K-Separator problem
Mohamed Ahmed Mohamed Sidi

To cite this version:

HAL Id: tel-01239838
https://tel.archives-ouvertes.fr/tel-01239838
Submitted on 8 Dec 2015

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.
Doctoral School EDITE

Thesis submitted for obtaining the

PHD DEGREE IN COMPUTER SCIENCE

Doctorate jointly delivered by

Telecom SudParis and Pierre et Marie Curie University- Paris 6

Speciality:

COMPUTER SCIENCE

Presented by

Mohamed Ahmed MOHAMED SIDI

Title

K-Seperator Problem

Committee in charge:

Mohamed DIDI-BIHA  Reviewer  Université de Caen Basse Normandie
Eric ANGEL  Reviewer  Université d’Evry
Sourour ELLOUMI  Examiner  CNAM
Eric GOURDIN  Examiner  Orange Labs
Evripidis BAMPIS  Examiner  Université Paris 6
Lucas LETOCART  Examiner  Université Paris 13
José NETO  Co-advisor  Telecom SudParis
Walid BEN-AMEUR  Supervisor  Telecom SudParis

Thesis N°: 2014TELE0032
Titre

Problème de k-Séparateur

Soutenue le 04 Décembre 2014 devant le jury composé de :

Mohamed DIDI-BIHA  
Rapporteur  
Université de Caen Basse Normandie

Eric ANGEL  
Rapporteur  
Université d’Evry

Sourour ELOUMI  
Examinateur  
CNAM

Eric GOURDIN  
Examinateur  
Orange Labs

Evripidis BAMPIS  
Examinateur  
Université Paris 6

Lucas LETOCART  
Examinateur  
Université Paris 13

José NETO  
Co-Encadrant  
Telecom SudParis

Walid BEN-AMEUR  
Directeur de thèse  
Telecom SudParis

Thèse N°: 2014TELE0032
To My Wife Varha and My Daughter Ezza.
Acknowledgments

I would like to thank Professor Walid Ben-Ameur for his excellent support and dedication during all the time I spent working on this doctorate thesis. I am very grateful for the time he spent helping and guiding my researches. I would like also to thank him for granting me the freedom of developing my ideas and for his suggestions and advice throughout this work.

A special thanks to my co-adviser, M. José Neto, for his advice and guidance in preparing this thesis. I really learned many things by working with him and benefited from his experience as well as his excellent research and technical skills.

I would like also to thank my uncle, Professor Hadrami OULD SIDI MOHAMED who helped me and encouraged me. A particular thanks to him for reading my manuscript and for the suggestions he made to improve the quality of this work.

Additional thanks go to my thesis Evaluation Committee including Professor Mohamed DIDI-BIHA, Professor Eric ANGEL, Professor Evripidis BAMPIS, Professor Sourour ELLOUMI, Professor Lucas LETOCART, Doctor Eric GOURDIN, Associated professor José NETO and Professor Walid BEN-AMEUR.

Many thanks to all my friends and colleagues, particularly, Bakr Sarakbi, Wassim Drida, Mazen Al Maarabani, Farouk Aissanou, Khliifa Tomy, Fayeçal Bessayah, Makhlouf Hadji and Mohamed-Haykel Zayani. Thanks also to Mrs. Valérie MA-TEUS for her help and support in completing the non technical part of my work.

Last and not least, I would like to thank my mother, my grandmother, my father and all my brothers and sisters for their encouragement and support.
Given a vertex-weighted undirected graph $G = (V, E, w)$ and a positive integer $k$, we consider the $k$-separator problem: it 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 $k$. If $k = 1$ we get the classical vertex cover problem. The case $k = 2$ is equivalent to computing the dissociation number of a graph (in the case of unit weights). We prove that this problem can be solved in polynomial time for some graph classes including bounded treewidth, $mK_2$-free, $(G_1, G_2, G_3, P_6)$-free, interval-filament, asteroidal triple-free, weakly chordal, interval and circular-arc graphs. Different formulations are presented and compared. Polyhedral results with respect to the convex hull of the incidence vectors of $k$-separators are reported. Numerical results are reported and approximation algorithms are also presented.

**Keywords:** Graph partitioning, Complexity theory, Optimization, Approximation algorithms, Vertex separators, Polyhedral approach, Polynomial-time algorithms, Integer programming.
Résumé

Soit $G$ un graphe non orienté dont les sommets sont pondérés. Nous cherchons à calculer un sous-ensemble de sommets de poids minimal dont la suppression nous donne un graphe où la taille de chaque composante connexe est inférieure ou égale à un entier positif donné $k$. Ce problème est denommé Problème de $k$-Séparateur et le sous-ensemble recherché, $k$-Séparateur. Le problème de $k$-Séparateur a de nombreuses applications. Si les poids des sommets sont tous égaux à 1, la taille d’un $k$-séparateur peut être utilisée pour évaluer la robustesse d’un graphe ou d’un réseau. On peut citer d’autres applications du problème de $k$-Séparateur tel que : partitionnement de graphe et décompositions de matrice de contraintes etc ...

Si $k = 1$, nous obtenons le problème classique Vertex Cover. De nombreuses formulations sont proposées pour ce problème dans notre thèse. Les relaxations linéaires de ces formulations sont comparées. Une étude polyédrale est proposée (inégalités valides, facettes et algorithmes de séparation). Des cas où le problème peut être résolu en temps polynomial sont présentés. Entre autres, le cas de chemins, de cycles, d’arbres, et plus généralement les graphes avec largeur arborescente bornée ainsi que des graphes ne contenant pas certains graphes particuliers comme sous graphes induits. Des algorithmes d’approximation de rapport $(k+1)$ sont également exposés et quelques résultats d’inapproximabilité. La plupart des algorithmes sont implémentés et comparés.

Mots Clés: Couverture par des sommets, Méthode de coupe, Problème de séparateur, Approches polyédrales, Algorithmes d’approximation.
# Contents

1 Introduction 13

1.1 General Context .......................... 13
1.2 Notation .................................. 15
1.3 Thesis plan ................................ 16

2 Related work 17

2.1 Introduction .............................. 18
2.2 Related problems to $k$-separator problem .................. 18

2.2.1 Vertex Cover problem .................... 18
2.2.2 Stable Set problem ........................ 19
2.2.3 Maximal Clique problem .................. 19
2.2.4 Hitting Set problem ....................... 19
2.2.5 Set Cover problem ........................ 19
2.2.6 Capacitated Vertex Cover problem .......... 20
2.2.7 Dissociation Set problem .................. 20

2.2.7.1 $K_{1,4}$-free bipartite graphs ............. 22
2.2.7.2 $C_4$-free bipartite graphs ............. 22
# Contents

4 Integer Formulations 71

4.1 Introduction ........................................ 71
4.2 Basic formulation ................................. 72
4.3 Stable set formulations ......................... 72
4.4 Metric formulations .............................. 73
4.5 Projected metric formulation .................. 75
4.6 Partitioning formulations ...................... 77
4.7 Conclusion ........................................ 79

5 Polyhedral study of $S_k(G)$ 80

5.1 Introduction ........................................ 80
5.2 Some general properties ....................... 81
5.3 The path and cycle cases ..................... 83
5.4 Valid inequalities for $S_k(G)$ ............... 86
  5.4.1 Hitting set inequalities .................... 86
  5.4.2 Connectivity inequalities .................. 87
  5.4.3 Cycle inequalities .......................... 88
  5.4.4 Wheel inequalities ........................ 90
  5.4.5 Antiweb inequalities ....................... 91
  5.4.6 Generalized projected metric inequalities .... 93
5.5 Conclusion ........................................ 95

6 Computational Results 96

6.1 Introduction ....................................... 96
6.2 Cutting-plane algorithms .................... 97
6.2.1 A cutting-plane algorithm related to the stable set formulation

\[ IP2 \] 
97

6.2.2 A cutting-plane algorithm related to the partitioning formulation \[ IP9 \] 
97

6.2.3 A cutting-plane algorithm related to formulations \[ IP1 \] and \[ IP7 \] 
98

6.3 Experimental results 
99

6.4 Conclusion 
105

7 Approximations

7.1 Introduction 
106

7.2 Approximation algorithms 
106

7.2.1 LP-Based approximation algorithms 
106

7.2.2 Primal Dual approach 
107

7.2.3 Greedy approximation algorithm 
108

7.3 Inapproximability 
108

7.4 Conclusion 
109

8 Conclusion 
110

A Résumé du manuscrit de thèse en français

A.1 Introduction 
111

A.2 Problématique 
112

A.3 Propositions 
112

A.3.1 Approches Polyédrales 
113

A.3.1.1 Cas polynomiaux 
113

A.3.1.1.a Graphes avec largeur arborescente bornée 
113
A.3.1.1.b Graphes sans couplage $mK_2$ induit . . . . . . 114
A.3.1.1.c Graphes sans $(G_1, G_2, G_3, P_6)$ induits . . . . 115
A.3.1.1.d Graphes de type Interval-filaments, Graphes
sans asteroidal triple et Graphes de type weakly
chordal . . . . . . . . . . . . . . . . . . . . . . . . . 115
A.3.1.1.e Graphes à intervalles et arc-circulaire . . . . 116
A.3.1.2 Formulations . . . . . . . . . . . . . . . . . . . . . . 116
A.3.1.2.a Formulation de base . . . . . . . . . . . . . 116
A.3.1.2.b Formulation de Stable de poids maximal . . . 117
A.3.1.2.c Formulation métrique . . . . . . . . . . . . . 117
A.3.1.2.d Formulation métrique projetée . . . . . . . 119
A.3.1.2.e Formulation de partitionnement . . . . . . . 119
A.3.2 Algorithmes d’approximation . . . . . . . . . . . . . . . . . . 121
A.4 Conclusion & Perspective . . . . . . . . . . . . . . . . . . . . . . 121

