %	CPU
Random	50	511	7	290	18	18	27	0.64	**0.30**
Random	70	993	7	296	19	19	29	0.65	**0.34**
Random	100	1985	7	319	18	18	27	0.64	**0.55**
Random	300	17792	7	530	19	19	29	0.65	7.08
Random	600	70673	7	792	19	19	29	0.65	25.46
Random	700	96432	7	879	18	18	27	0.64	44.17
Random	800	125978	10	1399	44	44	73	0.73	553.85
Random	900	159331	10	1696	51	51	87	0.76	2221.57
Random	1000	196771	10	1560	45	45	71	0.67	1623.42
Random	1200	283386	10	1916	44	44	73	0.73	2617.03
Random	1300	332861	10	1922	44	44	73	0.73	3625.82
Random	1500	442444	10	2133	47	47	79	0.74	5983.95
Random	1800	637307	15	2335	-	78	73	-	-
Random	2000	786639	15	2494	-	68	50	-	-
Random	2100	867411	15	2573	-	82	47	-	-
Random	2300	1041144	15	2754	-	70	13	-	-
Random	3000	1770773	15	3392	-	75	1	-	-
Random	3300	2142370	15	3654	-	80	1	-	-
Random	3800	2841805	15	4125	-	72	1	-	-
Random	4200	3472117	15	4510	-	81	1	-	-
Random	4500	3987339	15	4801	-	81	1	-	-
Random	4800	4537794	15	5074	-	74	1	-	-
Random	5000	4922710	15	5253	-	67	1	-	-

Table 4.8: Results with the Branch-and-Price algorithm of the Terminal-pair-separator formulation on random instances.

Instances	n	m	k	Cols	ST	TC	TT	Opt	Heur	No	Gap %	CPU
DIMACS	47	258	6	778	389	0	7	17	17	13	0.18	**0.76**
DIMACS	64	1477	6	279	12	0	12	16	16	1	0.00	**0.16**
DIMACS	74	624	6	229	65	0	55	13	14	7	0.04	**0.27**
DIMACS	80	533	6	295	80	0	59	16	16	3	0.09	0.45
DIMACS	87	835	6	280	63	0	33	14	14	1	0.04	**0.27**
DIMACS	95	778	6	306	30	0	30	18	18	1	0.00	**0.22**
DIMACS	96	2762	6	200	24	0	24	21	21	1	0.00	**0.25**
DIMACS	100	2967	8	320	19	0	22	22	22	1	0.00	**0.53**
DIMACS	120	1307	8	935	555	0	336	24	24	1	0.01	**7.29**
DIMACS	125	764	8	1259	174	0	66	23	23	1	0.00	3.40
DIMACS	128	804	8	342	54	0	47	22	24	1	0.00	**0.98**
DIMACS	128	2370	8	358	26	0	26	24	24	1	0.00	**1.50**
DIMACS	128	4256	8	311	17	0	17	24	24	1	0.00	**1.48**
DIMACS	128	6462	8	269	11	0	11	24	24	1	0.00	**1.71**
DIMACS	128	10426	8	234	11	0	11	24	24	1	0.00	1.21
DIMACS	138	1022	8	4897	1129	0	255	22	26	336	0.10	29.95
DIMACS	144	5224	8	420	12	0	12	25	25	1	0.00	**1.23**
DIMACS	188	3920	8	324	30	0	30	28	28	1	0.00	**0.88**
DIMACS	191	2387	8	388	17	1	17	22	22	1	0.00	**0.73**
DIMACS	196	8399	8	343	17	0	17	23	23	1	0.00	1.56
DIMACS	197	3952	8	303	16	0	16	18	18	1	0.00	**0.54**
DIMACS	206	3576	8	316	19	0	19	21	21	1	0.00	0.65
DIMACS	211	3573	8	508	12	0	12	20	20	1	0.00	0.75
DIMACS	211	4132	8	297	8	0	8	18	18	1	0.00	0.35
DIMACS	256	12674	8	520	15	0	15	28	28	1	0.00	2.98
DIMACS	385	19129	8	595	13	0	13	27	27	1	0.00	5.74
DIMACS	500	7140	8	976	69	1	69	25	25	1	0.00	**8.80**

Table 4.9: Results with the Branch-and-Cut-and-Price algorithm of the Terminal-pair-separator formulation on DIMACS instances.

Instances	n	m	k	Cols	ST	TC	TT	Opt	Heur	No	Gap %	CPU
Random	50	511	7	275	26	0	26	18	18	1	0.00	**0.23**
Random	70	993	7	214	17	0	17	19	19	1	0.00	**0.22**
Random	100	1985	7	224	10	0	10	18	18	1	0.00	**0.21**
Random	300	17792	7	445	9	0	9	19	19	1	0.00	2.74
Random	600	70673	7	786	8	0	8	19	19	1	0.00	13.08
Random	700	96432	7	805	8	0	8	18	18	1	0.00	18.22
Random	800	125978	10	1041	21	0	21	44	44	1	0.00	130.12
Random	900	159331	10	1153	29	0	29	51	51	1	0.00	334.00
Random	1000	196771	10	1273	22	0	22	45	45	1	0.00	336.67
Random	1200	283386	10	1465	18	0	18	44	44	1	0.00	709.12
Random	1300	332861	10	1544	19	0	19	44	44	1	0.00	882.31
Random	1500	442444	10	1731	30	0	31	47	47	1	0.00	1282.20
Random	1800	637307	15	2335	41	0	41	78	78	1	0.00	8719.97
Random	2000	786639	15	2585	27	0	27	68	68	1	0.00	11107.50
Random	2100	867411	15	2663	40	0	40	82	82	1	0.00	11693.40
Random	2300	1041144	15	2718	0	0	0	-	70	1	-	-
Random	3000	1770773	15	3335	0	0	0	-	75	1	-	-
Random	3300	2142370	15	3555	0	0	0	-	80	1	-	-
Random	3800	2841805	15	4034	0	0	0	-	72	1	-	-
Random	4200	3472117	15	4421	0	0	0	-	81	1	-	-
Random	4500	3987339	15	4713	0	0	0	-	81	1	-	-
Random	4800	4537794	15	5005	0	0	0	-	74	1	-	-
Random	5000	4922710	15	5195	0	0	0	-	67	1	-	-

Table 4.10: Results with the Branch-and-Cut-and-Price algorithm of the Terminal-pair-separator formulation on random instances.

Instances	n	m	k	Natural formulation No	Gap %	CPU	Isolating-separator No	Gap %	CPU	Terminal-pair-separator No	Gap %	CPU
DIMACS	47	258	6	7	35.20	7.96	21	0.44	0.33	13	0.18	0.76
DIMACS	64	1477	6	1	0.00	0.99	1	0.06	0.22	1	0.00	0.16
DIMACS	74	624	6	23	19.20	3.39	6	0.06	0.23	7	0.04	0.27
DIMACS	80	533	6	47	28.10	6.94	13	0.10	1.47	3	0.09	0.45
DIMACS	87	835	6	26	28.50	8.58	5	0.13	0.41	1	0.04	0.27
DIMACS	95	778	6	1	0.00	1.26	1	0.05	1.56	1	0.00	0.22
DIMACS	96	2762	6	1	0.00	1.45	1	0.04	0.84	1	0.00	0.25
DIMACS	100	2967	8	1	0.00	1.55	1	0.04	0.53	1	0.00	0.53
DIMACS	120	1307	8	9	35.40	16.50	1	0.04	48.29	1	0.01	7.29
DIMACS	125	764	8	1	0.00	2.33	1	0.04	6.97	1	0.00	3.40
DIMACS	128	804	8	29	28.10	5.75	1	0.06	1.12	1	0.00	0.98
DIMACS	128	2370	8	6	31.80	16.44	2	0.00	3.21	1	0.00	1.50
DIMACS	128	4256	8	1	0.00	3.62	1	0.03	3.96	1	0.00	1.48
DIMACS	128	6462	8	1	0.00	2.03	1	0.04	1.69	1	0.00	1.71
DIMACS	138	1022	8	74	16.60	17.99	147	0.05	125.09	336	0.10	29.95
DIMACS	144	5224	8	1	0.00	1.44	1	0.02	1.56	1	0.00	1.23
DIMACS	188	3920	8	1	0.00	2.69	1	0.03	1.86	1	0.00	0.88
DIMACS	191	2387	8	1	0.00	1.34	1	0.03	1.20	1	0.00	0.73
DIMACS	196	8399	8	1	0.00	1.24	1	0.04	0.84	1	0.00	1.56
DIMACS	197	3952	8	1	0.00	0.58	1	0.05	1.28	1	0.00	0.54
DIMACS	206	3576	8	1	0.00	2.03	1	0.04	2.28	1	0.00	0.65
DIMACS	256	12674	8	1	0.00	2.10	1	0.03	8.11	1	0.00	2.98
DIMACS	385	19129	8	1	0.00	1.08	1	0.03	5.13	1	0.00	5.74
DIMACS	500	7140	8	4	40.00	32.35	-	-	-	1	0.00	8.80
Random	50	511	7	1	0.00	1.24	1	0.05	0.28	1	0.00	0.23
Random	70	993	7	1	0.00	0.64	1	0.04	0.93	1	0.00	**0.22**
Random	100	1985	7	1	0.00	1.02	1	0.05	0.44	1	0.00	**0.21**
Random	300	17792	7	1	0.00	2.31	3	0.08	3.74	1	0.00	2.74
Random	600	70673	7	1	0.00	4.04	1	0.05	26.58	1	0.00	13.08
Random	700	96432	7	1	0.00	3.96	1	0.04	23.71	1	0.00	18.22
Random	800	125978	10	1	0.00	20.98	1	0.01	110.99	1	0.00	130.12
Random	900	159331	10	1	0.00	31.31	1	0.01	382.08	1	0.00	334.00
Random	1000	196771	10	1	0.00	34.35	1	0.02	417.03	1	0.00	336.67
Random	1200	283386	10	1	0.00	41.01	1	0.01	395.60	1	0.00	709.12
Random	1300	332861	10	1	0.00	56.91	1	0.02	494.92	1	0.00	882.31
Random	1500	442444	10	1	0.00	92.64	1	0.01	535.64	1	0.00	1282.20
Random	1800	637307	15	1	0.00	379.95	1	0.01	1083.80	1	0.00	8719.97
Random	2000	786639	15	1	0.00	468.78	1	0.01	1105.77	1	0.00	11107.50
Random	2100	867430	15	1	0.00	641.02	1	0.01	1470.35	1	0.00	11693.40
Random	2300	1041158	15	1	0.00	851.49	1	0.01	1949.58	-	-	-
Random	3000	1770773	15	1	0.00	1698.90	1	0.01	3445.06	-	-	-
Random	3300	2142370	15	1	0.00	2129.60	1	0.01	4962.08	-	-	-
Random	3800	2841805	15	1	0.00	2363.80	1	0.01	5556.17	-	-	-
Random	4200	3472117	15	1	0.00	3856.30	1	0.01	11047.50	-	-	-
Random	4500	3987339	15	1	0.00	5248.10	1	0.01	10305.40	-	-	-
Random	4800	4537794	15	1	0.00	4611.80	1	0.01	8110.66	-	-	-
Random	5000	4922710	15	1	0.00	4291.40	1	0.01	9372.34	-	-	-

Table 4.11: Comparison table of the Branch-and-Cut algorithm with the Branch-and-Cut-and-Price algorithms.

Tables 4.3, 4.5, 4.7 and 4.9 present results obtained for DIMACS instances. These

instances are induced by graphs associated with graphs having up to 500 nodes, and whose the number of terminals is fixed to 6 for graphs with less than 99 nodes and 8 for the others. However, Tables 4.4, 4.6, 4.8 and 4.10 concern the random instances which are induced by graphs having up to 5000 nodes, and whose the number of terminals is fixed to 7, 10 and 15 for graphs with less than 700 nodes, between 800 and 1500 nodes and between 1800 and 5000 nodes, respectively. The values given in bold, represent the instances for which the CPU time is better than the one of the Branch-and-Cut algorithm presented in Chapter 2. The symbol "-" in the tables means that the corresponding instance has not been solved in the time limit. We can remark that in some instances, the Gap is different from 0.00 whereas the instance is solved in the root node of the branching tree. This can be explained by the fact that the difference between the value of the linear relaxation and the value given by the heuristic is strictly lower than 1. In this case, the solution given by the heuristic is optimal.

Tables 4.1 and 4.2 give some numerical results obtained by the Branch-and-Price algorithm of the terminal-set formulation. As it appears from the tables, the algorithm could not solve most of the instances in the time limit of 4 hours. Only instances of less than 100 nodes, could be solved and the CPU time is high. We also notice that the number of generated columns is high, more than 1500 in most of the instances and for the instance of 87 nodes it reached 500703. Moreover, we can note that the Branch-and-Price algorithm does not start solving the instances with more than 1500 nodes. This is due to the high number of constraints which is related to the number of edges (more than 442444 edges for the instances with more than 1500 nodes).

Results obtained by the Branch-and-Price algorithm of the isolating-separator formulation are shown in Tables 4.3 and 4.4. Here the algorithm could solve 85% of the DIMACS instances in the time limit and 77% of the instances in less than 6 minutes. Moreover, the algorithm could not solve the instances with more than 800 nodes and 10 terminals. The Gap is around 60% in most of instances. This is may be explained by the fact that the linear relaxation is weak. This can be also seen from the number of nodes in the branching tree which is not less than 21 and which increases with the size of the instance. Despite this, the algorithm could solve 11 instances over 50, in shorter time, comparing with the Branch-and-Cut algorithm presented in Chapter 2.

Tables 4.5 and 4.6 concern results obtained by the Branch-and-Cut-and-Price algorithm based on the isolating-separator formulation. As it appears the algorithm could solve 96% of the DIMACS instances in less than 2 minutes and all the random instances in less than 3 hours. The Gap is less than 6% in most of instances, which implies that here the linear relaxation is quite strong. This is also clear from the fact that instances are solved in the root node of the branching tree. Moreover, we note that the algo-

rithm solved 14 instances over 50, in shorter time, comparing with the Branch-and-Cut algorithm presented in Chapter 2. Concerning the additional valid inequalities used in the algorithm, as it can be seen, the algorithm has not generated star tree inequalities for most of the instances. However, the terminal tree inequalities, which are present in all the instances, appear useful in the resolution of the problem. Comparing with the Branch-and-Price algorithm of the isolating-separator formulation, we can notice that the added inequalities, in particular the terminal tree inequalities, have much improved the linear relaxation. This also permitted to decrease the number of generated columns which does not exceed 9102 for all the instances. Moreover, the Branch-and-Cut-and-Price algorithm was more efficient than the Branch-and-Price algorithm. Indeed, we can notice that the Branch-and-Price algorithm could not solve neither the DIMACS instances with 120, 138 and 385 nodes nor the random instances with more than 800 nodes, within the limit time. Whereas, the Branch-and-Cut-and-Price algorithm solved all of them.

In Tables 4.7 and 4.8 we give the results obtained by the Branch-and-Price algorithm based on the terminal-pair-separator formulation. Here the algorithm was able to solve around 96% of DIMACS instances within the time limit of 4 hours, and around 88% in less than 3 minutes. However, the Gap is high, around 60% for most of the instances. Thus it also appears that the linear relaxation of the terminal-pair-separator formulation is not sufficiently strong. Comparing the algorithm with the Branch-and-Cut algorithm presented in Chapter 2, we can remark that 14 instances over 50 have been solved in shorter time.

Tables 4.9 and 4.10 are related to the Branch-and-Cut-and-Price algorithm based on the terminal-pair-separator formulation. As it appears from the tables the algorithm could solve all the DIMACS instances in less than 30 seconds. However, it could not solve the random instances with more than 2300 nodes and 15 terminals, within the time limit. Most of the instances have been solved in the root node, which means that the linear relaxation here is sufficiently strong. It also appears from the tables that the cycle inequalities have not almost played any role in resolution of the problem. Comparing with the terminal tree and the star tree inequalities which have been generated and used for all the instances. Comparing with the Branch-and-Cut algorithm of Chapter 2, we notices that 20 instances over 50 have been solved in less time. And comparing with the Branch-and-Price algorithm of the terminal-pair-separator formulation, it clearly appears that the added inequalities have been efficient and permitted to reduce a lot both the CPU time and the number of generated columns. Indeed, the Branch-and-Price algorithm could not solve neither the DIMACS instances with 500 nodes nor the random instances with 1800, 2000 and 2100 nodes, within the limit time. Whereas, the Branch-and-Cut-and-Price algorithm solved all of them.

The largest DIMACS instance solved by the Branch-and-Price algorithm of the terminal-set, the isolating-separator and the terminal-pair-separator formulations has 96, 256 and 385 nodes, respectively. Moreover, the Branch-and-Price algorithm of the isolating-separator formulation solved a random instance of 700 nodes in 48.41 seconds while the one of the terminal-set formulation solved a random instance of 70 nodes in 3629.78 seconds. We notice that the Branch-and-Cut-and-Price algorithm of the isolating-separator formulation could not solve one DIMACS instance of 500 nodes while it solved all random instances. On the other hand, the Branch-and-Cut-and-Price algorithm of the terminal-pair-separator formulation, solved all DIMACS instances while it could not solve the random instances with more than 2300 nodes.

In Table 4.11 we summarize the numerical results of the Branch-and-Cut algorithm related to the natural formulation, presented in Chapter 3, and the Branch-and-Cut-and-Price algorithms of the isolating-separator and the terminal-pair-separator formulations. For each instance, the underlined value represents the shortest CPU time among the three algorithms. As it appears from the table, the Branch-and-Cut algorithm solved all instances while the Branch-and-Cut-and-Price algorithm of the terminal-pair-separator formulation could not solve the instances with more than 2300 nodes. Moreover, among the three algorithms, the Branch-and-Cut algorithm was the fastest for solving the problem on 48% of the instances. Whereas, the Branch-and-Cut-and-Price algorithm of the terminal-pair-separator formulation was the fastest on only 39% of them.

4.6 Conclusion

In this chapter we have proposed three extended integer programming formulations for the MTVSP. For each of the formulations, we have developed a Branch-and-Price algorithm and presented extensive computational results. And for two of them, we added three families of valid inequalities presented in Chapter 2, to develop Branch-and-Cut-and-Price algorithms. For all the formulations, we have shown how to compute the dual bound during the column generation phase and analyzed the complexity of the pricing problem, the branching scheme and the strategy of the column generation. The computational results have permitted to measure the importance of our approaches.

Chapter 5

The variants of the multi-terminal vertex separator problem

Contents

In this chapter we discuss four variants of the MTVS problem. For each variant, we show that the problem is NP-hard, then we give an integer programming formulation for the problem and describe several valid inequalities.

In this chapter, we suppose that Hypotheses 2.1-2.5, given in Chapter 2, are satisfied.

5.1 The connected components separator problem

In this section we discuss a variant of the MTVSP, called *the connected components separator problem* (CCSP). Given a graph $G = (V \cup T, E)$ and a weight function $w : V \to \mathbb{Z}$, the CCS problem consists in finding a subset of vertices $S \subseteq V$ of minimum weight such that $G \setminus S$ has $|T|$ connected components, each one contains exactly one terminal. Set S is called a *connected components separator*.

Theorem 5.1 *The CCSP is NP-hard.*

Proof. This proof is based on a polynomial reduction from the vertex cover problem (VCP). Given a graph $H = (U, E')$, the VCP consists in finding a minimum cardinality subset of vertices $R \subseteq U$ such that each edge of E' is incident to at least one vertex of R. The VCP is a well-known NP-hard problem [60].

Consider the vertex cover problem on a graph $H = (U, E')$. Let $G = (V_1 \cup V_2 \cup V_3 \cup T, E)$ be the graph obtained from H as follows

- Add three vertices t_1, t_2 and t_3 in T.
- For each vertex $u \in U$, add
  - three vertices v_1^u in V_1, v_2^u in V_2 and v_3^u in V_3,
  - three edges $t_1v_1^u$, $t_2v_2^u$ and $t_3v_3^u$ in E,
  - two edges $v_1^uv_3^u$ and $v_2^uv_3^u$ in E.
- For each edge $uw \in E'$, add two edges $v_1^uv_2^w$ and $v_1^wv_2^u$ in E.

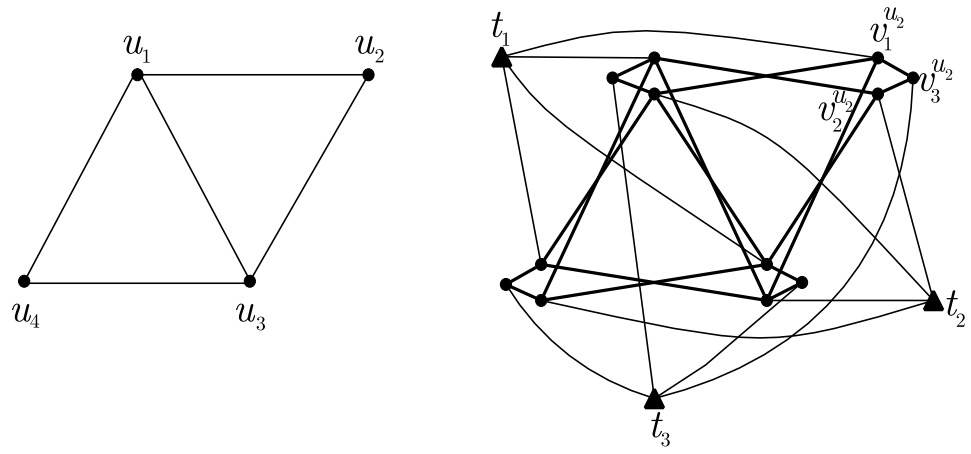


Figure 5.1: Graph transformation from the graph H to the graph G.

Figure 5.1 illustrates the above graph transformation.

Let R be a vertex cover in H and S a separator in G. Let $R^S \subseteq U$ be the set of vertices such that for all $u \in R^S$, $|S \cap \{v_1^u, v_2^u, v_3^u\}| \geq 2$ and let $S^R \subseteq V_1 \cup V_2 \cup V_3$ be the set of vertices defined as follows. For each vertex $u \in U$, if $u \in R$ then we add the two vertices v_1^u, v_2^u in S^R, otherwise we add v_3^u in S^R. In Section 2.1 of Chapter 2, we showed that if R is a vertex cover in H of minimum size, then S^R is a multi-terminal vertex separator in G of minimum size. And, if S is a multi-terminal vertex separator in G of minimum size, then R^S is a vertex cover in H of minimum size. Since each non-terminal vertex in G is connected to a terminal, any multi-terminal vertex separator in G is a connected components separator. Consequently, since the vertex cover is NP-hard, the connected components separator problem is also NP-hard. ■

5.1.1 Formulation

A *free connected component* $W \subseteq V$ is a subset of vertices such that $G[W]$ is connected and W contains no terminal. For a vertex $v \in V$, a set $S^v \subset V$ is called an *isolating separator* of v, if the graph $(V \setminus S^v, E)$ is not connected, and v belongs to a free connected component. Let $\mathcal{S}^v$ be the set of all minimal isolating separators of v. An isolating separator S^v is minimal if it contains no isolating separator of v.

Consider the following system

$$x(P_{tt'}) \geq 1, \qquad \forall P_{tt'} \in \Gamma, \tag{5.1}$$

$$x(S^u) \leq |S^u| - 1 + x(u), \qquad \forall u \in \overline{V}, \forall S^u \in \mathcal{S}^u, \tag{5.2}$$

$$x(v) \leq 1, \qquad \forall v \in V, \tag{5.3}$$

$$x(v) \geq 0, \qquad \forall v \in V, \tag{5.4}$$

$$x(v) \text{ integer.} \qquad \forall v \in V \tag{5.5}$$

where $\overline{V} = V \setminus N(T)$ and Γ is the set of all terminal paths of G .(Recall that a terminal path is a path between two terminals).