Bibliography 123
List of Figures

2.1 Maximal dissociation sets of graph $P_5$ [90] ................................. 21
2.2 $K_{1,4}$ bipartite graph ......................................................... 22
2.3 $K_{2,2}$ bipartite graph in the left and $C_4$ in the right .................. 23
2.4 Example of planar graph ......................................................... 23
2.5 Example of line graph ............................................................ 23
2.6 Some $P_k$ and $K_{1,n}$ graphs .................................................. 24
2.7 Left : web $W^3_{10}$ and right :antiweb $AW^3_{10}$ .......................... 35
2.8 1-wheel graph [12] ............................................................... 35
2.9 Inheritance of valid inequalities [53] ............................................ 43
2.10 Inheritance of facets inequalities [53] ........................................ 43
3.1 The graphs $G_1$, $G_2$, $G_3$ and $P_6$ ........................................ 60
3.2 Example of an interval-filament graph ........................................ 65
3.3 Asteroidal triple-free graph ....................................................... 65
3.4 Chordal and weakly chordal graphs .......................................... 66
3.5 Example of an interval and circular-arc graphs .............................. 67
3.6 On the dynamic programming approach to solve problem (3.5) . . . 69
List of Figures

5.1 Illustration of inequality (5.6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
## List of Tables

2.1 Complexity of MDS, MIM and MIS [90] .......................... 21

6.1 Compact Formulation (IP2) applied to random graphs ............. 99
6.2 IP1 and IP7 formulations applied to random graphs ............... 101
6.3 IP1 and IP7 formulations applied to NETLIB instances ........... 102
6.4 IP9 formulation applied to NETLIB instances ..................... 102
6.5 IP1 and IP7 formulations applied to MIPLIB instances .......... 103
6.6 IP9 formulation applied to MIPLIB instances ..................... 104
Chapter 1

Introduction

Contents

1.1 General Context ................................................. 13

1.2 Notation ....................................................... 15

1.3 Thesis plan .................................................... 16

1.1 General Context

Given a vertex-weighted undirected graph $G = (V, E, w)$, the minimum vertex cover problem consists in computing a minimum-weight set of vertices $S \subset V$ such that $V \setminus S$ is a stable set. A minimum-weight vertex cover can then be exhibited if one can find a maximum-weight stable set. While the problem can be solved in polynomial time in some cases (bipartite graphs, perfect graphs, etc.), it is known to be generally NP-hard (see, e.g., [4, 76]). Many valid inequalities are known for the vertex cover problem and the stable set problem. A 2-approximation algorithm for the vertex cover problem is given by a simple greedy algorithm (see, e.g., [4]).

Let $k$ be a positive number, we consider the following natural generalization of the vertex cover problem. We want to compute a minimum-weight subset of vertices $S$ whose removal leads to a graph where the size of each connected component is less than or equal to $k$. Let us call such a set a $k$-separator. If $k = 1$ we get
1.1. General Context

the classical vertex cover problem. The case \( k = 2 \) is equivalent to compute the
dissociation number of a graph (in the case of unit weights) [91]. This problem is
NP-hard even if the graph is bipartite.

The \( k \)-separator problem has many applications. If vertex weights are equal
to 1, the size of a minimum \( k \)-separator can be used to assess the robustness of
a graph or a network. Intuitively, a graph for which the size of the minimum \( k \)-
separator is large, is more robust. Unlike the classical robustness measure given by
the connectivity, the new one seems to avoid the underestimate of robustness when
there are only some local weaknesses in the graph. Consider for example a graph
containing a complete subgraph and a vertex connected to exactly one vertex of the
subgraph. Then the vertex-connectivity of this graph is 1 while the graph seems
to be robust everywhere except in the neighborhood of one vertex. The size of a
minimum \( k \)-separator of this graph is \(|V| - 1 - k|\).

The minimum \( k \)-separator problem has some other network applications. A classical
problem consists in partitioning a graph/network into different subgraphs with
respect to different criteria. For example, in the context of social networks, many
approaches are proposed to detect communities. By solving a minimum \( k \)-separator
problem, we get different connected components that may represent communities.
The \( k \)-separator vertices represent people making connections between communities.
The \( k \)-separator problem can then be seen as a special partitioning/clustering graph
problem.

Computing a \( k \)-separator can also be useful to build algorithms based on divide-and-conquer approaches. In some cases, a problem defined on a graph can be de-
composed into many subproblems on smaller subgraphs obtained by the deletion of
a \( k \)-separator (see, e.g., [77]).

The \( k \)-separator problem is closely related to the vertex-separator problem where
we aim to remove a minimum-weight set of vertices such that every connected com-
ponent in the remaining graph has a size less than \( \alpha|V| \) (for a fixed \( \alpha < 1 \)). A
polyhedral study of this problem is proposed in [21] (see also the references therein).
When the vertex-separator problem is considered, the graph is generally partitioned
into 3 subgraphs: the separator, and two subgraphs each of size less than \( \alpha|V| \). The
philosophy is different in the case of the \( k \)-separator where the graph is partitioned
into many components each one having a size less than \( k \).

The \( k \)-separator problem was considered in one published paper [53] where it was
Chapter 1. Introduction

presented as a problem of disconnecting graphs by removing vertices. An extended formulation is proposed in [53] with some polyhedral results. Some other applications were also mentioned in [53]. This includes a constraint matrix decomposition where each row $A_i$ of a matrix $A$ is represented by a vertex $v_i$ and two vertices $v_i$ and $v_j$ are adjacent if there is at least one column $h$ with nonzero coefficients in the corresponding two rows ($a_{ih} \neq 0$ and $a_{jh} \neq 0$). The problem is to assign as many rows as possible to the so-called blocks such that no more than $k$ rows are assigned to the same block, and rows assigned to different blocks are not connected (i.e., there is no any column $h$ such that $a_{ih}a_{jh} \neq 0$ if $A_i$ and $A_j$ are in different blocks) [67]. This matrix decomposition may help the solution process of linear or integer programs where the constraint matrix is defined by $A$.

Another application is related to the field of group technology (see [53] for details).

1.2 Notation

Given an undirected graph $G = (V, E)$ and a vertex subset $U \subset V$, the complement of $U$ in $G$, i.e. the vertex set $V \setminus U$ is denoted $\overline{U}$. The set of vertices (resp. edges) of the graph $G$ may also be denoted by $V(G)$ (resp. $E(G)$). An edge $e \in E$ with endnodes $u$ and $v$ is denoted by $(u, v)$. For a vertex-weighted undirected graph $G = (V, E, w)$, $w_v$ denotes the weight of the vertex $v \in V$.

Given a vertex subset $S \subset V$, the set of vertices in $S$ that are adjacent to at least one vertex in $S$ is denoted $N(S)$. $N_S(k)$ denotes the set of neighbors of a vertex $k$ in subset $S$. Given two subsets of vertices $A$ and $B$, $A$ and $B$ are adjacent if either $A \cap B \neq \emptyset$ or $N(A) \cap B \neq \emptyset$.

Given a subset of vertices $S \subset V$, $\chi^{(S)} \in \{0, 1\}^n$ denotes the incidence vector of $S$, with $n = |V|$. The convex hull of all the incidence vectors of $k$-separators in the graph $G$ is indicated by $S_k(G)$. We also use $G(S)$ to refer to the subgraph of $G$ that is induced by a subset of vertices $S \subset V$.

The order of a graph indicates its number of vertices. $K_n$ denotes a complete graph with order $n$. Given some integer $m$, $mK_2$ denotes a matching with $m$ edges. If $p$ and $q \leq p$ are two positive integer, then \( \binom{p}{q} = \frac{p!}{q!(p-q)!} \).

If the graph $G$ does not contain an induced subgraph isomorphic to some given graph $H$, then we say that $G$ is $H$-free.
1.3 Thesis plan

If $G$ is a simple path with vertex set $\{v_1, \ldots, v_n\}$ and edge set $\{(v_i, v_{i+1}) : i = 1, \ldots, n - 1\}$, then the notation $[v_i, v_j]$ (resp. $[v_i, v_j]$, $[v_i, v_j]$, $[v_i, v_j]$) with $i < j$, $i, j \in \{1, \ldots, n\}$ stands for the vertex set $\{v_i, v_{i+1}, \ldots, v_j\}$ (resp. $\{v_{i+1}, \ldots, v_{j-1}\}$, $\{v_i, v_{i+1}, \ldots, v_{j-1}\}$, $\{v_{i+1}, \ldots, v_j\}$). The set of all the simple paths joining $i$ and $j$ will be denoted $P_{ij}$. Given a simple path $p$ joining $i$ and $j$, $x(p)$ stands for the sum of the $x_v$ values over all vertices belonging to $p$ (including $i$ and $j$). Let $N$ denote the set of natural numbers.

1.3 Thesis plan

This thesis is organized as follows:

1. In chapter 2, we present the state of the art, precisely related works and the used technics.

2. In chapter 3, some cases where the problem can be solved in polynomial time are shown.

3. In chapter 4, we describe integer programming formulations of the $k$-separator problem. The linear relaxations of these formulations are also compared when this is possible.

4. A polyhedral study of the convex hull of the incidence vectors of $k$-separators is proposed in chapter 5.

5. Some numerical experiments follow in chapter 6.

6. In chapter 7, we present some approximation algorithms.

7. The final chapter of this thesis concludes our work. We summarize our contributions and present some perspectives and possible future directions to extend our work.
Chapter 2

Related work

Contents

2.1 Introduction .................................................. 18
2.2 Related problems to $k$-separator problem .................. 18
  2.2.1 Vertex Cover problem .................................. 18
  2.2.2 Stable Set problem .................................... 19
  2.2.3 Maximal Clique problem .............................. 19
  2.2.4 Hitting Set problem .................................. 19
  2.2.5 Set Cover problem .................................... 19
  2.2.6 Capacitated Vertex Cover problem .................... 20
  2.2.7 Dissociation Set problem ............................. 20
    2.2.7.1 $K_{1,4}$-free bipartite graphs .................. 22
    2.2.7.2 $C_4$-free bipartite graphs ...................... 22
    2.2.7.3 Planar graphs .................................. 22
    2.2.7.4 Line graphs .................................... 23
    2.2.7.5 $(P_k,K_{1,n})$-free graphs ....................... 23
2.3 Used methods and techniques ............................... 27
  2.3.1 The primal-dual method .............................. 27
  2.3.2 The Rounding Approach ................................ 30
  2.3.3 The Greedy Method .................................. 31
  2.3.4 Polyhedral Approach ................................ 32
2.4 Further connections between the $k$-separator problem and other problems ............................. 36
2.1 Introduction

Combinatorial optimization is a well-known field of applied mathematics, combining techniques from combinatorics, linear programming, and the theory of algorithms, to solve optimization problems over discrete structures [85]. One of the most studied problems in combinatorial optimization is the vertex cover problem [15]. For a decade there has been an increasing interest to generalize this issue to another one [45]. This chapter provides some state of art techniques used to solve some classical optimization problems that have a relation with the $k$-separator problem [83, 84]. It demonstrates also some related works to our thesis main problem, i.e. $k$-separator problem. This chapter is organized as follows: In section 2.2 we introduce many classical combinatorial optimization problems close to the $k$-separator problem [84]. In section 2.3 we mention the techniques used in this thesis. For the sake of clarity we present disconnecting graphs and vertex separator problems in section 2.4 after we have shown the polyhedral method in 2.3.4. Finally, in section 2.5 we conclude this chapter.

2.2 Related problems to $k$-separator problem

2.2.1 Vertex Cover problem

Given a graph $G$, we search a minimum size set of vertices such that, for every edge, at least one of the endpoints that belongs to this set. In the weighted version of vertex cover, each vertex has a weight. We are looking for the minimum total weight set of vertices with the property given earlier [42]. In other words, given $G(V, E)$ with weights $w_i \geq 0$ for all vertices $i \in V$, we must select a minimum weight vertex cover. Observe that a vertex cover is a $k$-separator for $k = 1$. 

18
2.2.2 Stable Set problem

A stable (or independent) set problem deals with a set of vertices where no two of them are adjacent. In other words, each edge in the graph has at most one endpoint in this set (or a set of pairwise nonadjacent nodes). If the graph is weighted, we aim to compute a maximum-weight stable set. Notice that the complementary of such a maximum stable set is a minimum-weight vertex cover.

2.2.3 Maximal Clique problem

A clique is a complete subgraph, i.e. all nodes are connected to each other. A maximal clique is a biggest one. The maximal clique problem is aimed at computing a maximal clique in a given graph. When the graph is weighted, a maximum-weight clique is obviously a maximum stable set in the complementary graph.

2.2.4 Hitting Set problem

Given a set $A = \{a_1, \ldots, a_n\}$, a collection $B_1, B_2, \ldots, B_m$ of subsets of $A$. A hitting set is defined as a set $H \subset A$, if $H \cap B_i \neq \emptyset$ for $1 \leq i \leq m$. We can observe that a vertex cover is a hitting set with each subset reduced to an edge.

As mentioned above, the $k$-separator problem is a natural generalization of the vertex cover problem. However, there are other possible extensions of the vertex cover problem. Two of them are described below.

2.2.5 Set Cover problem

Given a set of elements $E = \{e_1, e_2, \ldots, e_n\}$ and a set of $m$ subsets of $E$, $S = \{S_1, S_2, \ldots, S_m\}$, the set cover problem is to find a minimum size collection $C$ of sets from $S$ such that $C$ covers all elements in $E$ (i.e., such that $\bigcup_{S_i \in C} S_i = E$). In the weighted version, a weight $w_j$ is associated to each subset $S_j$ and we aim to compute a minimum weight collection covering $E$. If $E$ is the set of edges of a weighted graph, and $S_v$ is the set of edges incident to vertex $v$, then we get the classical vertex cover problem.
2.2. Related problems to $k$-separator problem

2.2.6 Capacitated Vertex Cover problem

Let $G = (V, E)$ be an undirected graph, $V = \{1, 2, ..., n\}$ be a vertex set and $E$ be an edge set. Let $w_v$ denote the weight of vertex $v$ and $k_v$ denote its capacity. As defined in [72] a capacitated vertex cover is a function $x : V \to \mathbb{N}$ such that there exists an orientation of the edges of $G$ in which the number of edges directed into vertex $v \in V$ is at most $x_v k_v$. These edges are said to be covered by $v$. The weight of cover is $\sum_{v \in V} x_v w_v$. The minimum capacitated vertex cover problem wants to compute the minimum capacitated vertex cover. The main idea of [72] is to use a rounding technique to improve approximation algorithms for this problem. It can be seen that if $k_v = |V| - 1$ for every $v \in V$, the problem is reduced to the minimum weight vertex cover. The problem is NP-hard since it generalizes a NP-hard problem.

In section 2.2.7 we draw a relationship between dissociation set and $k$-separator problems.

2.2.7 Dissociation Set problem

When weights are unitary and $k = 2$, the $k$-separator problem is equivalent to compute the dissociation number of the graph [91]. A subset of vertices in a graph $G$ is called a dissociation set if it induces a subgraph so that each vertex has degree at most 1, and the dissociation number is the size of a largest dissociation set. A dissociation set $D$ is maximal if they are not containing in any another dissociation set in $G$ [90]. An example of maximal dissociation set is shown in figure 2.1 as a set of encircled vertices. A minimum maximal dissociation number is defined by $\text{diss}^-(G) = \min\{|D| : D \in DS(G)\}$ [90]

And a maximum dissociation number is given by $\text{diss}^+(G) = \max\{|D| : D \in DS(G)\}$ [90]

where $DS(G) = \{S \subset V : S \text{ is a maximal dissociation set in } G\}$

A maximum dissociation set is a dissociation set that contains $\text{diss}^+(G)$ nodes and the minimum maximal dissociation set is a maximal dissociation set that contains $\text{diss}^-(G)$ vertices [90]. In figure 2.1 $\text{diss}^+(P_3) = 4$ and $\text{diss}^-(P_3) = 3$. $\text{diss}^+(G)$ is a lower bound for the 1-improper chromatic number of a graph $G$ [69]. The dissociation set problems refer to maximum dissociation set problem and minimum
Chapter 2. Related work

Figure 2.1: Maximal dissociation sets of graph $P_5$ [90]

maximal dissociation set problem [90]. The first problem is a maximum dissociation set problem ($MDS$), it can be announced as follows: given a graph $G$ and an integer $k$, does there exist a dissociation set $D$ in $G$ such that $|D| \geq k$ (i.e., $diss^+(G) \geq k$) [90]? This problem has been introduced for the first time by Yannakakis in [91].

The second problem is minimum maximal dissociation set with the same input as $MDS$, it is resumed to the question: Is there a maximal dissociation set $D$ in $G$ such that $|D| \leq k$ (i.e., $diss^-(G) \leq k$) [90]? The $MDS$ is close to maximum independent set ($MIS$) and maximum induced matching ($MIM$) problems. The maximum cardinality of a stable (independent) set of $G$, let $\alpha(G)$ be this number, is called the independent number. The maximum cardinality of an induced matching of $G$ is called the induced matching number, and it is denoted by $\Sigma(G)$. The decision maximum independent set problem is defined by: given a graph $G$ and an integer $k$, is $\alpha(G) \geq k$?, and the decision problem of maximum induced matching is described by: given a graph $G$ and an integer $k$, is $\Sigma(G) \geq k$?

<table>
<thead>
<tr>
<th>Table 2.1: Complexity of MDS, MIM and MIS [90]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph classes/Problems</strong></td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>Triangle-free graphs</td>
</tr>
<tr>
<td>Claw-free graphs</td>
</tr>
<tr>
<td>Weakly chordal graphs</td>
</tr>
<tr>
<td>Circular graphs</td>
</tr>
</tbody>
</table>
The following inequalities hold for any graph $G$: $\alpha(G) \leq \text{diss}^+(G)$ and $2\Sigma(G) \leq \text{diss}^+(G)$ \[90\]. We have also in the reference \[90\] for any positive integer $r$:

\[
diss^+(H_r - \alpha(H_r)) = r, \quad \text{diss}^+(H_r) = 4r \quad \text{and} \quad \alpha(H_r) = 3r,
\]

where $H_r$ is the graph formed by identifying one vertex from $r$ copies of cycle $C_7$. For the graph $K_{1,r+2}$, we have:

\[
diss^+(K_{1,r+2}) - 2\Sigma(K_{1,r+2}) = r, \quad \text{diss}^+(K_{1,r+2}) = r + 2 \quad \text{and} \quad \Sigma(K_{1,r+2}) = 1 \[90\].
\]

Before presenting these results, let’s us recall some definitions.

### 2.2.7.1 $K_{1,4}$-free bipartite graphs

A graph is called $K_{1,4}$-free bipartite graph if and only if it is a bipartite graph and does not contain a complete bipartite graph or biclique $K_{1,4}$ \[66\], see figure 2.2.

### 2.2.7.2 $C_4$-free bipartite graphs

A graph $G$ is said to be a $C_4$-free bipartite graph if it does not contain a subgraph isomorphic to $C_4$ \[66\] (see figure 2.3) and it is a bipartite graph. Note that $C_4$ and $K_{2,2}$ are isomorphic graphs.

### 2.2.7.3 Planar graphs

A graph is planar if it is isomorphic to a plane graph \[19\]. In other words, If a graph can be drawn without edges crossing except at endpoints. See figure 2.4 for an example of planar graph.

![Figure 2.2: $K_{1,4}$ bipartite graph](image-url)
Chapter 2. Related work

2.2.7.4 Line graphs

The line graph \( L(G) = (V(L(G)), E(L(G))) \) of graph \( G = (V, E) \) is defined as another graph that represents the adjacencies between edges of \( G \). \( V(L(G)) = E \) and \( (u', v') \in E(L(G)) \) if \( u' \) and \( v' \) have a common vertex in \( G \). This class of graph was introduced in [26]. Figure 2.5 shows a construction of a sample line graph.

2.2.7.5 \((P_k, K_{1,n})\)-free graphs

A graph that does not contain an induced subgraph \( P_k \) and \( K_{1,n} \) [81]. Figure 2.6 shows some examples.
2.2. Related problems to \( k \)-separator problem

![Figure 2.6: Some \( P_k \) and \( K_{1,n} \) graphs](image)

Let’s now go back to the dissociation set problem.

The table 2.1 shows the complexity of MDS, MIM and MIS. Some polynomial (P) and NP-Complete (NP-c) cases have been identified. Some cases where the question of complexity still open (?) is presented also. As shown in table 2.1 the MDS problem is NP-complete for line graphs [90]. We have this result by a polynomial time reduction from a variant of partition into isomorphic subgraphs problem [17]. A partition into isomorphic subgraphs problem is defined by: given graphs \( G \) and \( H \) with \( |V(G)| = q|V(H)| \) where \( q \) is a positive integer, the problem can be posed as follow: does there exist a partition \( \bigcup_{i=1}^{q} V_i \) of \( V(G) \) such that \( G(V_i), \forall i = 1, \ldots, q \), contains a subgraph isomorphic to \( H \) ?

Theorem 2.1 shows the complexity of MDS problem in the case of line graphs

**Theorem 2.1** [90] Maximum dissociation set is a NP-complete problem for line graphs.

The proof of the theorem 2.1 uses the lemma 2.1.