Inequalities (5.1) ensure that all terminal paths intersect the vertices of the solution. Inequalities (5.2) guarantee that each component of G, after deleting the vertices of the solution, contains a terminal.

Theorem 5.2 *Every connected components separator satisfies inequalities (5.2).*

Proof. Let us assume the contrary, that is there exists a connected components separator S, a vertex $u \in \overline{V}$ and an isolating separator $S^u \in \mathcal{S}^u$ such that

$$x^S(S^u) > |S^u| - 1 + x^S(u).$$

It follows that $x^S(u) = 0$ and $x^S(S^u) = |S^u|$. Therefore, $u \notin S$ and $S^u \subseteq S$. Since S^u is an isolating set of u, it follows that u belongs to a free connected component of $G \setminus S^u$. As $S^u \subseteq S$, then u belongs to a free connected component of $G \setminus S$. But this contradicts the fact that S is a connected components separator. ■

From inequalities (5.1), all solutions of (5.1) − (5.5) are MTV separators. The following theorem shows that these MTV separators are also connected components separators.

Theorem 5.3 *Every multi-terminal vertex separator of G satisfying inequalities (5.2) is a connected components separator.*

Proof. Suppose there exists a multi-terminal vertex separator S of G satisfying inequalities (5.2) such that S is not a connected components separator of G. It then follows that there exists a vertex u that belongs to a free connected component of

$G \setminus S$. Thus S is an isolating separator of u. However, as $x^S(u) = 0$, the following inequality (5.2)

$$x(S) \leq |S| - 1 + x(u),$$

is violated by x^S, contradicting the fact that x^S satisfies all inequalities (5.2). ■

From Theorems 5.2 and 5.3, we have the following corollary

Corollary 5.4 *The connected components separator problem is equivalent to the following integer linear program*

$$\min\{ \sum_{v \in V} w(v)x(v) \mid x \text{ satisfies } (5.1) - (5.5)\}.$$

5.1.2 Polyhedral analysis

Let $P(G,T)$ be the convex hull of the solutions of $(5.1) - (5.5)$, that is,

$$P(G,T) = conv\{x \in \{0,1\}^V | \ \ x \text{ satisfies } (5.1) - (5.2)\}.$$

In this subsection, we will discuss $P(G,T)$, we will give its dimension, and describe necessary and sufficient conditions for inequalities (5.2) to be facet defining.

We first establish the dimension of $P(G,T)$.

Theorem 5.5 *$P(G,T)$ is full dimensional.*

Proof. We need to exhibit $n+1$ connected components separators such that their incidence vectors are affinely independent. Let $S_0 = V$. Clearly, $G \setminus S_0$ has k components, each one contains one terminal. Thus S_0 is a connected components separator of G. Given a vertex $v \in V$, let P^v be the shortest path (in terms of number of edges) between v and T (the shortest path among all shortest paths between v and each terminal of T). Let $S_v = V \setminus P^v$.

Claim 1. Set S_v is a multi-terminal vertex separator of G.

Proof. Suppose that S_v is not a multi-terminal vertex separator of G. It follows that $G \setminus S_v$ has a terminal path $P_{tt'}$ between t and t', not intersecting S_v. From the definition of S_v, all internal vertices of $P_{tt'}$ are included in P^v. From Hypothesis 2.2 given in Chapter 2, each vertex of V is adjacent to at most one terminal. It follows that P^v has an internal vertex adjacent to a terminal, and this contradicts the fact that P^v is the shortest path between v and T. □

Claim 2. Set S_v is a connected components separator of G.

Proof. From Claim 1 set S_v is a multi-terminal vertex separator of G. Clearly $G \setminus S_v$ has $k-1$ components with exactly one terminal and one components with the path P^v. It follows that S_v is a connected components separator of G. □

From Claim 2 for all $v \in V$, S_v is a connected components separator of G. This constitutes a set of $n+1$ connected components separators of G. Moreover, their incidence vectors are clearly affinely independent. ■

Consider a graph $G = (V \cup T, E)$ and an isolating separator S^u of vertex u in G. A *free dominating set* $D \subseteq S^u$ associated with S^u is a subset of vertices such that there is edges between D and all free connected components of $G \setminus S^u$, and an edge between D and a connected component of $G \setminus S^u$ containing a terminal. Let r_{S^u} be the size of the minimum free dominating set associated with S^u. r_{S^u} is called the *power* of S^u. Let $\mathcal{W}^{S^u}$ be the set of all vertices belonging to free connected components in $G \setminus S^u$. For all $v \in V$, let W_v be the free connected component of $G \setminus S^u$ containing v.

Consider a vertex $u \in \overline{V}$ and an isolating separator $S^u \subseteq V$ of u.

Lemma 5.6 *If S^u has a power greater than or equal to 2 than for all $s \in S^u$, there exists a free connected component W of $G \setminus S^u$ different from W_u such that s is not adjacent either to W or to W_u.*

Proof. Assume the contrary, there exists $s \in S^u$ such that for all free connected component W of $G \setminus S^u$, s is adjacent to W_u and to W. Clearly $\{s\}$ is a free dominating set of S^u. It follows that the power of S^u is 1 which contradicts the fact that $r_{S^u} \geq 2$. ■

Lemma 5.7 *If $r_{S^u} \geq 2$ then the following inequality*

$$x(S^u) \leq |S^u| - 2 + x(u) + x(v)$$

is valid for $P(G,T)$, where u and $v \in V$ belong to two disjoint free connected components of $G \setminus S^u$.

Proof. Let us assume the contrary that there exists a connected components separator S of G and an isolating separator S^u with $r_{S^u} \geq 2$ such that

$$x^S(S^u) \geq |S^u| - 1 + x^S(u) + x^S(v)$$

- If $x^S(S^u) = |S^u|$, then $S^u \subseteq S$ and either $u \notin S$ or $v \notin S$. Clearly $G \setminus S$ has at least one free connected component, contradicting the fact that S is a connected components separator of G.
- If $x^S(S^u) = |S^u| - 1$, then $S^u \setminus S = \{s\}$ and $u, v \notin S$. Since $r_{S^u} \geq 2$, by Lemma 5.6 s is not adjacent either to W_u or to W_v. Hence $G \setminus S^u$ has at least one free connected component, contradicting the fact that S is a connected components separator of G. ■

Theorem 5.8 *Given a vertex $u \in \overline{V}$ and an isolating separator S^u of u, inequality (5.2) associated with S^u and u defines a facet for $P(G,T)$ if and only if*

1- the power of S^u is 1,

2- the isolating separator S^u is minimal, and

3- no vertex of $V \setminus S^u$ is an isolating separator of u.

Proof.
($\Rightarrow$)

1- If the power of S^u is at least 2, then $G \setminus S^u$ has at least two free connected components, say W_u and W_w containing vertices u and w, respectively. By Lemma 5.7, the following inequality

$$x(S^u) \leq |S^u| - 2 + x(u) + x(w)$$

is valid for $P(G,T)$. Inequality (5.2) associated with S^u and vertex u can then be obtained by summing the above inequality and $x(w) \leq 1$. Hence (5.2) cannot define a facet.

2- If S^u is not minimal, then there exists a vertex $s \in S^u$ such that $S' = S^u \setminus \{s\}$ is an isolating separator of u. Thus, inequality

$$x(S') \leq |S'| - 1 + x(u)$$

is valid for $P(G,T)$. Inequality (5.2) associated with S^u and vertex u can be obtained from the above inequality and $x(s) \leq 1$.

3- If there exists a vertex $w \in V \setminus S^u$ that is an isolating separator of u, then we have the following inequality (5.2) associated with w

$$x(w) \leq x(u).$$

Moreover, we notice that S^u must be an isolating separator of w. Thus, we have the following inequality (5.2) associated with S^u and vertex w,

$$x(S^u) \leq |S^u| - 1 + x(w).$$

Inequality (5.2) associated with S^u and vertex u can be obtained by summing the two above inequalities.

($\Leftarrow$)
Denote by $ax \geq \alpha$ inequality (5.2) associated with S^u and vertex u. Let $bx \geq \beta$ be an inequality that defines a facet of $P(G,T)$. Suppose that $\{x \in P(G,T) : ax = \alpha\} \subseteq \{x \in P(G,T) : bx = \beta\}$. We will show that there exists ρ such that $b = \rho a$.

For $v \in V \setminus (S^u \cup \mathcal{W}^{S^u})$, let P_v be the shortest path (in terms of number of edges) in $G \setminus (S^u \cup \mathcal{W}^{S^u})$ between v and a terminal in T. This path exists, otherwise $v \in \mathcal{W}^{S^u}$. Let $S^{P_v} = V \setminus P_v$. Since P_v is the shortest path and no vertex is adjacent to two terminals, it follows that S^{P_v} is a connected components separator of G. Moreover, $ax^{S^{P_v}} = \alpha$. Hence, $bx^{S^{P_v}} = bx^{S^{P_w}} = bx^V$ for every $w \in P_v$. Therefore,

$$b_w = 0, \quad \forall w \in P_v,$$

Since v is arbitrary chosen in $V \setminus (S^u \cup \mathcal{W}^{S^u})$, if follows that

$$b_w = 0 \quad \forall w \in V \setminus (S^u \cup \mathcal{W}^{S^u}),$$

Let D be the minimum free dominating set associated with S^u. From Condition 1, $D = \{s\}$. Let P_v^s be the shortest path (in terms of number of edges) from $v \in \mathcal{W}^{S^u}$ and s, in $G[W_v \cup \{s\}]$. For $w \in S^u$, let $\widetilde{P}^w$ be the path between $w \in S^u$ and a terminal

in $G \setminus (\mathcal{W}^{S^u} \cup (S^u \setminus \{w\}))$. From Condition 2, this path exists. For all $v \in \mathcal{W}^{S^u} \setminus W_u$, let $S_v = V \setminus (P_v^s \cup P_u^s \cup \widetilde{P}^s)$ and $S_u = V \setminus (P_u^s \cup \widetilde{P}^s)$. It is clear that sets S_v and S_u are two separators of G. Moreover, $ax^{S_v} = ax^{S_u} = \alpha$. Hence, $bx^{S_v} = bx^{S_u}$ for every $v \in \mathcal{W}^{S^u} \setminus W_u$. Consequently,

$$b_v = 0 \qquad \forall v \in \mathcal{W}^{S^u} \setminus W_u.$$

For $v \in W_u \setminus \{u\}$, let $\overline{P}_v^s$ be the shortest path between v and s containing u in $G[W_u \cup \{s\}]$. By Condition 3, such a path exists. Let $S^1 = V \setminus (\overline{P}_v^s \cup \widetilde{P}^s)$ and $S^2 = V \setminus (P_u^s \cup \widetilde{P}^s)$. Sets S^1 and S^2 are two connected components separators of G. Moreover, $ax^{S^1} = ax^{S^2} = \alpha$. Hence, $bx^{S^1} = bx^{S^2}$ for every $v \in W_u \setminus \{u\}$. Therefore,

$$b_v = 0 \qquad \forall v \in W_u \setminus \{u\}.$$

For $v \in S^u$, let $S'_v = V \setminus (\widetilde{P}^v \cup W_u)$. Set S'_v is a connected components separator of G. Moreover, $ax^{S'_v} = \alpha$. Hence, $bx^{S'_v} = bx^V$, yielding

$$b_v = -b_u = \rho \qquad \forall v \in S^u,$$

which ends the proof. ■

Theorem 5.9 *Inequalities* (5.2) *can be separated in polynomial time.*

Proof.
Let $y \in \{0,1\}^V$ such that $y_v = 1 - x(v)$ for all $v \in V$. Inequalities (5.2) are equivalent to

$$\sum_{v \in S} y_v \quad \geq \quad y_u \quad S \in \mathcal{S}^u, \forall u \in \overline{V}. \tag{5.6}$$

Consider a vertex $v \in \overline{V}$ and let $x^* \in [0,1]^V$. Let $y^* \in [0,1]^V$ such that $y_u^* = 1 - x_u^*$ for all $u \in V$. Let G_v be the graph obtained from G by merging all terminals into a super terminal t_s and considering v as a terminal. Now, we look for a minimum multi-terminal vertex separator S^v between t_s and v in G_v w.r.t y^*. Note that this can be done in polynomial time. If inequality (5.6) associated with S^v is violated by y^*, then we add the inequality. By iterating over all vertices in $\overline{V}$, the separation of the isolating separator inequality can be performed in polynomial time. ■

5.2 The multi-terminal connected separator problem

In this section we discuss a variant of the MTVSP, called *the multi-terminal connected separator problem* (MTCSP). Given a graph $G = (V \cup T, E)$ and a weight function $\omega : V \to \mathbb{Z}$, the MTCS problem consists in finding a subset of vertices $S \subseteq V$ of minimum weight such that $G \setminus S$ has $|T|$ components (not necessarily connected), each one contains exactly one terminal and $G[S]$ is connected. Set S is called a *connected separator*.

Theorem 5.10 *The MTCSP is NP-hard.*

Proof. This proof is based on a polynomial reduction from the node-weighted steiner tree problem, which is known to be NP-hard [49]. This problem consists, given a graph $G = (V \cup T, E)$ and a weight function $\omega : V \to \mathbb{Z}$, in finding a subset of vertices $S \subseteq (V \cup T)$ of a minimum weight such that $T \subseteq S$ and $G[S]$ is connected. S is called a *node-weighted steiner tree*.

Consider a graph $H = (V' \cup T', E')$ and let $G = (V \cup T, E)$ be the graph obtained from H as follows

- For each terminal $u \in V'$, add a vertex v_u to V of weight $w(u)$.
- For each terminal $t \in T'$, add a vertex v_t to V of weight 0.
- $E = E'$.
- For $t \in T'$, add two terminals t_1 and t_2 to T adjacent to v_t.

Figure 5.2 illustrates the above graph transformation, such that the graph Figure 5.2.(a) represents the graph H and Figure 5.2.(b) the graph G.

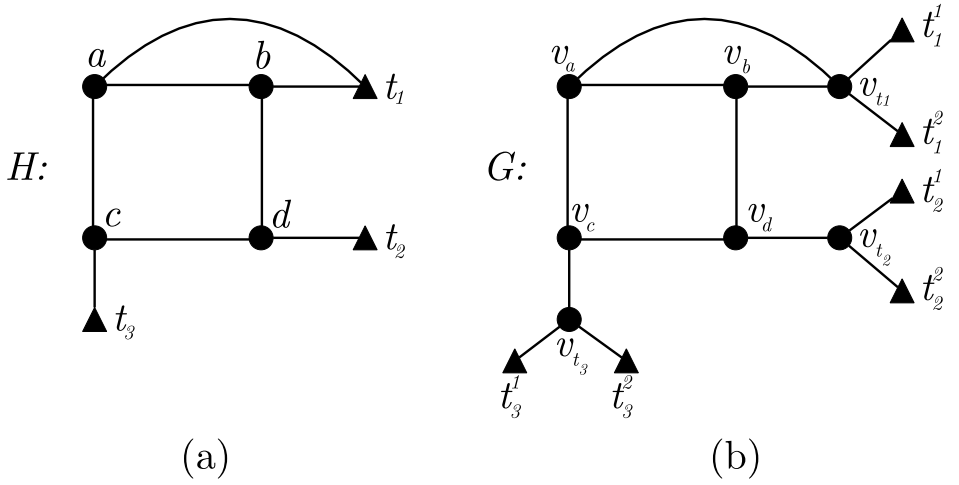


Figure 5.2: Graph transformation

Claim 1. Any multi-terminal connected separator of G is a node-weighted steiner tree in H.

Proof. We note that each vertex in G associated with a terminal of H is adjacent to two terminals. It follows that these vertices belong to all connected separators of G. Consider a connected separator $S \subseteq V$ of G and let $S' \subseteq (V' \cup T')$ be the set of the vertices in H associated with the vertices of S. Set S' contains all terminals of T'. Moreover, since $G[S]$ is connected, it follows that $H[S']$ is also connected and contains all terminals of T'. Consequently, S' is a node-weighted steiner tree in H. □

Claim 2. Any node-weighted steiner tree in H is a connected separator of G.

Proof. Let $S' \subseteq (V' \cup T')$ be a node-weighted steiner tree in H. Let $S \subseteq V$ be the set of the vertices in G associated with the vertices of S'. Since $H[S']$ is connected, it follows that $G[S]$ is connected. Moreover, since S' contains all terminals of T', $N(T) \subseteq S$. Consequently, S is a connected separator of G. □

By Claims 1 and 2 looking for a node-weighted steiner tree of minimum weight in H is equivalent to looking for a connected separator of minimum weight of G. Since the node-weighted steiner tree is NP-hard, it follows that the multi-terminal connected separator problem is also NP-hard. ■

Consider a graph $G = (V \cup T, E)$. For two vertices $u, v \in V$, a subset $S_v^u \subseteq V$ is

called a *disconnecting subset* of u and v if there is no path between u and v in $G \setminus S_v^u$. For two non-adjacent vertices $u, v \in V$, let $\mathcal{S}_v^u$ be the set of all disconnecting subsets of u and v.

In the rest of this section we consider the following Hypotheses

5.5.1- For all $v \in V \cup T$, each component of $G \setminus \{v\}$ contains a terminal. Otherwise, the component without terminals can be deleted from the graph.

5.5.2- Graph (V, E) (without terminals) is connected.

5.5.3- For all vertex cut $v \in V$, $G \setminus \{v\}$ has exactly two connected components. Otherwise, by Hypothesis 5.5.1, v must belong to the connected separator.

5.5.4- For all vertex cut $v \in V$ of G, at least one connected component of $G \setminus \{v\}$, has at most one terminal. Otherwise, by Hypotheses 5.5.1 and 5.5.3, v must belong to the connected separator.

5.2.1 Formulation

In what follows we give a formulation for the MTCS problem. For $v \in V$, let $x(v)$ be a variable which takes 1 if v belongs to the connected separator and 0 if not.

The multi-terminal connected separator problem is equivalent to the following integer linear program

$$\min \sum_{v \in V} \omega(v) x(v)$$

$$x(P_{tt'}) \geq 1 \qquad \forall P_{tt'} \in \Gamma, \tag{5.7}$$

$$x(S) \geq x(u) + x(v) - 1 \quad \forall S \in \mathcal{S}_v^u, uv \notin E, \tag{5.8}$$

$$x(v) \leq 1 \qquad \forall v \in V, \tag{5.9}$$

$$x(v) \geq 0 \qquad \forall v \in V, \tag{5.10}$$

$$x(v) \text{ } integer \qquad \forall v \in V \tag{5.11}$$

where Γ is the set of all terminal paths of G (a path between two terminals).

Inequalities (5.7) ensure that all terminal paths are disconnected. Inequalities (5.8) guarantee that the separator is connected.

Theorem 5.11 *Inequalities* (5.8) *can be separated in polynomial time.*

Proof. Consider a graph $G = (V \cup T, E)$, vector $x^* \in [0,1]^V$ and two non-adjacent vertices $u, v \in V$. Let $\alpha_v^u = x^*(u) + x^*(v) - 1$. Inequalities (5.8) associated with u and v can then be written as

$$x^*(S) \geq \alpha_v^u \quad \forall S \in \mathcal{S}_v^u.$$

Let $G_u^v = (V' \cup T', E')$ be the graph obtained from G as follows.

- For all $w \in V \setminus \{u, v\}$, add a vertex w' to V' of weight $\omega(a)$.
- For all $w \in T$, add a vertex w' to V' of weight $|V|$.
- For the two vertices u and v, add two terminals u' and v' to T'.
- For all edge $wz \in E$, add an edge $w'z'$ to E'.

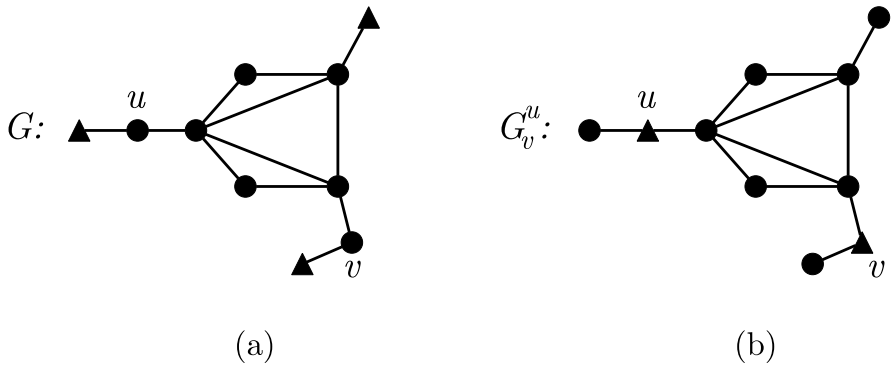


Figure 5.3: Graph transformation (triangles represent the terminals)

Figure 5.3 illustrates the above graph transformation. Thus, for u and v, separating inequalities (5.8) is equivalent to finding a minimum multi-terminal vertex separator in the graph G_u^v, where u and v are the terminals. Since G_u^v has only two terminals, solving the MTVS problem can be done in a polynomial time. By iterating over each pair of vertices u and v of V, we obtain a polynomial algorithm for separating inequalities (5.8). ■

5.2.2 Polyhedral analysis

Let $P(G,T)$ be the convex hull of all solutions of the above integer program, that is,

$$P(G,T) = conv\{x \in \{0,1\}^V |\ \ x \text{ satisfies } (5.7) - (5.8)\}.$$

In this subsection, we will discuss $P(G,T)$, we will give its dimension and describe necessary and sufficient conditions for inequalities (5.8) to be facet defining.

We first establish the dimension of $P(G,T)$.

Theorem 5.12 *$P(G,T)$ is full dimensional.*

Proof. We need to exhibit $n+1$ connected separators such that their incidence vectors are affinely independent. Let $S_0 = V$. From Hypothesis 5.5.2, S_0 is a connected separator of G. For each $v \in V$, let $\{W_1^v, \ldots, W_q^v\}$ be the set of all components of $G \setminus \{v\}$. From Hypothesis 5.5.1, each component contains at least one terminal. Moreover, from Hypotheses 5.5.3 and 5.5.4, $q \leq 2$, and at most one component has two terminals. If $q = 1$, then let $S_v = V \setminus \{v\}$. If $q = 2$, then we suppose that W_1^v contains exactly one terminal and let $S_v = V \setminus (\{v\} \cup W_1^v)$. Since no vertex of V is connected to two terminals, set S_v is a connected separator of G. The sets S_0 and S_v for all $v \in V$ constitute a set of $n+1$ connected separators of G. Moreover, their incidence vectors are clearly affinely independent. ■

Consider a graph $G = (V \cup T, E)$, two non-adjacent vertices $u, v \in V$ and a disconnecting subset S_v^u. Let $W_v \subset V$ be the component containing $v \in V$ in $G \setminus S_v^u$.

Theorem 5.13 *Inequality (5.8) associated with S_v^u defines a facet for $P(G,T)$ if and only if*

1- S_v^u does not contain a terminal path (internal vertices of the path between two terminals).

2- S_v^u is minimal, i.e., for all $w \in S_v^u$, $(S_v^u \setminus \{w\})$ is not a disconnecting subset of u and v.

3- No vertex of $V \setminus (S_v^u \cup \{u, v\})$ is connected to a terminal $t \in T$ and to two vertices of S_v^u, such that these two vertices are connected to the same terminal $t' \in T \setminus \{t\}$.

4- For every two connected components W_u and W_v of $G \setminus S_v^u$, $|T(V \setminus W_u)| \leq 1$ and $|T(V \setminus W_v)| \leq 1$. Which implies that $|T(W_u)| = 1$ and $|T(W_v)| = 1$.