**Lemma 2.1** [92] Partition into subgraphs isomorphic to \( P_3 \) is an NP-complete problem for planar bipartite graphs of a maximum vertex degree of 4 in which every vertex of degree 4 is a cut-vertex.

Let \( \alpha_w(G) \) denote the weight of a maximum weight independent set of \( G \) in the case of maximum weight independent set problem. The idea presented in [81] and also used in chapter 3 consists in a construction of an extended graph \( G^* \) from the graph \( G \) such that the MDS problem in \( G \) becomes equivalent to maximum weight independent set problem in \( G^* \). The transformation is described as follow: given a graph \( G(V, E) \), let \( G^*(V^*, E^*) \) be a graph defined by:

- \( V^* = V \cup E \)
- \( (u^*, v^*) \in E^*, if : \)

24
Chapter 2. Related work

1. $u^*, v^* \in V$ and $(u^*, v^*) \in E$
2. $u^* \in V$, $v^* = (x, y) \in E$ and $N_G(u^*) \cap \{x, y\} \neq \emptyset$
3. $u^* = (x, y) \in E$, $v^* = (z, t) \in E$ and $N_G(x) \cap \{z, t\} \neq \emptyset$

Lozin et al. [81] have shown that the following lemma 2.2 holds.

**Lemma 2.2** [81] An independent set of maximum weight in $G^*$ corresponds to a maximum dissociation set in $G$. In particular, $\alpha_w(G^*) = \text{diss}^+(G)$.

By the same construction as the one presented above [81] we have this important theorem 2.2.

**Theorem 2.2** [90] The graph $G^*$ of a graph $G$ is chair-free if and only if $G$ is $(G_1, G_2, G_3)$-free (for $G_1$, $G_2$ and $G_3$ graphs see section 3.5)

The following theorem 2.3 proved in the reference [82] by using the method of modular decomposition [2] is needed to prove theorem 2.4.

**Theorem 2.3** [82] The maximum weight independent set problem can be solved in polynomial time in the class of chair-free graphs.

Lemma 2.1 and theorems 2.2 and 2.3 imply the theorem 2.4

**Theorem 2.4** [90] The maximum dissociation set problem can be solved in polynomial time in the class of $(G_1, G_2, G_3)$-free graphs.

Another result is given by theorem 2.5 for the $mK_2$-free graphs (see section 3.4 for detail).

**Theorem 2.5** [90] Let $m \geq 2$ be an integer. The graph $G^*$ of a graph $G$ is $mK_2$-free if and only if $G$ is $mK_2$-free.

For some classes of graphs, the complexity of finding the maximum dissociation number can be specified [90]. The theorem 2.6 concerns the case of graphs containing a Hamiltonian path.

**Theorem 2.6** [90] Let $G$ be a graph with $n$ vertices and containing a Hamiltonian path. Then
\[ \text{diss}^+(L(G)) = \lfloor \frac{2n}{3} \rfloor. \]

Theorem 2.7 is focused on the weakly chordal graphs.
2.2. Related problems to \(k\)-separator problem

**Theorem 2.7** [90] Minimum maximal dissociation set is NP-complete for weakly chordal graphs.

In order to give an inapproximability result related to the problem of computing the dissociation number Orlovich et all. in [90] start with the lemma 2.3.

**Lemma 2.3** [90] For each instance \((C,X)\) of 3-SAT with a set \(C\) of \(m\) clauses and a set \(X\) of \(n\) variables and for each integer \(t\), there exists a bipartite graph \(G\) on \(3n+2tn(n+m)\) vertices such that the following property holds for the minimum maximal dissociation number:

\[
diss^\mathrm{G} \begin{cases} 
\leq 2n, & \text{if } C \text{ is satisfiable} \\
> 2nt, & \text{if } C \text{ is not satisfiable}
\end{cases}
\]

By using the result of lemma 2.3 we find in [90] the following theorem 2.8 in the case of bipartite graphs for the minimum maximal dissociation set problem.

**Theorem 2.8** [90] Assuming that \(P \neq NP\), minimum maximal dissociation set for bipartite graphs cannot be approximated in polynomial time within a factor of \(p^{1-\varepsilon}\) for any constant \(\varepsilon > 0\), where \(p\) denotes the number of vertices in the input graph.

And then [90] gives also an inapproximate result (theorem 2.9) for the maximum dissociation set problem.

**Theorem 2.9** [90] Assuming that \(P \neq NP\), maximum dissociation set cannot be approximated in polynomial time within a factor of \(p^{1-\varepsilon}\) for any constant \(\varepsilon > 0\), where \(p\) is the number of vertices in the input graph.

Thus, computing the dissociation number is NP-hard if the graph is bipartite [91]. The NP-hardness still holds for \(K_{1,4}\)-free bipartite graphs [66], \(C_4\)-free bipartite graphs with a maximum vertex degree of 3 [66], planar graphs with a maximum vertex degree of 4 [11], and line graphs [90]. Several cases where the dissociation problem can be solved in polynomial time have been shown in the literature: chordal and weakly chordal graphs, asteroidal triple-free graphs [39], \((P_k, K_{1,n})\)-free graphs (for any positive numbers \(k\) and \(n\)) [81] and \((G_1, G_2, G_3)\)-free graphs [90]. The graphs mentioned here are defined in the cited references and are also recalled in chapter 3.
Chapter 2. Related work

2.3 Used methods and techniques

This section is organized as follows: In 2.3.1 we introduce the primal-dual method with application on minimum-weight vertex cover. In 2.3.2 we present the rounding approach. In 2.3.3 we present that the greedy approach can be beneficial to our case. Finally, in 2.3.4 we show the polyhedral approach and an application on the stable set problem.

2.3.1 The primal-dual method

The primal-dual method is one of the oldest techniques used by many researchers, where a good overview methods can be found in [86]. It was proposed by Dantzig, Ford and Fulkerson for the first time [27]. D. Williamson gives in [87] a good survey for some NP-hard problems where he used the primal-dual method. First, we will define what a primal-dual method is, and then we will apply it to the minimum weight vertex cover problem. Consider a general linear program (LP) formulation [27]:

\[
\begin{align*}
\text{min } cx \\
Ax & \geq b \\
x & \geq 0
\end{align*}
\]

Its DUAL is

\[
\begin{align*}
\text{max } yb \\
yA & \leq c \\
y & \geq 0
\end{align*}
\]

And Complementary Slackness Conditions (CS)

\[
\begin{align*}
\text{PRIMAL: } x_i > 0 \Rightarrow \sum_j y_j a_{ji} = c_i \\
\text{DUAL: } y_j > 0 \Rightarrow \sum_i a_{ij} x_i = b_j
\end{align*}
\]

The idea to solve (LP) is: If \( x \) and \( y \) are optimal for the primal and the dual respectively, then they satisfy \( cx = yb \) and also they satisfy PCS (Primal CS) and DCS (Dual CS).

We now present how the primal-dual method can be applied to the hitting set problem in order to give us an approximation algorithm. Given a ground set of elements
2.3. Used methods and techniques

Algorithm 1 Solve LP Problem by primal-dual \[27\]

\[\begin{align*}
    y & \leftarrow 0. \\
    \text{while} & \text{ Does not exist feasible } x \text{ satisfying CS do} \\
    & \text{ Increase the dual as much as possible and still maintaining dual feasibility.} \\
    \text{end while} \\
    & \text{Return } x.
\end{align*}\]

For a set \(E\), nonnegative costs \(C_e\), \(\forall e \in E\), and subsets \(T_1, \ldots, T_p \subset E\), we want to find a minimum-cost subset \(A \subset E\) so that \(A\) has a nonempty intersection with each subset \(T_i\). Such subset is called a hitting set, \(A\), for subsets \(T_i\) for \(i \in \{1, \ldots, p\}\). In \[87\] we find an algorithm to select a set \(A\), see algorithm 2.

Algorithm 2 Algorithm to Select a subset \(A\) \[87\]

\[\begin{align*}
    y & \leftarrow 0. \\
    A_1 & \leftarrow \emptyset. \\
    l & \leftarrow 1 \ (l \text{ is a counter}). \\
    \text{while} & \text{ } A_l \text{ is not feasible do} \\
    & \text{ Choose a subset } V_l \text{ of violated sets} \\
    & \text{ Increase } y_k \text{ uniformly for all } T_k \in V_l \text{ until } \exists e_l \notin A_l \text{ such that } \sum_{i: e_l \in T_i} y_i = C_{e_l}. \\
    & A_l \leftarrow A_l \cup \{e_l\}. \\
    & l \leftarrow l+1. \\
    & A' \leftarrow A_{l-1} \\
    \text{for } j & \leftarrow l-1 \text{ down to } 1 \text{ do} \\
    & \text{ if } A' - \{e_j\} \text{ is still feasible then} \\
    & \quad A' \leftarrow A' - \{e_j\} \\
    & \text{ end if} \\
    \text{end for} \\
    \text{end while} \\
    & \text{Return } A'.
\end{align*}\]

In which the elements of \(A\) were added) of not needed elements in a given feasible solution \(A\). In other words, once a feasible solution \(A\) has been obtained, we should examine the elements of \(A\) and delete any that are not needed for a feasible solution \[87\]. Let \(A_l\) be the set of elements in \(A\) at the beginning of the \(l^{th}\) iteration, let \(e_l\) be the element added in the \(l^{th}\) iteration, and let \(A'\) be the final set returned by the algorithm 2 \[87\]. We start by an empty set \(A_1\). Then we loop until we find a feasible solution. In each iteration \(l\), we choose a subset of violated subset \(V_l\), a set \(T_k\) is violated if \(T_k \cap A_l = \emptyset\), and then we increase \(y_k\) for all \(T_k \in V_l\) until \(\exists e_l \notin A_l\) such that \(\sum_{i: e_l \in T_i} y_i = C_{e_l}\). Finally, we start the deletion step by removing from \(A'\) the not necessary elements and still maintain it feasible.

To illustrate the primal-dual method, we consider the minimum-weight vertex
Chapter 2. Related work

cover problem. The following integer program (IP 2.3) describes the problem:

\[ \begin{align*}
\text{IP 2.3} \quad & \min \sum_{i \in V} w_i x_i \\
\text{Subject to:} \quad & x_i + x_j \geq 1 \forall (i, j) \in E \\
& x_i \in \{0, 1\} \forall i \in V
\end{align*} \]

For the linear relaxation "\( x_i \in \{0, 1\} \)" is replaced by \( x_i \in [0, 1] \).

The dual program is:

\[ \begin{align*}
\text{LP 2.4} \quad & \max \sum_{(i,j) \in E} y_{(i,j)} \quad (1) \\
\text{Subject to:} \quad & \sum_{k: (i,k) \in E} y_{(i,k)} \leq w_i \forall i \in V \\
& y_{(i,j)} \geq 0 \forall (i, j) \in E
\end{align*} \]

The primal-dual algorithm begins with the dual feasible solution in which all \( y \) variables are set to 0, and a primal infeasible solution in which all \( x \) variables are set to 0. If there exists some uncovered edge \((i,j)\) for which \( x_i + x_j = 0 \), we increase its corresponding dual variable \( y_{(i,j)} \) as much as possible and maintaining dual feasibility, so that the dual constraint (1) becomes tight, i.e.

\[ \sum_{k: (i,k) \in E} y_{(i,k)} = w_i \Rightarrow x_i = 1 \]

or

\[ \sum_{k: (j,k) \in E} y_{(j,k)} = w_j \Rightarrow x_j = 1 \]

Eventually we achieve a primal feasible solution \( x \) such that

\[ \sum_{i \in V} w_i x_i = \sum_{i \in V} (\sum_{k: (i,k) \in E} y_{(i,k)}) x_i. \]

Define \( S = \sum_{i \in V} (\sum_{k: (i,k) \in E} y_{(i,k)}) x_i. \)

Because for each edge \((i, j)\) \( E \), we have two features \( y_{(i,j)} x_i \) and \( y_{(j,i)} x_j \) in the summation, and since \( y_{(i,j)} = y_{(j,i)} \), we obtain:

\[ S = \sum_{(i,j) \in E} (x_i + x_j) y_{(i,j)} \]

Hence:

\[ S = \sum_{i \in V} (\sum_{k: (i,k) \in E} y_{(i,k)}) x_i = \sum_{(i,j) \in E} (x_i + x_j) y_{(i,j)} \leq 2 \sum_{(i,j) \in E} y_{(i,j)} \]

because \( x_i + x_j \leq 2 \), so we obtain:

29
2.3. Used methods and techniques

\[ \sum_{i \in V} w_i x_i \leq 2 \sum_{(i,j) \in E} y_{(i,j)} \quad (2) \]

Thus, the inequality (2) cited above shows that the algorithm is a 2-approximation algorithm. In this thesis, we will develop in chapter 7 the basic idea cited above into a primal-dual algorithm for a generic problem, by using a hitting set concept.

2.3.2 The Rounding Approach

It is easy to formulate many combinatorial optimization problems as integer linear programs (ILPs). The usual technique consists to solve the linear relaxation of the ILP and then rounding the solution to an integer one. Below we will present an application of the rounding approach on a problem related to vertex cover problem.

An IP formulation (IP 2.5) for capacitated vertex cover problem 2.2.6 is as follows [72]:

\[
\begin{align*}
\text{Minimize} & \quad \sum_{v} w_v x_v \\
\text{Subject to:} & \\
& y_{eu} + y_{ev} \geq 1 \quad e = \{u, v\} \in E \\
& k_v x_v - \sum_{e \in \delta(v)} y_{ev} \geq 0 \quad v \in V \\
& x_v \geq y_{ev} \quad v \in e \in E \\
& y_{ev} \in \{0, 1\} \quad v \in e \in E \\
& x_v \in N \quad v \in V
\end{align*}
\]

where: \( \delta(v) \) is a subset of edges incident to \( v \), \( d(v) = |\delta(v)| \) is a degree of \( v \), \( x\{i, j\} \) means that the edge is oriented from \( i \) to \( j \) and \( y_{ev} = 1 \) denotes that the edge \( e \in E \) is covered by vertex \( v \).

The reference [68] presents the following algorithm 3:

We can easily obtain this theorem 2.10:

**Theorem 2.10** [36] If \( \alpha = \frac{2}{3} \) then the algorithm 3 is a 3-approximation.

In the bounded version introduced by Chuzhy and Noar [13] where there is a bound \( b_v \) on \( x_v \), i.e. a vertex \( v \) can used at most \( b_v \) times to cover edges, we can obtain a 2-approximation in which \( x_v^* \leq 2 x_v \).

The weights on the vertices constitute a generalization of the problem. Denote by \( c_{eu} \) cost of assignment an edge \( e \) to vertex \( u \). The change in the IP (IP 2.5) is to add \( \sum_{e \in E} \sum_{u \in e} c_{eu} y_{eu} \) to the objective function.
Chapter 2. Related work

Algorithm 3 Threshold and Round [68]

Solve the above LP (relaxation of IP 2.5) to obtain a optimal fractional solution.
Pick a value $\alpha$ uniformly at random in the interval $[\frac{1}{2}, 1]$.

for Each edge $e = (u, v)$ do
  if $y_{eu} \geq \alpha$ then
    set $y^*_{eu} = 1$.
  else
    if $y_{ev} \geq \alpha$ then
      set $y^*_{ev} = 1$.
    else
      if $y^*_{eu} < \alpha$ and $y^*_{ev} < \alpha$
        We will use dependent rounding, let $E' = \{e(u, v) \in E : y^*_{eu} < \alpha \ and \ y^*_{ev} < \alpha\}$
        denote this subset of edges. Create a bipartite graph as follows. Let one side contain a
        vertex corresponding to each edge in $E'$. The other side contains a vertex corresponding
to each vertex in $V$. There is an edge $e \in E'$ if $e$ is incident to $u$.
        The weight of this edge is $y_{eu}$. W.l.o.g, $y_{eu} + y_{ev} = 1$. We now use dependent rounding
        to round the $y_{eu}$ values to integers $y^*_{eu}$. We define $x^*_u = \lceil \sum_{e \in E(u)} y^*_{eu} \rceil$. In other words,
after rounding the $y_{eu}$ values to $\{0, 1\}$. We simply define the $x^*_u$ value to be the number
of copies of $u$ that are required to cover all the edges assigned to it.

  end if
  end if
end for

Suppose that $OPT_{LP} = OPT^v_{LP} + OPT^e_{LP}$, where:

$OPT^v_{LP}$ : denotes the optimum fractional cost of chosen vertices.

and $OPT^e_{LP}$ : denotes the optimum assignment cost of edges.

In [36] we found this theorem 2.11:

**Theorem 2.11** [36] Algorithm Threshold and Round finds a solution $x^*$, $y^*$ such
that the expected weight of vertices is at most $2OPT^v_{LP}$ and the expected assignment
cost is at most $(4 - 2\sqrt{2})OPT^e_{LP}$. Thus this gives a 2-approximation for the problem
with vertex weights and assignment costs (since the total cost is at most $2OPT_{LP}$).

2.3.3 The Greedy Method

To solve an optimization problem we can use greedy method. This approach consists
into a construction of a solution thought different stages. At each stage we make a
decision that is locally optimal according to some greedy criterion. Moreover, once
a decision is made, it is never revoked. The greedy method does not always lead to
an optimal solution but there are a few optimization problems that can be solved
exactly by the greedy method. Algorithm 4 below describes how to make a solution
by a greedy method.
2.3. Used methods and techniques

Algorithm 4 Greedy Algorithm

Require: I Set of elements.
Ensure: S Initialized with ∅.
while S is not complete do
  Select the best element $x$ of $I$.
  Put $x$ in $S$.
  Remove $x$ from $I$.
end while

And now we will detail one application of this approach. A greedy algorithm consists in selecting of one set at a time that contains most elements among the uncovered ones. In [37, 47] it was proved that the greedy algorithm is a $H(d)$-approximation algorithm for the unweighed set cover problem, with $H(d) = \sum_{i=1}^{d} \frac{1}{i}$ and $d$ is the size of the largest set.

Chvátal [34] extends this algorithm to the weighed set cover problem and proves that this algorithm is still a $H(d)$-approximation algorithm.

Algorithm 5 The Greedy Algorithm [CHVÁTAL] [34]

Step 1:
Set $C^0 = \emptyset$; $S_j^1 = S_j$, $j \in J$; $I = \{1, ..., m\}$; $k = 0$.
Step 2:
Set $k \leftarrow k + 1$. Select a set $S_{j\lambda}$, such that $\frac{w_{j\lambda}}{|S_{j\lambda}|} = \min_{j \in J} \frac{w_j}{|S_j|}$. Set $C^k = C^k \cup \{j\lambda\}$ and $S_j^{k+1} = S_j^k \setminus S_{j\lambda}$, $j \in J$, $I \leftarrow I \setminus S_{j\lambda}$.
Step 3:
if $I = \emptyset$ then
  Stop and output cover $C^k$.
else
  Go to Step 2.
end if

The basic idea of algorithm 5 is to select a set which covers a maximum number of elements not already covered by applying a criterion on weights in each iteration. The weight condition is $\frac{w_j}{|S_j|} = \min_{j \in J} \frac{w_j}{|S_j|}$. The greedy algorithm is thus an $O(\log(n))$-approximation algorithm. A natural extension of the greedy method to the $k$-separator problem is proposed in chapter 7.

2.3.4 Polyhedral Approach

The polyhedral approach had been introduced by Edmonds in 1965 [24]. Combined with branch-and-bound [55] or branch-and-cut, it is on one of the most powerful methods to solve NP-hard combinatorial optimization problems. The objective of this method is to reduce an integer program to a linear program by generating a
Chapter 2. Related work

description of the convex hull of feasible solutions, $\text{Conv}(X)$, where $X$ is the set of solutions. For NP-hard problems, it is difficult to obtain a complete description for $\text{Conv}(X)$. If the inequalities define facets of $\text{Conv}(X)$, these inequalities are needed for the description of $\text{Conv}(X)$.

In practice, we need to generate efficient methods (exact or heuristic) to separate these inequalities.

It is important to introduce the notion of cutting plane and separation method.

Given the following integer program:

$$\max \{ cx : x \in X \subset \mathbb{R}^n \}$$

Let denote it by $IP_0$.

The separation problem associated with $IP_0$ is the problem defined by: given $x' \in \mathbb{R}^n$, is $x' \in \text{Conv}(X)$? If not, find an inequality $\pi x \leq \pi_0$ satisfied by all points in $X$, but violated by the point $x'$ \[88\].