Proof.
($\Rightarrow$)

1- If S_v^u contains a terminal path, then inequality (5.8) associated with S_v^u is dominated by the terminal path inequality (5.7) and hence cannot define a facet.

2- If S_v^u is not minimal, then there exists a vertex $w \in S_v^u$ such that $(S_v^u \setminus \{w\})$ is a disconnecting subset of u and v. Thus,

$$x(S_v^u \setminus \{w\}) \geq x(u) + x(v) - 1,$$

is valid for $P(G, T)$. Inequality (5.8), associated with S_v^u, can be obtained by summing the above inequality and $x(w) \geq 0$.

3- If there exists a vertex $z \in V \setminus (S_v^u \cup \{u, v\})$ that is adjacent to a terminal $t \in T$ and to two vertices $a, b \in S_v^u$, such that these two vertices are also adjacent to the same terminal $t' \in T \setminus \{t\}$, then the following inequality

$$x(S_v^u) \quad \geq \quad x(u) + x(v) - x(z) \tag{5.12}$$

is valid for $P(G, T)$. Indeed, if $x(z) = 1$, then the inequality is nothing but (5.8). If $x(z) = 0$, it follows that z does not belong to the connected separator. Since (t, z, a, t') and (t, z, b, t') are two terminal paths, it follows from the terminal path inequalities that $x(a) = x(b) = 1$. Thus, (5.12) is valid for $P(G, T)$. Inequality (5.8) associated with S_v^u can be obtained by summing (5.12) greater with $1 \geq x(z)$.

4- Suppose that $|T(W_v)| \geq 2$. Thus for all connected separator of G, at least one vertex of W_v must belong to it. Since S_v^u is a disconnected subset of W_u and W_v, it follows that if u belongs to the connected separator, then at least one vertex of S_v^u also belongs to the connected separator. Thus, the following inequality

$$x(S_v^u) \geq x(u)$$

is valid for $P(G, T)$. Inequality (5.8) associated with S_v^u can be obtained by summing the above inequality and $1 \geq x(v)$.

($\Leftarrow$)
Assume that all conditions of the theorem are satisfied. From Condition 4, graph G has only two terminals, one in W_u and the other in W_v.

Denote by $ax \geq \alpha$ inequality (5.8). Let $bx \geq \beta$ be an inequality that defines a facet of $P(G,T)$. Suppose that $\{x \in P(G,T) : ax = \alpha\} \subseteq \{x \in P(G,T) : bx = \beta\}$. We will show that there exists ρ such that $b = \rho a$.

For all $w \in S_v^u$, let $S_w = W_u \cup W_v \cup \{w\}$. From Condition 2, w is adjacent to a vertex of W_u and to a vertex of W_v. So, $G[S_w]$ is connected. From Condition 1, S_v^u does not contain any terminal path. Thus, S_w is a connected separator of G. Moreover, $ax^{S_w} = \alpha$. Hence, $bx^{S_w} = bx^{S_{w'}}$ for every $w, w' \in S_v^u$. Therefore,

$$b(w) = \rho \quad \text{for all } w \in S_v^u \text{ and some scalar } \rho \in \mathbb{R}.$$

From Condition 3, each vertex of $W_u \setminus \{u\}$ adjacent to a terminal of W_u, is adjacent to at most one vertex of S_v^u that is adjacent to a terminal of W_v. Thus, for all $z \in W_u \setminus \{u\}$, let $S_w^z = S_w \setminus \{z\}$, such that if z is connected to the terminal of W_u, w is the vertex of S_v^u that is adjacent to z and to the terminal of W_v, if it exists. Otherwise, w is any vertex from S_v^u. Since S_w is a connected separator, by Condition 3 it follows that S_w^z is also a connected separator of G. Moreover, $ax^{S_w^z} = \alpha$. Hence, $bx^{S_w} = bx^{S_w^z}$ for every $z \in W_u \setminus \{u\}$. Therefore,

$$b(z) = 0 \quad \text{for all } z \in W_u \setminus \{u\}.$$

Along the same line, we can also show that

$$b(z) = 0 \quad \text{for all } z \in W_v \setminus \{v\}.$$

Now consider a component W of $G \setminus S_v^u$, different from W_u and W_v. From Condition 4, W has no terminals. From Hypothesis 2.3, each vertex of W belongs to a terminal path. Thus, there must exist at least two vertices $w_1, w_2 \in S_v^u$ adjacent to two different vertices $z_1, z_2 \in W$. Assume that z_1 (z_2) is adjacent to w_1 (w_2). For $z \in W \setminus \{z_1\}$, let $S_1^z = (S_{w_1} \cup W)$ and $\overline{S}_1^z = (S_{w_1} \cup W) \setminus \{z\}$. Since S_{w_1} is a connected separator of G, it follows that S_1^z and $\overline{S}_1^z$ are also two connected separators of G. Moreover, $ax^{S_1^z} = \alpha$. Hence, $bx^{S_1^z} = bx^{\overline{S}_1^z}$ for every $z \in W \setminus \{z_1\}$. Therefore,

$$b(z) = 0 \quad \text{for all } z \in W \setminus \{z_1\}.$$

Along the same line we can also prove that

$$b(z_1) = 0.$$

Let $S'_u = W_u$. From Condition 3, it follows that S'_u is a connected separator of G. Moreover, $ax^{S'_u} = \alpha$. Hence $bx^{S'_u} = bx^{S_w}$, and consequently,

$$b(v) = -b(w) = -\rho.$$

By symmetry, it follows that

$$b(u) = -b(w) = -\rho,$$

This completes the proof. ■

5.3 The multi-terminal k-separator problem

In this section we discuss a variant of the MTVSP, called *the multi-terminal k-separator problem* (MTkSP). Given a graph $G = (V \cup T, E)$, an integer k and a weight function $\omega : V \to \mathbb{Z}$, the problem consists in finding a subset of vertices $S \subseteq V$ of minimum weight such that graph $G \setminus S$ has k disjoint subsets with

- at most one terminal, if $k > |T|$,
- exactly one terminal, if $k = |T|$,
- at least one terminal, if $k < |T|$.

Set S is called *k-separator*. We notice that if $k = |T|$, the multi-terminal k-separator problem is nothing but the multi-terminal vertex separator problem. In the two following subsections, we will discuss the two remaining cases of the multi-terminal k-separator problem, namely when $k > |T|$ and $k < |T|$.

5.3.1 The multi-terminal k-separator problem, when $k > |T|$

In this section we discuss the multi-terminal k-separator problem when $k > |T|$. We first define some notation used in the rest of the subsection. A vertex $v \in V$ is called a *possible terminal* if there exists an independent subset $T' \subset V \cup T$ of size k such that $T \subseteq T'$ and $v \in T'$. Clearly, each vertex of $N(T)$ cannot be a possible terminal. The set of internal vertices of a path between a terminal of T and a possible terminal $a \in V$ is called a *one-terminal path* and denoted by P_a. The set of internal vertices of a path

between two possible terminals $a, b \in V$ is called a *possible-terminal path* and denoted by P_{ab}.

Consider the following Hypotheses

5.1.1- For all $v \in V$, at least one of the following items is satisfied

- a- v is a possible terminal.
- b- There exists a minimal terminal path containing v.
- c- There exists a minimal one-terminal path, containing v.
- d- If $k - |T| \geq 2$, there is a minimal possible-terminal path containing v.

Otherwise, v cannot belong to a minimal k-separator. So we can delete it.

5.2.2- There exists an independent set of size k containing T, otherwise there is no solution for the problem.

Theorem 5.14 *The multi-terminal k-separator problem, when $k > |T|$, is NP-hard.*

Proof. We will show that the multi-terminal k-separator problem, when $k = |T| + 1$ is NP-hard using a polynomial reduction from the multi-terminal vertex separator problem. Given a graph $G = (V \cup T, E)$, let G' be a graph obtained from G by adding one vertex w of degree 0, having a weight $|V| \max_{v \in V}\{\omega(v)\}$. Then we have the following claim.

Claim 1. A multi-terminal vertex separator in G is minimum if and only if it is a k-separator of minimum weight in G'.

Proof. ($\Rightarrow$)
Given a MTV separator S in G, let $\mathcal{W} = \{W_1, \ldots, W_k\}$ be the set of components of $G \setminus S$. Let $\mathcal{W}' = \mathcal{W} \cup \{w\}$. It can be easily seen that S is a k-separator of G' and it is optimal.
($\Leftarrow$)
Since the weight of w is $|V| \max_{v \in V}\{\omega(v)\}$, w cannot belong to the k-separator of minimum weight in G'. Let S be the k-separator of minimum weight in G'. It can easily seen that it is a multi-terminal vertex separator in G. □

Since the multi-terminal vertex separator problem is NP-hard, it follows, by Claim 1, that the multi-terminal k-separator problem so is. ■

5.3.1.1 Formulation

We will introduce here a formulation for the MTkS problem when $k > |T|$. The following ILP formulation permits to select an independent subset $T' \subseteq V \cup T$ of size k containing T, such that the minimum vertex separator S intersecting all paths between each pair of vertices in T' has a minimum weight. The vertices of $T' \setminus T$ are called *artificial terminals*. We need two vectors of variables $x \in \{0,1\}^V$ and $y \in \{0,1\}^V$ such that for all $v \in V$, $x(v) = 1$ if vertex v belongs to the k-separator, and 0 otherwise and $y(v) = 1$ if vertex v is an artificial terminal, and 0 otherwise.

The multi-terminal k-separator is equivalent to the following integer linear program

$$\min \sum_{v \in V} w(v)x(v)$$

$$y(V) = k - |T| \tag{5.13}$$

$$y(V) = 0 \qquad \forall v \in N(T), \tag{5.14}$$

$$y(a) + y(b) \leq 1 \qquad \forall \{a,b\} \subseteq V \setminus N(T), \forall ab \in E, \tag{5.15}$$

$$y(a) + x(a) \leq 1 \qquad \forall a \in V \setminus N(T), \tag{5.16}$$

$$x(P_{ab}) \geq y(a) + y(b) - 1 \qquad \forall \textit{ possible-terminal path } P_{ab}, \tag{5.17}$$

$$x(P_a) \geq y(a) \qquad \forall \textit{ one-terminal path } P_a, \tag{5.18}$$

$$x(P) \geq 1 \qquad \forall \text{ terminal path } P, \tag{5.19}$$

$$x(v) \in \{0,1\} \qquad \forall v \in V, \tag{5.20}$$

$$y(v) \in \{0,1\} \qquad \forall v \in V \setminus N(T). \tag{5.21}$$

Inequalities (5.13) and (5.14) ensure that exactly $k - |T|$ possible vertices are selected as artificial terminals. Inequalities (5.15)-(5.16) guarantee that two artificial terminals are not adjacent and no artificial terminal belongs to the k-separator. Inequalities (5.17)-(5.19) ensure that all paths between each pair of vertices in T' intersect the multi-terminal k-separator.

Remark 5.15 *When $k = |T|$, inequalities (5.17) and (5.18) are redundant.*

Remark 5.16 *When $|T| = 0$, inequalities (5.18) and (5.19) do not exist.*

Theorem 5.17 *Inequalities* (5.17) *can be separated in polynomial time.*

Proof. Consider a solution $(\overline{x}, \overline{y})$ such that $\overline{x} \in [0,1]^V$ and $\overline{y} \in [0,1]^V$. The separation problem consists in finding a possible-terminal path P_{ab} such that $\overline{x}(P_{ab}) < \overline{y}(a) + \overline{y}(b) - 1$. This is equivalent to find, for every non-adjacent vertices $a, b \in V \setminus N(T)$, a shortest path between a and b in G w.r.t $\overline{x}$. ■

Theorem 5.18 *Inequalities* (5.18) *and* (5.19) *can be separated in polynomial time.*

Proof. The separation problem is reduced to a shortest path problem. ■

5.3.1.2 Valid inequalities

Consider a graph $G = (V \cup T, E)$. Let $P(G,T)$ be the convex hull of all solutions of the integer linear system (5.13) – (5.21). In what follows, we present some valid inequalities for $P(G,T)$.