Let $F$ be a family of valid inequalities $\pi x \leq \pi_0$, $(\pi, \pi_0) \in F$ for $X$.

we can use the algorithm below \[88\] for the cutting-plan and separation for $IP_0$, that generates "useful" inequalities from $F$.

**Algorithm 6 Cutting Plane Algorithm \[88\]**

Initialization: Set $t = 0$ and $P^0 = P$.

Iteration $t$: Solve the linear program $\max \{ cx : x \in P^t \}$

Let $x^t$ be an optimal solution.

if $x^t \in \mathbb{Z}^n$ then

Stop and $x^t$ is an optimal solution for IP.

else

$x^t \notin \mathbb{Z}^n$ solve the separation problem for $x^t$ and the family $F$.

end if

if an inequality $(\pi^t, \pi_0^t) \in F$ is found with $\pi^t x^t > \pi_0^t$ then

Set $P^{t+1} = P^t \cap \{ x : \pi^t x \leq \pi_0^t \}$, and augment $t$.

else

Stop.

end if

If the algorithm finishes without finding a solution for IP, the linear relaxation is improved by adding a violated valid inequality. In practice, it is better to add many violated cuts in each step, and not necessary just one at time. In this paragraph we will analyze some inequalities related to the stable (independent) set polytope.

Remember that the stable set problem is related to the $k$-separator problem. The stable set polytope $P_G$ is the convex hull of the characteristic vectors of stable sets of the graph $G$.  

33
2.3. Used methods and techniques

\[ P_G = \text{Conv}\{x \in \{0, 1\}^V : \{v \in V : x_v = 1\} \text{ is a stable of } G\} \]

In \[14, 56\] we find some well-known valid inequalities for \( P_G \):

- \( x_v \geq 0 \) for \( v \in V \): Trivial Inequalities.
- \( \sum_{v \in C} x_v \leq k \) where \( C \) is the vertex set of a cycle of length \( 2k + 1 \): Cycle Inequalities.
- \( \sum_{v \in S} x_v \leq 1 \) where \( S \) induces a clique: Clique Inequalities.

We describe below the approach to solve the separation problem for the class consisting of the cycle inequalities.

Given \( x^* \in \mathbb{R}^{|V|} \), we define edge-weights as follows: \( w_e^* = \frac{1}{2}(1 - x_u^* - x_v^*), \forall e = (u,v) \in E \). Suppose \( C = (v_1, v_2, \ldots, v_{2k+1}) \) is an odd cycle in \( G \). Then \( w^*(C) = k + \frac{1}{2} - \sum_{i=1}^{2k+1} x_i^* \) (remember that \( w_e^* = \frac{1}{2}(1 - x_u^* - x_v^*), \forall e = (u,v) \in E \)). Hence \( x^* \) violates the cycle inequality corresponding to \( C \) if and only if \( W^*(C) < \frac{1}{2} \).

Therefore a most-violated cycle inequality corresponds to an odd cycle in \( G \) having minimum weight (with respect to \( w^* \)).

A minimum-weight odd cycle can be computed using the algorithm introduced by Grötschel and Pulleyblank \[50\] sketched below.

Let \( G'(V'_1 \cup V'_2, E') \) be a bipartite graph constructed from \( G \), where \( V'_1 \) and \( V'_2 \) are copies of \( V \) with \( (u_1, v_2) \) and \( (u_2, v_1) \) in \( E' \) if and only if \( (u, v) \in E \); furthermore, \( C'(u_1, v_2) = C'(u_2, v_1) = C(u, v) \). Hence a minimum-weight path (with respect to \( C' \)) from \( v_1 \) to \( v_2 \) in \( G' \) corresponds to a minimum-weight odd closed walk (with respect to \( C \), a walk is a finite non-empty sequence \( (v_0, e_1, v_1, e_2, v_2, \ldots, e_l, v_l) \)) containing \( v \) in \( G \). So we can find a minimum-weight odd closed cycle. Moreover, such an odd cycle can be found in \( O(|V|^3) \) time.

Another family of valid inequalities for \( P_G \) called antiweb inequalities are introduced by Trotter in \[79\]. Before presenting this class of inequalities, we give definition for web and antiweb graphs. Let \( m \) and \( p \) be integers satisfying \( p \geq 2 \) and \( m \geq 2p + 1 \). As defined in \[79\], the web \( W_m^p = (V(W), E(W)) \) is a graph, where \( V(W) = \{v_1, \ldots, v_m\} \) is a vertex set and the edge set is \( E(W) = \{(v_i, v_j)|v_i, v_j \in V(W) \text{ and } p \leq |i - j| \leq m - p\} \). (for a sample see figure 2.7). The antiweb \( AW_m^p = (V(W), E(W)) \) is the complement of the web \( W_m^p \) (an example is shown in
Chapter 2. Related work

Figure 2.7: Left: web $W_{10}^3$ and right: antiweb $AW_{10}^3$

Figure 2.8: 1-wheel graph [12]

The inequality $\sum_{i=1}^{m} x_i \leq \left\lfloor \frac{m}{p} \right\rfloor$ is the antiweb inequality described in [79].

Cheng and Cunningham [12] generalized a set of valid inequalities for $P_G$ called "wheel inequalities". They derived these inequalities in the case of simple 1-wheel configurations (subdivisions of wheels in which each face-cycle is odd, see figure 2.8) as their support graphs. Cheng and Vries [23] enlarged this class of separable inequalities to a new large class antiweb-wheel inequalities valid for $P_G$. Before providing some valid inequalities in the case cited above, we will introduce some definitions [12].

35
2.4. Further connections between the $k$-separator problem and other problems

Let $k$ be a positive integer, $G = (V, E)$ an undirected graph with $V = \{v_0, v_1, \ldots, v_{2k+1}\}$ and $E = \{(v_0, v_i), (v_i, v_{i+1}) : 1 \leq i \leq 2k + 1\}$. We take $v_{2k+1} = v_1$. We denote by $P_{0,i}$ a path obtained from $v_0$ to $v_i$ and $P_{i,i+1}$ a path obtained from $v_i$ to $v_{i+1}$.

A graph is a 1-wheel of size $2k + 2$ if each cycle $C_i$ constructed by concatenation of $P_{0,i}, P_{i,i+1}, P_{i,i+1}$ is odd for any $i$. Consider $W = W(v_0, v_1, v_2, \ldots, v_{2k+1})$ is a 1-wheel. $v_0$ is the hub ($\{a\}$ in figure 2.8). $P_{0,1}, P_{0,2}, \ldots, P_{0,2k+1}$ are the spokes. $(v_1, v_2, \ldots, v_{2k+1})$ are the spoke-ends ($\{b, c, d, g, i\}$ in figure 2.8). $P_{1,2}, P_{2,3}, \ldots, P_{2k+1,1}$ is the rim. $E = \{v_i \in V : \text{where } P_{0,i} \text{ is an even path}\}$ ($\{b, c, i\}$ in figure 2.8). $O = \{v_i \in V : \text{where } P_{0,i} \text{ is an odd path}\}$ ($\{d, g\}$ in figure 2.8). $S = S(W)$ is the set of internal vertices of the spokes ($\{j, k, l\}$ in figure 2.8). $R = R(W)$ is the set of internal vertices of the rim-paths ($\{m, e, f, h\}$ in figure 2.8).

Now we mention some valid inequalities for the polytope $P_G$ \[12\]:

\begin{itemize}
  \item $(2k + 1)x_0 + 2 \sum_{i=1}^{2k+1} x_i + 2 \sum_{v \in S} x_v + \sum_{v \in R} x_v \leq |S| + \frac{1}{2}|R| + 2k + 1 \quad (3).
  \item (2k + 1)x_0 + 2 \sum_{i=1}^{2k+1} x_i + 2 \sum_{v \in E} x_v + 2 \sum_{v \in S \cup R} x_v \leq |S| + |R| + |E| + 2k + 1 \quad (4).
\end{itemize}

In the case of p-wheel inequalities, we have \[12\]:

\begin{itemize}
  \item $2(2k + 1) \sum_{i=1}^{p+1} x_{0i} + 2(p + 1) \sum_{v \in E} x_v + 2 \sum_{v \in S} x_v + \sum_{v \in R} x_v \leq 2k + 1 + |S| + \frac{1}{2}|R| + p|E| \quad (5).
  \item $2(2k + 1) \sum_{i=1}^{p+1} x_{0i} + 2(p + 1) \sum_{v \in E} x_v + 4 \sum_{v \in O} x_v + 2 \sum_{v \in S \cup R} x_v \leq 3(2k + 1) + |S| + |R| + (p - 1)|E| \quad (6).
\end{itemize}

In this thesis we generalize these inequalities for the $k$-separator problem in chapter 5.

2.4 Further connections between the $k$-separator problem and other problems

As mentioned in the introduction, the case $k = 1$ corresponds to the vertex cover problem (or the stable set problem) that received a lot of attention in literature. In this section we present two problems close to $k$-separator problem \[83\]. It starts in 2.4.1 with the first part by describing the disconnecting graphs problem \[53\]. The problem consists in disconnecting a graph by removing a set of vertices of minimum
size, such that each connected component has a size less than a given positive number. Finally, the 2.4.2 second and last part is devoted to vertex-separator problem [21]. It is a subset of vertices C, where the graph without it is divided into two parts A and B, where there is no edge between A and B, and |C| is minimized subject to a bound on max \{ |A|, |B| \} [21].

2.4.1 Disconnecting Graphs problem

The only paper where the $k$-separator problem [84] is considered in its general form is [53] where the goal is to remove vertices to disconnect graphs. The following \{0,1\}-programming formulation is proposed in [53]. Let $G(V,E)$ be an undirected graph, with $n = |V|$ and $m = \binom{n}{2}$

$$\begin{align*}
\text{max} \sum_{i \in V} y_i \\
st. : \\
x_{ij} + x_{ik} - x_{jk} &\leq 1, \forall i, j, k \in V, i \neq j \neq k \quad (2.6.1) \\
\sum_{j \in V \setminus \{i\}} x_{ij} &\geq c - 1, \forall i \in V \quad (2.6.2)
\end{align*}$$
and an integer $c \geq 1$

$$\begin{align*}
y_i + y_j - x_{ij} &\leq 1, \forall (i, j) \in E \quad (2.6.3) \\
x_{ij} &\in \{0, 1\}, \forall i, j \in V, i \neq j \quad (2.6.4) \\
y_i &\in \{0, 1\}, \forall i \in V \quad (2.6.5)
\end{align*}$$

(denotes capacity of each component). The formulation proposed in [53] uses variables $y_i \in \{0, 1\}$ and $x_{ij} \in \{0, 1\}$ for all $i, j \in V$, $i \neq j$ where $y_i = 0$ if and only if $i \in V$ is deleted from $G$ and $x_{ij} = 0$ if and only if $i, j \in V$ are not in the same component.

The conditions (2.6.1) are called triangle inequalities and they mean that, for any $i, j, k \in V$, such that $i \neq j \neq k$, if vertices $i$ and $j$ and also vertices $j$ and $k$ are assigned to the same component, then vertices $i$ and $k$ must be assigned to the same component. The constraints (2.6.2) are named the capacity constraints and they ensure that each component must have at most $c$ vertices. Moreover, the inequalities (2.6.3) are called the connectivity constraints, and they imply that for the two vertices of every edge $(i, j) \in E$, they must be in the same component or they will be deleted from the graph $G$. At last, constraints (2.6.4), (2.6.5) are the integrality constraints.
2.4. Further connections between the $k$-separator problem and other problems

In [53] we have the following lemma 2.4.

Lemma 2.4 [53] Let $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$ be a feasible solution to the Linear Programming relaxation of (IP 2.1) with $y_i \in \{0, 1\}$ for all $i \in V$. Then there exists a feasible solution $(\pi, \overline{y}) \in \{0, 1\}^{m+n}$ to (IP 2.1) with $\overline{y}_i = y_i$ for all $i \in V$.

Therefore, by lemma 2.4 the inequalities $x_{ij} \in \{0, 1\}$, such that $i \neq j$ can be relaxed to $0 \leq x_{ij} \leq 1$, by this relaxation the number of variables in (IP 2.1) can be reduced to $n$ [53]. In [53] we have a polyhedral study of the polytope related to the formulation above, we will show some of this results below. Many applications of this problem are mentioned in [53]. This includes a process planning (or scheduling) application. It consists to consider a set of machines $M_1, M_2, \ldots, M_n$, and a set of process $P_1, P_2, \ldots, P_l$. Additionally, $a_{ij} = 1$ (where $a_{ij} \in n \times l$ \{0,1\}-matrix $A$) if and only if $P_j (j = 1, \ldots, l)$ has to be runned on machine $M_i (i = 1, \ldots, n)$ and an positive integer $d \geq 1$. The problem is to assign a maximum process to the so-called production cells, where each one contains no more than $d$ processes, each process and each machine take place in at most one cell and processes assigned to different cells are not connected (i.e., there is not any machine $h$ such that $a_{ih}a_{hj} \neq 0$ if $P_i$ and $P_j$ are in different cells) [62]. See also [89, 31] for more details. Other applications are mentioned in [53]. Let $P(G, c)$ denote a polytope of the disconnecting graphs problem (DG). We can observe that $P(G, 1)$ is isomorphic to the independent set polytope. The polytope $P(G, c)$ with $c \geq 2$ is full dimensional [53]. Theorem 2.12 gives some conditions to define a facet for $P(G, c)$.

Theorem 2.12 [53] Let $G = (V, E)$ be a graph, $c \geq 2$ an integer, and let $(a, b)^T(x, y) \leq a_0$ be a facet-defining inequality for $P(G, c)$.

1. If, for any $d \geq c$, the inequality $(a, b)^T(x, y) \leq a_0$ is valid for $P(G, d)$, then $(a, b)^T(x, y) \leq a_0$ is facet-defining for $P(G, d)$.

2. If, for any $E' \subseteq E$, the inequality $(a, b)^T(x, y) \leq a_0$ is valid for $P((V, E'), c)$, then $(a, b)^T(x, y) \leq a_0$ is facet-defining for $P((V, E'), c)$.

For subgraph of $G$ we can define also a facet in some conditions, the result is given in theorem 2.13.
Chapter 2. Related work

Theorem 2.13 [53] Let $G = (V, E)$ be a graph, let $c \geq 2$ be an integer, let $G' = (U, E(U))$ be a subgraph of $G$, and let

$$\sum_{i,j \in U, i \neq j} a_{ij}x_{ij} + \sum_{i \in U} b_iy_i \leq a_0 \quad (2.1)$$

be a facet-defining inequality for $P(G', c)$ such that the following conditions hold.

1. Inequality 2.1 is valid for $P(G, d)$ for some integer $d \geq c$.

2. There exists an order $(e_1, \ldots, e_p)$ of the elements in $\hat{E} := \{(i, j) | i \in U, j \in V \setminus U\}$ such that for each $e_k \in \hat{E}$ there exists $(x, y) \in P(G, d)$ satisfying 2.1 at equality with $x_{e_k} = 1$ and $x_{e_l} = 0$ for all $l = k + 1, \ldots, p$.

3. For all $i \in V \setminus U$ there exists $(x, y) \in P(G, d)$ with $y_i = 1$ satisfying 2.1 at equality. Then 2.1 is facet-defining for $P(G, d)$.

Some trivial facet-defining inequalities are shown in theorem 2.14.

Theorem 2.14 [53] Let $G = (V, E)$ be a graph, let $c \geq 2$ be an integer and let $(a, b)^T(x, y) \leq a_0$ be any facet-defining inequality for $P(G, c)$.

1. The inequalities $x_{ij} \geq 0 \ (i, j \in V, i \neq j)$ define (trivial) facets of $P(G, c)$.

2. The inequalities $x_{ij} \leq 1 \ (i, j \in V, i \neq j)$ do not define facets of $P(G, c)$.

3. The inequalities $y_i \geq 0 \ (i \in V)$ define (trivial) facets of $P(G, c)$.

4. The inequalities $y_i \leq 1 \ (i \in V)$ define facets of $P(G, c)$.

5. $a_0 \geq 0$.

6. For nontrivial facets : $b_i \geq 0$ for all $i \in V$.

In [53] we found also a relationship between clique partitioning polytope, capacitated clique partitioning polytope, maximal weighted clique polytope, boolean quadric polytope and a polytope denoted by $P(G, B, C)$ on the one hand and $P(G, c)$ polytope on the other hand. Let us first describe these polytopes. Then in a second stage we show a relationship between these polytopes and $P(G, c)$.

Given a complete graph $K_n = (V_n, E_n)$ where $|V_n| = n$ and with edge weights
2.4. Further connections between the $k$-separator problem and other problems

$w_{ij} \in R$ for all $\{i, j\} \in E_n$, the clique partitioning problem (CLP) \cite{53} is defined by

\[
\text{CLP} \quad \begin{cases}
\max \sum_{\{i,j\} \in E_n} w_{ij}x_{ij} \\
s.t. : \\
x_{ij} + x_{ik} - x_{jk} \leq 1, \forall i, j, k \in V, i \neq j \neq k \\
x_{ij} \in \{0, 1\}, \forall i, j \in E_n
\end{cases}
\tag{2.2}
\]

Let $P_{n}^{\text{CLP}}$ denote a polytope of (CLP). This polytope is studied in \cite{51, 54}.

If we add a bound on the number of nodes in a clique (called it $c$), we obtain the capacitated clique partitioning problem (CCLP) \cite{53}: add to (CLP) the inequalities

\[
\sum_{j \in V \setminus \{i\}} x_{ij} \leq c - 1 \text{ for all } i \in V
\tag{2.3}
\]

The polytope associated with this problem is declared $P_{n,c}^{\text{CCLP}}$. A depth study can be found in \cite{80, 28}. If we add the variables $y_i \in \{0, 1\}, \forall i \in V$ to (CCLP) and the appropriate connectivity constraints we get the polytope $P(G, c)$.

When we look for a single maximal weighted clique it’s sufficient to add the constraints \cite{53}:

\[
x_{ij} + x_{jk} + x_{kl} - x_{ik} - x_{jl} \leq 1, \forall i, j, k, l \in V \text{ and } i \neq j \neq k \neq l
\tag{2.4}
\]

The polytope of this problem, called $P_{n,c}^{\text{CLI}}$ is studied by Dijkhuizen and Faigle \cite{28, 53}. Park and Lee studied in \cite{41} the same problem but they introduced the following extended formulation:

\[
\text{EXTCLI} \quad \begin{cases}
\max \sum_{\{i,j\} \in E_n} w_{ij}x_{ij} \\
s.t. : \\
y_i + y_j - x_{ij} \leq 1 \forall \{i,j\} \in E_n \\
\sum_{i \in V_n} y_i \leq c \quad (2.5) \\
x_{ij} - y_i \leq 0, x_{ij} - y_j \leq 1, \forall \{i,j\} \in E_n \\
x_{ij} \in \{0, 1\}, \forall \{i,j\} \in E_n \\
y_i \in \{0, 1\}, \forall i \in V_n
\end{cases}
\]

Let $P_{n,c}^{\text{EXTCLI}}$ denote a polytope of (EXTCLI). This polytope is also investigated in \cite{25}. When we remove constraint 2.5 from the formulation above we get the boolean
Chapter 2. Related work

quadric polytope $P_{n}^{bspd}$, which has been studied by Padberg in [64].

The last polytope is based on formulation proposed by Borndörfer et al. [67] in the field of decomposition constraint matrices, and by Kumar et al. [44] in a cell-formation context. This formulation has a supplementary input parameter, an integer $B \in \mathbb{N}$, where $B$ is an upper bound on the number of components to be created. The formulation uses variables $z_{ib} \in \{0, 1\}, \forall i \in V$ and $b \in \{1, \ldots, B\}$, such as

\[
z_{ib} = \begin{cases} 
1 & \text{if vertex } i \in V \text{ is assigned to component } b, \\
0 & \text{otherwise.}
\end{cases}
\]

The corresponding integer program is [53]:

\[
\begin{align*}
\text{MD} \left\{ 
\max & \sum_{i \in V} \sum_{b=1}^{B} z_{ib} \\
\text{s.t. :} & \\
& \sum_{b=1}^{B} z_{ib} \leq 1, \forall i \in V \\
& \sum_{i \in V} z_{ib} \leq c, \forall b \in \{1, \ldots, B\} \\
& z_{ib} + z_{jb'} \leq 1, \forall \{i, j\} \in E, b, b' \in \{1, \ldots, B\}, b \neq b' \\
& z_{ij} \in \{0, 1\}, \forall i \in V, b \in \{1, \ldots, B\}
\end{align*}
\]

Let $P(G, B, c)$ denote a polytope of $MD$. The block-invariant inequalities have been defined in [67]. This inequalities can be written as

\[
\sum_{i \in V} a_{i} \sum_{b=1}^{B} z_{ib} \leq a_{0} \tag{2.7}
\]

The following lemma 2.5 shows the relationship between the polytopes introduced above.