5.3.1.2.1 Path inequalities

Theorem 5.19 *Given a terminal path $P_{tt'}$ of G between two terminals $t, t' \in T$, the following inequality*

$$x(P_{tt'}) \geq 1 + y(P_{tt'}) \tag{5.22}$$

is valid for $P(G,T)$.

Proof. Consider a solution (x, y). We distinguish the following cases.

- If $y(P_{tt'}) = 0$, then inequality (5.22) is valid by (5.19).
- If $y(P_{tt'}) = 1$, then let $u \in P_{tt'}$ such that $y(u) = 1$. Thus, u is an artificial terminal. Let P_t (resp. $P_{t'}$) be the path in $P_{tt'}$ between u and t (resp. t'). Hence at least one vertex of P_t (resp. $P_{t'}$) belongs to the k-separator. It follows that inequality (5.22) is valid.
- If $y(P_{tt'}) \geq 2$, then there exist $y(P_{tt'}) - 1$ internal vertices of $P_{tt'}$ that are artificial terminals. It follows that $P_{tt'}$ contains exactly two one-terminal paths and $y(P_{tt'}) - 1$ possible-terminal paths, with no internal vertex in common. Each of these paths must intersect the k-separator. Thus, one should have $y(P_{tt'}) + 1$ vertices from $P_{tt'}$ in the k-separator. ■

Theorem 5.20 *Inequalities* (5.22) *can be separated in polynomial time.*

Proof. Consider a solution $(\overline{x}, \overline{y})$ such that $\overline{x} \in [0,1]^V$ and $\overline{y} \in [0,1]^V$. Let $z \in [0,1]^V$ be a vector such that for all $v \in V \setminus T$, $z(v) = \overline{x}(v) - \overline{y}(v)$. For all $t, t' \in T$, consider the shortest path between t and t' in G w.r.t z. And consider the path having the minimum length among these paths. If these length is less than 1, then the corresponding path induces an inequality of type (5.22) violated by $(\overline{x}, \overline{y})$. As the shortest path problem is polynomial, the separation of (5.22) is polynomial. ■

5.3.1.2.2 Star tree inequalities

Let $H = (V(H) \cup T(H), E(H))$ be a star tree defined in Chapter 2.

Theorem 5.21 *If H is a subgraph of G, the following inequality*

$$x(V(H) \setminus \{v_r\}) + (q-1)x(v_r) \geq q - 1 + y(V(H)) \tag{5.23}$$

is valid for $P(G,T)$.

Proof. Given a solution (x, y). If $y(V(H)) = 0$, then (x, y) satisfies inequality (5.23). Indeed, x satisfies the star tree inequality (2.10).

Let us assume that $y(V(H)) \geq 1$ and let $G' = (V' \cup T', E')$ be the graph obtained from G, by replacing each vertex $v \in V$ with $y(v) = 1$ by a terminal. Clearly, if x satisfy the following inequality

$$x(V'(H) \setminus \{v_r\}) + (q'-1)x(v_r) \geq q' - 1 \tag{5.24}$$

then (x, y) satisfies inequality (5.23), where $q' = q + y(V(H))$.

- If $y(v_r) = 0$, then G' has one star tree of q terminals, and $y(V(H))$ terminal paths with no internal vertex in common. Thus from the star tree inequalities (2.10) and the terminal path inequalities (2.6), x satisfies inequality (5.24).
- If $y(v_r) = 1$, it follows that G' has q' terminal paths, with no internal vertex in common. Consequently, from the terminal path (2.6), x satisfies inequality (5.24).

It follows that (x, y) satisfies inequality (5.23). Since (x, y) is arbitrarily chosen, it follows that inequality (5.23) is valid for $P(G, T)$. ■

Theorem 5.22 *Inequalities* (5.23) *can be separated in polynomial time.*

Proof. Consider a solution $(\overline{x}, \overline{y})$ such that $\overline{x} \in [0, 1]^V$ and $\overline{y} \in [0, 1]^V$. Let $z \in [0, 1]^V$ be a vector such that for all $v \in V \cup \{v_r\}$, $z(v) = \overline{x}(v) - \overline{y}(v)$ and $z(v_r) = \overline{x}(v_r) - \frac{1}{q-1}\overline{y}(v_r)$. Thus, separating inequalities (5.23) is equivalent to separating the following inequalities

$$z(V(H) \setminus \{v_r\}) + (q-1)z(v_r) \geq q - 1.$$

The algorithm to separate the above inequalities is exactly the one of separating the star tree inequality (2.10), that can be done in a polynomial time (Theorem 2.27 in Chapter 2). ■

5.3.1.2.3 Extended terminal cycle inequalities

Let $J = (V(J) \cup T(J), E(J))$ be an extended terminal cycle defined in Chapter 2, such that each vertex of the cycle is of degree 3.

Theorem 5.23 *If J is a subgraph of G, the following inequality*

$$x(V(J)) \geq \lceil \frac{q}{2} \rceil + y(V(J)) \tag{5.25}$$

is valid for $P(G, T)$.

Proof. The proof is by induction on the value of $y(V(J))$. It is clear that if $y(V(J)) = 0$, by Theorem 2.22, inequality (5.25) is valid for $P(G, T)$. We assume that inequality (5.25) is valid for any value of $y(V(J)) < s$, and we prove that the inequality is also valid for $y(V(J)) = s$. Consider a feasible solution (x^*, y^*) such that $y^*(V(J)) = s$. Let $(\overline{x}, \overline{y})$ be a solution obtained from (x^*, y^*) by setting $\overline{y}(v) = 0$ for a vertex $v \in V(J)$ such that $y^*(v) = 1$. Solution $(\overline{x}, \overline{y})$ remains feasible. Since J is a terminal cycle, there must exist a path P_v between v and a terminal or an artificial terminal u, such that all the internal vertices are of degree 2 and $y^*(P_v \setminus \{u, v\}) = 0$. Clearly, $x^*(P_v) \geq 1$. Let (x', y') be a new vector obtained from $(\overline{x}, \overline{y})$ by setting $x'(P_v) = 0$. Solution (x', y') is also feasible. Thus by the induction hypothesis,

$$x'(V(J)) \geq \lceil \frac{q}{2} \rceil + y'(V(J)).$$

Since $\overline{x}(V(J)) \geq x'(V(J)) + 1$, it follows that

$$\overline{x}(V(J)) \geq x'(V(J)) + 1 \geq \lceil \frac{q}{2} \rceil + y'(V(J)) + 1.$$

As $\overline{y}(V(J)) = y'(V(J) \setminus N(T(J))) = y^*(V(J)) - 1$ and $\overline{x} = x^*$, then

$$x^*(V(J)) \geq \lceil \frac{q}{2} \rceil + y^*(V(J)).$$

Which ends the proof of the theorem. ■

5.3.1.2.4 Clique inequalities

Theorem 5.24 *For all clique K subgraph of G, the following inequality*

$$y(K) \leq 1 \tag{5.26}$$

is valid for $P(G,T)$.

Proof. Since the artificial terminals cannot be adjacent, it follows that in a clique only one vertex can be an artificial terminal. ■

5.3.1.2.5 Clique star inequalities

Let $Q = (V(Q) \cup T(Q), E(Q))$ be a clique star defined in Chapter 2.

Theorem 5.25 *If Q is subgraph of G, the following inequality*

$$x(V(Q)) \geq q - 1 + y(V(Q)) \tag{5.27}$$

is valid for $P(G,T)$.

Proof. Similar to the proof of Theorem 5.21. ■

5.3.1.2.6 Terminal tree inequalities

Let $R = (V(R) \cup T(R), E(R))$ be a terminal tree defined in Chapter 2.

Theorem 5.26 *If R is a subgraph of G, the following inequality*

$$\sum_{v \in V(R)} (d_R(v) - 1)x(v) \geq q - 1 + y(V(R)) \tag{5.28}$$

is valid for $P(G, T)$.

Proof. The proof is by induction on q. If $q \leq 3$, inequality (5.28) is nothing but a star tree inequality (5.23) associated with R, and then it is valid. Let us suppose that $q \geq 3$, and that for each terminal tree with less than $q-1$ terminals, the associated inequality (5.28) is valid for $P(G, T)$. Since $q \geq 3$, there must exist a vertex u belonging to two leaf branches P_t and $P_{t'}$. Let us consider the terminal tree R^1 (resp. R^2) obtained from R by removing the vertices of $P_t \setminus \{u\}$ (resp. $P_{t'} \setminus \{u\}$). Thus, R^1 and R^2 have each one $q-1$ leaves and hence $q-1$ terminals. By the induction Hypothesis, the terminal tree inequalities (5.28) associated with R^1 and R^2

$$\sum_{v \in V(R) \setminus (P_t \cup P_{t'})} (d_R(v) - 1)x(v) + \sum_{v \in P_{t'} \setminus \{u\}} (d_R(v) - 1)x(v) + (d_R(u) - 2)x(u) \geq q - 2 + y(V(R) \setminus (P_t \setminus \{u\})),$$

$$\sum_{v \in V(R) \setminus (P_t \cup P_{t'})} (d_R(v) - 1)x(v) + \sum_{v \in P_t \setminus \{u\}} (d_R(v) - 1)x(v) + (d_R(u) - 2)x(u) \geq q - 2 + y(V(R) \setminus (P_{t'} \setminus \{u\})),$$

are valid for $P(G, T)$. By summing these inequalities together with the inequality $x(P_t \cup P_{t'}) \geq 1 + y(P_t \cup P_{t'})$ induced by the terminal path $P_t \cup P_{t'}$, inequality $x(u) \geq 0$ and inequality $y(u) \geq 0$, we obtain the inequality

$$\sum_{v \in V(R) \setminus (P_t \cup P_{t'})} 2(d_R(v) - 1)x(v) + \sum_{v \in P_t \cup P_{t'}} (2(d_R(v) - 1))x(v) \geq 2q - 3 + 2y(V(R)).$$

By dividing by 2 and rounding up the right hand side, we obtain (5.28). ■

5.3.1.2.7 Odd cycle inequalities

Theorem 5.27 *For an odd cycle $C \subseteq V \cup T$, the following inequality*

$$y(C) \leq \lfloor \frac{|C|}{2} \rfloor - |C \cap T| \tag{5.29}$$

is valid for $P(G,T)$.

Proof. Clearly, the sum of the number of artificial terminals and the number of terminals in C, should be less than or equal to $\lfloor \frac{|C|}{2} \rfloor$. ■

5.3.2 The multi-terminal k-separator problem, when $k < |T|$

In this section we discuss the multi-terminal k-separator problem when $k < |T|$.

Theorem 5.28 *The multi-terminal k-separator problem, when $k < |T|$, is NP-hard.*

Proof. Using a polynomial reduction from the MTVSP, the proof is similar to the one of Theorem 5.14. ■

5.3.2.1 Formulation

The following ILP formulation permits to identify k subsets of terminals and then to find the k-separator intersecting all paths between these subsets. For this formulation we need two vectors of variables $x \in \{0,1\}^V$ and $y \in \{0,1\}^{T \times K}$ such that for all $v \in V$, $x(v)$ is equal to 1 if vertex v belongs to the k-separator, and 0 otherwise, and for all $t \in T$, y_{ti} is equal to 1 if terminal t belongs to subset V_i, and 0 otherwise.

The multi-terminal k-separator problem, when $k < |T|$ is equivalent to the following integer linear program

$$\min \quad \sum_{v \in V} w(v)x(v)$$

$$\sum_{t \in T} y_{ti} \geq 1, \qquad \forall i \in \{1, \ldots, k\}, \tag{5.30}$$

$$\sum_{i \in \{1,\ldots,k\}} y_{ti} = 1, \qquad \forall t \in T, \tag{5.31}$$

$$x(P_{tt'}) \geq y_{ti} + \sum_{j \in \{1,\ldots,k\} \setminus \{i\}} y_{t'j} - 1, \qquad \forall P_{tt'},\ \forall i \in \{1, \ldots, k\}, \tag{5.32}$$

$$x(v) \in \{0, 1\}, \qquad \forall v \in V, \tag{5.33}$$

$$y_{ti} \in \{0, 1\} \qquad \forall t \in T, \quad i \in \{1, \ldots, k\}. \tag{5.34}$$

Inequalities (5.30) ensure that each subset contains at least one terminal. Inequalities (5.31) guarantee that each terminal belongs to exactly one subset. Inequalities (5.32) disconnect each pair of terminals that belong to different subsets.

5.3.2.2 Valid inequalities

Let $P(G, T)$ be the convex hull of all solutions of (5.30)-(5.34). In what follows, we present some valid inequalities for $P(G, T)$.

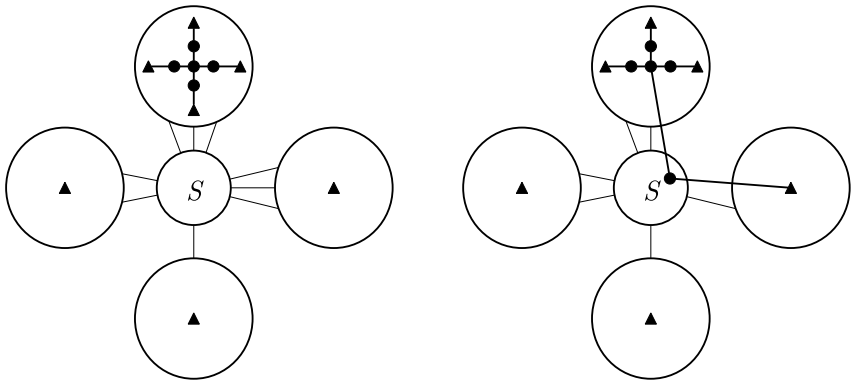
5.3.2.2.1 Star tree inequalities

Theorem 5.29 *For a star tree H subgraph of $G = (V \cup T, E)$, with a root vertex $v_r \in V$ and q terminals, the following inequality*

$$x(V(H) \setminus \{v_r\}) + (k - 1 - (|T| - q))x(v_r) \geq k - 1 - (|T| - q) \tag{5.35}$$

is valid for $P(G, T)$

Proof.



(a) (b)

Figure 5.4: Two examples of solutions, when $|T| = 7$, $|T| = 6$ and $k = 4$

It is clear that each k-separator of G containing v_r satisfies inequality (5.35). In what follows we show that all k-separators of G not containing v_r satisfy also inequality (5.35). Assume that inequality 5.35 is not valid for $P(G, T)$. It follows that there exists a k-separator $S \subset V \setminus \{v_r\}$ of G such that

$$x^S(V(H) \setminus \{v_r\}) + (k - 1 - (|T| - q))x^S(v_r) < k - 1 - (|T| - q),$$

Since $x^S(v_r) = 0$, it follows that

$$x^S(V(H) \setminus \{v_r\}) \leq k - 2 - (|T| - q).$$

Hence $|S| \leq k - 2 - (|T| - q)$. It is easy to see that deleting $p \in \mathbb{N}$ of $V(H) \setminus \{v_r\}$ from H, induces a graph with at most $p + 1$ components with at least one terminal. Thus $H \setminus S$ has at most $(k - 2 - (|T| - q)) + 1$ components with at least one terminal. Since the number of terminals in $T \setminus T(H)$ is $|T| - q$, it follows that $G \setminus S$ has at most $(k - 1 - (|T| - q)) + |T| - q = k - 1$ components with at least one terminal, which contradicts the fact that S is a k-separator. ■

5.3.2.2.2 Clique star inequalities

Theorem 5.30 *For a clique star Q subgraph of G, the following inequality*

$$x(V(Q)) \geq k - 1 - (|T| - q) \tag{5.36}$$

is valid for $P(G, T)$.

Proof. Similar to the proof of Theorem 5.29.

5.3.2.2.3 Terminal cycle inequalities

Theorem 5.31 *For a terminal cycle J subgraph of G, the following inequality*

$$x(J) \geq \lceil \frac{k+q-|T|-1}{2} \rceil \tag{5.37}$$

is valid for $P(G,T)$.

Proof. Similar to the proof of Theorem 5.29.

5.4 Conclusion

In this chapter we have considered four variants of the multi-terminal vertex separator problem. We have shown that each problem is NP-hard, and for each one, we have proposed an integer programming formulation. For the connected components separator and the multi-terminal connected separator problems, we have discussed some facet defining inequalities and for the multi-terminal vertex k-separator problems, we have adapted several inequalities introduced in Chapter 2 and Section 2.3 and proved their validity for the associated polytopes.

Conclusion

In this dissertation, we have studied the multi-terminal vertex separator problem within a polyhedral context. In the first part of the thesis we have proposed two integer programming formulations for the problem, investigated the basic properties of the associated polytope, and derived new classes of valid inequalities. Moreover, we have described necessary and sufficient conditions for these inequalities to define facets. The have thereafter been used to devise a Branch-and-Cut algorithm for the MTVS problem. This has been implemented to solve DIMACS and random instances. The experiment results show in particular the efficiency of the valid inequalities and the separation procedure used in the algorithm.

Afterwards, we have studied the structure of the multi-terminal vertex separator polytope. We have given a complete description of the MTVS polytope for terminal paths, star trees and clique stars. Moreover, we have shown that the two linear systems given by the star tree and clique star inequalities together with the trivial inequalities are totally dual integral for the star trees and the clique stars, respectively. Then, we have examined the polytope in the graphs decomposable by one node cutsets. We have shown that, if G is a graph that decomposes into G_1 and G_2, the multi-terminal vertex separator polytope in G can be obtained from two linear systems related to G_1 and G_2.

Moreover, we have proposed three extended formulations for the MTVSP. We have developed Branch-and-Price algorithms for all the formulations and Branch-and-Cut-and-Price algorithms for two of them. For each formulation we presented a column generation scheme to solve the linear relaxation, the way to compute the dual bound and the branching scheme. We have also presented extensive computational results, and discussed the performance of each algorithm.

In the last part of the dissertation, we have considered four variants of the multi-terminal vertex separator problem. We have shown that each variant is NP-hard, and for each variant we have proposed an integer programming formulation along with some

valid inequalities.

As perspectives, there are different directions in which our future research related to the MTVSP can be conducted.

First, more efficient separation heuristics and more sophisticated preprocessing methods can be developed in order to improve the resolution of the problem. Moreover, it would be interesting to characterize the graphs for which the terminal tree, clique star and terminal cycle inequalities suffice to describe the multi-terminal vertex separator polytope. Indeed, as it appears from the computational results in Chapter 2, using these inequalities, many instances have been solved in the root of the Branch-and-Cut tree. In [95], the authors characterize the class of graph for which the system given by the terminal path inequalities and the trivial inequalities is TDI. Since the terminal tree, clique star and terminal cycle inequalities generalize the terminal path inequalities, it would be also interesting to describe graphs for which the system, given by these classes of inequalities, is TDI.

Finally, it would be interesting to investigate some stabilization methods in column generation in order to improve the convergence of our Branch-and-Price and Branch-and-Cut-and-Price algorithms. Also comparing different column generation strategies and heuristics to solve the pricing problems would be of interest.

Bibliography

[1] http://www.info.univ-angers.fr/pub/porumbel/graphs/.

[2] https://lemon.cs.elte.hu/trac/lemon.

[3] http://www.boost.org/doc/libs/1_60_0/libs/graph/.

[4] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[5] https://gephi.org/.

[6] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

[7] https://www.yworks.com/products/yed.

[8] C. Alves and J. V. de Carvalho. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, 35(4):1315 – 1328, 2008.

[9] K. Andreev and H. Räcke. Balanced graph partitioning. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 120–124. ACM, 2004.

[10] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems, 2003.

[11] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 284–293. ACM, 1995.

[12] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

[13] M. Baïou, F. Barahona, and A. R. Mahjoub. Separation of partition inequalities. *Mathematics of Operations Research*, 25(2):243–254, 2000.

[14] E. Balas and C. C. de Souza. The vertex separator problem: a polyhedral investigation. *Mathematical Programming*, 103(3):583–608, 2005.

[15] F. Barahona, J. Fonlupt, and A. R. Mahjoub. Compositions of graphs and polyhedra iv: Acyclic spanning subgraphs. *SIAM Journal on Discrete Mathematics*, 7(3):390–402, 1994.

[16] F. Barahona and A. R. Mahjoub. Facets of the balanced (acyclic) induced subgraph polytope. *Mathematical Programming*, 45(1-3):21–33, 1989.

[17] F. Barahona and A. R. Mahjoub. Compositions of graphs and polyhedra i: Balanced induced subgraphs and acyclic subgraphs. *SIAM Journal on Discrete Mathematics*, 7(3):344–358, 1994.

[18] F. Barahona and A. R. Mahjoub. Compositions of graphs and polyhedra ii:. *SIAM Journal on Discrete Mathematics*, 7(3):359–371, 1994.

[19] F. Barahona and A. R. Mahjoub. Compositions of graphs and polyhedra iii: Graphs with no w_4 minor. *SIAM Journal on Discrete Mathematics*, 7(3):372–389, 1994.

[20] F. Barahona and A. R. Mahjoub. On two-connected subgraph polytopes. *Discrete Mathematics*, 147(1):19–34, 1995.

[21] C. Barnhart, A. M. Cohn, E. L. Johnson, D. Klabjan, G. L. Nemhauser, and P. H. Vance. Airline crew scheduling. In *Handbook of Transportation Science*, volume 56 of *International Series in Operations Research & Management Science*, pages 517–560. 2003.

[22] C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.*, 48(2):318–326, 2000.

[23] M. Bateni, M. Hajiaghayi, P. N. Klein, and C. Mathieu. A polynomial-time approximation scheme for planar multiway cut. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 639–655. Society for Industrial and Applied Mathematics, 2012.

[24] W. Ben-Ameur and M. D. Biha. Algorithms and formulations for the minimum cut separator problem. *Electronic Notes in Discrete Mathematics*, 36:977–983, 2010.

[25] W. Ben-Ameur and M. Didi Biha. On the minimum cut separator problem. *Networks*, 59(1):30–36, 2012.

[26] W. Ben-Ameur, M.-A. Mohamed-Sidi, and J. Neto. The k-separator problem. In *Computing and Combinatorics*, pages 337–348. Springer, 2013.

[27] W. Ben-Ameur, M.-A. Mohamed-Sidi, and J. Neto. The k-separator problem: polyhedra, complexity and approximation results. *Journal of Combinatorial Optimization*, 29(1):276–307, 2015.

[28] A. Berger, A. Grigoriev, and R. Zwaan. Complexity and approximability of the k-way vertex cut. *Networks*, 63(2):170–178, 2014.

[29] M. D. Biha. Personal communication. page 2014.

[30] M. D. Biha. On the 3-terminal cut polyhedron. *SIAM Journal on Discrete Mathematics*, 19(3):575–587, 2005.

[31] M. D. Biha and A. R. Mahjoub. k-edge connected polyhedra on series-parallel graphs. *Operations research letters*, 19(2):71–78, 1996.

[32] M. D. Biha and M.-J. Meurs. An exact algorithm for solving the vertex separator problem. *Journal of Global Optimization*, 49(3):425–434, 2011.

[33] V. Boginski and C. W. Commander. Identifying critical nodes in protein-protein interaction networks. *Clustering challenges in biological networks*, pages 153–167, 2009.

[34] M. Bouchakour. *Composition de graphes et le polytope des absorbants. Un algorithme de coupes pour le problème du flot à coûts fixes.* PhD thesis, Université de Rennes 1, 1996.

[35] T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.

[36] G. Călinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 48–52. ACM, 1998.

[37] J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.

[38] R. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk. Designing fpt algorithms for cut problems using randomized contractions. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 460–469. IEEE, 2012.

[39] S. Chopra and J. H. Owen. Extended formulations for the a-cut problem. *Mathematical programming*, 73(1):7–30, 1996.

[40] S. Chopra and M. R. Rao. On the multiway cut polyhedron. *Networks*, 21(1):51–89, 1991.

[41] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[42] D. Cornaz, F. Furini, M. Lacroix, E. Malaguti, A. R. Mahjoub, and S. Martin. Mathematical formulations for the balanced vertex k-separator problem. In *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on*, pages 176–181. IEEE, 2014.

[43] D. Cornaz, Y. Magnouche, A. R. Mahjoub, and S. Martin. The multi-terminal vertex separator problem: Polyhedral analysis and branch-and-cut. In *International Conference on Computers & Industrial Engineering (CIE45)*, 2015.

[44] W. H. Cunningham and L. Tang. Optimal 3-terminal cuts and linear programming. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 114–125. Springer, 1999.

[45] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On multiway cut parameterized above lower bounds. In *Parameterized and Exact Computation*, pages 1–12. Springer, 2011.

[46] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[47] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):pp. 101–111, 1960.

[48] C. de Souza and E. Balas. The vertex separator problem: algorithms and computations. *Mathematical Programming*, 103(3):609–631, 2005.

[49] E. D. Demaine, M. Hajiaghayi, and P. N. Klein. Node-weighted steiner tree and group steiner tree in planar graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 328–340. Springer, 2009.

[50] G. Desaulniers, J. Desrosiers, and M. M. Solomon. Column generation. volume 5. Springer-Verlag New York Incorporated, 2005.

[51] M. Di Summa, A. Grosso, and M. Locatelli. Branch and cut algorithms for detecting critical nodes in undirected graphs. *Computational Optimization and Applications*, 53(3):649–680, 2012.

[52] T. N. Dinh, Y. Xuan, M. T. Thai, E. Park, and T. Znati. On approximation of new optimization methods for assessing network vulnerability. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[53] R. G. Downey, V. Estivill-Castro, M. R. Fellows, E. Prieto, F. A. Rosamond, et al. Cutting up is hard to do: the parameterized complexity of k-cut and related probelms| nova. the university of newcastle's digital repository. 2003.

[54] J. Edmonds. Covers and packings in a family of sets. *Bulletin of the American Mathematical Society*, 68(5):494–499, 1962.

[55] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards (B) 69*, 69:9–14, 1965.

[56] N. Fan and P. M. Pardalos. Robust optimization of graph partitioning and critical node detection in analyzing networks. In *International Conference on Combinatorial Optimization and Applications*, pages 170–183. Springer, 2010.

[57] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*, pages 241–247. ACM, 1988.

[58] J. Fonlupt and A. R. Mahjoub. Critical extreme points of the 2-edge connected spannning subgraph polytope. In *Integer Programming and Combinatorial Optimization*, pages 166–182. Springer, 1999.

[59] J. Fukuyama. Np-completeness of the planar separator problems. *J. Graph Algorithms Appl.*, 10(2):317–328, 2006.

[60] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, 1979.

[61] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976.

[62] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in directed and node weighted graphs. In *Automata, Languages and Programming*, pages 487–498. Springer, 1994.

[63] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50(1):49–61, 2004.

[64] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[65] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem-Part II. *Operations Research*, 11(6):863–888, 1963.

[66] O. Goldschmidt and D. S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of operations research*, 19(1):24–37, 1994.

[67] M. Grötschel and C. L. Monma. Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM Journal on Discrete Mathematics*, 3(4):502–523, 1990.

[68] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[69] S. Guillemot. Fpt algorithms for path-transversals and cycle-transversals problems in graphs. In *International Workshop on Parameterized and Exact Computation*, pages 129–140. Springer, 2008.

[70] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier, and J. Uhlmann. Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs. *European Journal of Operational Research*, 186(2):542–553, 2008.

[71] A. Hadjar. *Composition de polyèdres associés aux problèmes d'optimisation combinatoire*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 1996.

[72] X. He. An improved algorithm for the planar 3-cut problem. *Journal of Algorithms*, 12(1):23–37, 1991.

[73] D. S. Hochbaum and D. B. Shmoys. An o(|v|ˆ2) algorithm for the planar 3-cut problem. *SIAM Journal on Algebraic Discrete Methods*, 6(4):707–712, 1985.

[74] T. C. Hu. *Integer programming and network flows*. Addison-Wesley Publ., 1970.

[75] X. Ji and J. E. Mitchell. Branch-and-price-and-cut on the clique partitioning problem with minimum clique size requirement. *Discrete Optimization*, 4(1):87–102, 2007.

[76] Y. Kamidoi, N. Yoshida, and H. Nagamochi. A deterministic algorithm for finding all minimum k-way cuts. *SIAM Journal on Computing*, 36(5):1329–1341, 2006.

[77] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.

[78] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. 1972.

[79] G. Karypis and V. Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

[80] K.-i. Kawarabayashi and M. Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 160–169. IEEE, 2011.

[81] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[82] D. König. Graphok és alkalmazásuk a determinánsok és a halmazok elméletére. *Mathematikai és Természettudományi Ertesito*, 34:104–119, 1916.

[83] R. Krauthgamer and U. Feige. A polylogarithmic approximation of the minimum bisection. *SIAM review*, 48(1):99–130, 2006.

[84] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[85] M. E. Lübbecke and J. Desrosiers. Selected Topics in Column Generation. *Operations Research*, 53(6):1007–1023, 2005.

[86] A. R. Mahjoub. Two-edge connected spanning subgraphs and polyhedra. *Mathematical Programming*, 64(1-3):199–208, 1994.

[87] A. R. Mahjoub. On perfectly two-edge connected graphs. *Discrete Mathematics*, 170(1):153–172, 1997.

[88] A. R. Mahjoub. *Polyhedral Approaches*, pages 261–324. Wiley Online Library, 2013.

[89] F. Margot. *Composition de polytopes combinatoires: une approche par projection*, volume 4. PPUR presses polytechniques, 1995.

[90] S. Martin. *Analyse structurelle des systèmes algébro-différentiels conditionnels : compléxité, modèles et polyèdres*. PhD thesis, Université Paris-Dauphine, 2011.

[91] D. Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

[92] T. C. Matisziw and A. T. Murray. Modeling s–t path availability to support disaster vulnerability assessment of network infrastructure. *Computers & Operations Research*, 36(1):16–26, 2009.

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

[94] Y.-S. Myung and H.-j. Kim. A cutting plane algorithm for computing k-edge survivability of a network. *European Journal of Operational Research*, 156(3):579–589, 2004.

[95] G. Naves and V. Jost. The graphs with the max-mader-flow-min-multiway-cut property. *arXiv preprint arXiv:1101.2061*, 2011.

[96] G. L. Nemhauser and G. Sigismondi. A Strong Cutting Plane/Branch-and-Bound Algorithm for Node Packing. *The Journal of the Operational Research Society*, 43(5):443–457, 1992.

[97] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, 1988.

[98] M. Oosten, J. H. Rutten, and F. C. Spieksma. Disconnecting graphs by removing vertices: a polyhedral approach. *Statistica Neerlandica*, 61(1):35–60, 2007.

[99] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5(1):199–215, 1973.

[100] M. W. Padberg. A note on 0-1 programming. *Operations Research*, 23:833–837, 1979.

[101] C. Papadopoulos. Restricted vertex multicut on permutation graphs. *Discrete Applied Mathematics*, 160(12):1791–1797, 2012.

[102] S. Ropke and J. F. Cordeau. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 43(3):267–286, 2009.

[103] A. L. Rosenberg and L. S. Heath. *Graph separators, with applications*. Springer Science & Business Media, 2001.

[104] H. Saran and V. V. Vazirani. Finding k-cuts within twice the optimal. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 743–751. IEEE, 1991.

[105] H. Saran and V. V. Vazirani. Finding k cuts within twice the optimal. *SIAM Journal on Computing*, 24(1):101–108, 1995.

[106] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45(6):pp. 831–841, 1997.

[107] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1998.

[108] A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency. Algorithms and Combinatorics*, volume 24. Springer, 2003.

[109] M. Thorup. Minimum k-way cuts via deterministic greedy tree packing. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 159–166. ACM, 2008.

[110] F. Vanderbeck. *Decomposition and column generation for integer programming.* PhD thesis, Université Catholique de Louvain, Belgium, 1994.

[111] M. Xiao. Simple and improved parameterized algorithms for multiterminal cuts. *Theory of Computing Systems*, 46(4):723–736, 2010.

[112] M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM Journal on Computing*, 10(2):310–327, 1981.

Résumé

Étant donné un graphe G = (V U T, E), tel que V U T représente l'ensemble des sommets où T est un ensemble de terminaux, et une fonction poids w associée aux sommets non terminaux, le problème du séparateur de poids minimum consiste à partitionner V U T en k + 1 sous-ensembles {S, V1,..., Vk} tel qu'il n'y a aucune arête entre deux sous-ensembles différents Vi et Vj, chaque Vi contient exactement un terminal et le poids de S est minimum. Dans cette thèse, nous étudions le problème d'un point de vue polyèdral. Nous donnons deux formulations en nombres entiers pour le problème, et pour une de ces formulations, nous étudions le polyèdre associé. Nous présentons plusieurs inégalités valides, et décrivons des conditions de facette. En utilisant ces résultats, nous développons un algorithme de coupes et branchement pour le problème. Nous étudions également le polytope des séparateurs dans les graphes décomposables par sommets d'articulation. Si G est un graphe qui se décompose en G1 et G2, alors nous montrons que le polytope des séparateurs dans G peut être décrit à partir de deux systèmes linéaires liés à G1 et G2. Ceci donne lieu à une technique permettant de caractériser le polytope des séparateurs dans les graphes qui sont récursivement décomposables. Ensuite, nous étudions des formulations étendues pour le problème et proposons des algorithmes de génération de colonnes et branchement ainsi que des algorithmes de génération de colonnes, de coupes et branchement. Pour chaque formulation, nous présentons un algorithme de génération de colonnes, une procedure pour le calcul de la borne duale ainsi qu'une règle de branchement. De plus, nous présentons quatre variantes du problème du séparateur. Nous montrons que celles-ci sont NP-difficiles, et pour chacune d'elles nous donnons une formulation en nombres entiers et présentons certaines classes d'inégalités valides.

Mots Clés

Optimisation combinatoire, Polytope, Facette, Algorithme de coupes et branchements, Algorithme de génération de colonnes et branchements, Composition de polyèdres.

Abstract

Given a graph G = (V U T, E) with V U T the set of vertices, where T is a set of terminals, and a weight function w, associated with the nonterminal nodes, the multi-terminal vertex separator problem consists in partitioning V U T into k + 1 subsets {S, V1,..., Vk} such that there is no edge between two different subsets Vi and Vj, each Vi contains exactly one terminal and the weight of S is minimum. In this thesis, we consider the problem from a polyhedral point of view. We give two integer programming formulations for the problem, for one of them, we investigate the related polyhedron. We describe some valid inequalities and characterize when these inequalities define facets. Using these results, we develop a Branch-and-Cut algorithm for the problem. We also study the multi-terminal vertex separator polytope in the graphs decomposable by one node cutsets. If G is a graph that decomposes into G1 and G2, we show that the multi-terminal vertex separator polytope in G can be described from two linear systems related to G1 and G2. This gives rise to a technique for characterizing the multi-terminal vertex separator polytope in the graphs that are recursively decomposable. Moreover, we propose three extended formulations for the problem and derive Branch-and-Price and Branch-and-Cut-and-Price algorithms. For each formulation we present a column generation scheme, the way to compute the dual bound, and the branching scheme. Finally, we discuss four variants of the multi-terminal vertex separator problem. We show that all these variants are NP-hard and for each one we give an integer programming formulation and present some class of valid inequalities.

Keywords

Combinatorial optimization, Polytope, Facet, Branch-and-Cut algorithm, Branch-and-Price algorithm, Composition of polyhedra.