Lemma 2.5 [53]

1. An inequality valid for $P_{n}^{CLP}$ is valid for $P_{n,c}^{CCLP}$ for all $n, c \in \mathbb{N}$.

2. An inequality valid for $P_{n,c}^{CCLP}$ is valid for $P_{n,c}^{CLI}$ for all $n, c \in \mathbb{N}$ as well as valid for $P(G, c)$.

3. An inequality valid for $P_{n,c}^{CLI}$ is valid for $P(G, c)$ if $G$ is a clique, for all $c \in \mathbb{N}$. 

41
2.4. Further connections between the $k$-separator problem and other problems

4. A block-invariant inequality valid for $P(G, B, c)$ is valid for $P(G, c)$ when substituting $y_i = \sum_{b=1}^{B} z_{ib}$, for $i = 1, \ldots, n$.

5. An inequality valid for $P(G, c)$ is valid for $P_{n,c}^{\text{extcli}}$ for all $n, c \in \mathbb{N}$.

A graphical representation of lemmas 2.5 is given in figure 2.9 [53].

Lemma 2.6 [53]

1. An inequality $a^T x \leq a_0$ facet-defining for $P_{n}^{\text{CLP}}$ is also facet-defining for $P_{n, c}^{\text{CCLP}}$ for all $c \geq k_{n}^{\text{CLP}}$.

2. An inequality $a^T x \leq a_0$ facet-defining for $P_{n, c}^{\text{CCLP}}$ is facet-defining for $P(G, c)$ for all graphs $G = (V, E)$ with $|V| = n$.

3. An inequality $a^T x \leq a_0$ facet-defining for $P_{n, c}^{\text{extcli}}$ which is valid for $P(G, c)$ is facet-defining for $P(G, c)$ for all graphs $G = (V, E)$ with $|V| = n$.

4. An inequality $a^T x \leq a_0$ facet-defining for $P_{n}^{\text{BOQD}}$ is also facet-defining for $P(G, c)$ for all graphs $c \geq k_{n}^{\text{BOQD}}$ and for all graphs $G = (V, E)$ with $|V| = n$.

5. A block-invariant inequality

\[ \sum_{i \in V} b_i \sum_{b=1}^{B} z_{ib} \leq b_0 \quad (2.8) \]

facet-defining for $P(G, B, c)$ is facet-defining for $P(G, c)$ when substituting $y_i = \sum_{b=1}^{B} z_{ib}$, for $i = 1, \ldots, n$.

A graphical representation of lemmas 2.6 is given in figure 2.10 [53].

The inheritance of facet-defining inequalities is represented in figure 2.10 [53].

Given a facet-defining inequality $a^T x \leq a_0$ of $P_{n}^{\text{CLP}}$ ($P_{n}^{\text{BOQD}}$) there exists a set $M_a$ of $\dim(P_{n}^{\text{CLP}})$ ($\dim(P_{n}^{\text{BOQD}})$) affinely independent solutions satisfying this inequality at equality. Let us denote by $K_{M_a}^{\text{CLP}}$ ($K_{M_a}^{\text{BOQD}}$) the size of the largest clique (in terms of number of vertices) present in a solution in $M_a$, and let

\[ K_{a}^{\text{CLP}} := \min\{K_{M_a}^{\text{CLP}} : M_a \text{ affinely independent solutions}, \] \[ |M_a| = \dim(P_{n}^{\text{CLP}}), a^T x = a_0 \ \forall x \in M_a \text{ where } a^T x \leq a_0 \text{ facet-defining for } P_{n}^{\text{CLP}} \} \]

and
Chapter 2. Related work

Figure 2.9: Inheritance of valid inequalities [53]

Figure 2.10: Inheritance of facets inequalities [53]
2.4. Further connections between the \( k \)-separator problem and other problems

\[ K^\text{BOQD}_a := \min \{ K^\text{BOQD}_M : M \text{ affinely independent solutions,} \} \]

\[ |M_a| = \dim(P_n^\text{BOQD}), a^T x = a_0 \ \forall x \in M_a \text{ where } a^T x \leq a_0 \text{ facet-defining for } P_n^\text{BOQD} \]

They are many other valid inequalities for \( P(G, c) \) in the reference [53], among these we found the following constraints used in the branch-and-cut algorithm presented in the same reference (see [53] for more details).

- **The cover inequalities**:
  
  Let \( W \subseteq V \) such that \(|W| = c+k, c \geq 2\) and \((W, E(W))\) is \(k\)-vertex connected, we have the following valid inequality:

  \[ \sum_{i \in W} y_i \leq c \] (2.9)

  Borndörfer gives in [67] the following heuristic (algorithm 7) to separate cover inequalities with \( k = 1 \) and \( k = 2 \). Let \( E(\{v\}, W) \) denote the set of edges incident to \( v \) and \( W \).

  **Algorithm 7** Searching \( W \subseteq V \) such that \((W, E(W))\) is two connected [53]

  ```
  for each edge \( \{i, j\} \in E \) do
    Set \( W := \{i, j\} \) and \( N := N(i) \cap N(j) \)
    while \( (N \neq \emptyset) \) and \( (|W| < c + 2) \) do
      Choose \( l \in N \)
      \( W := W \cup \{l\} \)
      \( N := \{v \in V \setminus W : |E(\{v\}, W)| \geq 2\} \)
    end while
    if \(|W| = c + 2\) then
      Check cover inequality
    end if
  end for
  ```

- **Clique inequalities**:

  Let \( c, p \) two integers with \( c \geq p + 1 \) and let \( U \subseteq V \) be such that \((U, E(U))\) is a clique in \( G \). Then the inequality 2.10 defines a facet iff \(|U| \geq p + 2 \) [53].

  \[ p \sum_{i \in U} y_i - \sum_{\{i,j\} \in E(U)} x_{ij} \leq \binom{p+1}{2} \] (2.10)

  Separation of this family of inequalities based on recursive algorithm 8 search a maximal clique in \( G \). The subroutine 'Clique' is an exact algorithm for finding all maximal cliques in graph but has an exponential time in the worst case.
Algorithm 8 Searching the maximal clique [53]

\[ U = \emptyset \text{ and } N = V \]

Call the subroutine \texttt{Clique}(U, N)

\textbf{CLIQUE (U, N)}

\begin{verbatim}
BEGIN
\textbf{k} := \text{min}\{l| (v_l \in N) \land (l > \max\{m| v_m \in U\})\}
\textbf{while} (k \leq n) \textbf{do}
  \textbf{U} := \textbf{U} \cup \{v_k\}
  \textbf{N}' := \{v \in N| \{v, v_k\} \in E\}
  \textbf{if} (\textbf{N}' = \emptyset) \textbf{then}
    \textbf{U} is maximal clique
  \textbf{else}
    \textbf{CLIQUE}(\textbf{U}, \textbf{N}')
  \textbf{end if}
  \textbf{U} := \textbf{U} \setminus \{v_k\}
  \textbf{k} := \text{min}\{l| (v_l \in N) \land (l > k)\}
\textbf{end while}
END
\end{verbatim}

- Star inequalities:
  Let \( c \geq 2 \) an integer, and let \( i \in V \) be such as \(|N(i)| \geq c\). Then the inequality below is valid for \( P(G,c) \)

\[
(N(i)| - c + 1) . y_i + \sum_{j \in N(i)} y_j \leq |N(i)| \tag{2.11}
\]

The number of star inequalities is \( n \), and therefore separation of this class of inequalities is done by enumeration [53].

- \( K_{1,2} \) inequalities:
  Let \( c \geq 3 \) be an integer, and \( i, j, k \in V, \ i \neq j \neq k \) such that \( \{i, j\}, \{i, k\} \in E \) and \( \{i, j\} \notin E \). Then \( K_{1,2} \) inequalities

\[
y_i + y_j + y_k - x_{jk} \leq 2 \tag{2.12}
\]

is valid and facet-defining for \( P(G,c) \). The separation algorithm is done by enumeration and the number of \( K_{1,2} \)-inequalities is \( O(n^3) \).

- Arrow inequalities:
  Let \( c \geq 3 \) an integer and \( U = i, j, k, l \subseteq V \) such that \((U, E(U))\) is a clique in \( G \). Then the inequality

\[
y(i, j, l) + x_{ik} - x(E(j, k, l)) \leq 2 \tag{2.13}
\]

45
2.4. Further connections between the $k$-separator problem and other problems

is valid and facet-defining for $P(G,c)$. The separation is done by an enumeration algorithm in $O(n^4)$.

- **Four-cycle inequalities**: 
  The four-cycle inequality is
  \[ y(C) - x_{v_1,v_3} - x_{v_2,v_4} \leq 2 \]  
  (2.14)

  Where $c \geq 3$ is an integer and $C := v_1, v_2, v_3, v_4$ is a subset of $V$ that induces a cycle in $G$. This inequality is valid and facet-defining for $P(G,c)$. The separation of these inequalities is done by enumeration.

- **Triangle inequalities**: 
  The separation of the following triangle inequalities is done by enumeration
  \[ x_{ij} + x_{ik} - x_{jk} \leq 1, \text{ for all distinct } i,j,k \in V \]  
  (2.15)

2.4.2 Vertex Separator problem

The last problem is the vertex separator problem. Given a connected graph $G$, a vertex separator in $G$ is a subset of vertices whose removal disconnects $G$. Balas in [21] proposes a polyhedral study of a vertex separator problem (VSP). A VSP can be defined as a subset of vertices, whose removal divides the graph into two disjoint subgraphs. In [21] we have the following definition of VSP: given $G(V,E)$ an undirected graph, an integer $b \leq n$ and $C_i$ a cost for vertex $i \in V$, we want to split $V$ into three sets $A, B$ and $C$, where $A$ and $B$ are not empty (called shores), $(i,j) \notin E$, $i \in A$ and $j \in B$ (condition (i)), $\max\{|A|,|B|\} \leq b$ (condition (ii)) and $\sum_{j \in C} C_j$ is minimized subject to the two conditions mentioned before (i.e., condition (i) and condition (ii)).

A mixed integer formulation is proposed in [21].
Chapter 2. Related work

Let $\max \sum_{i \in V} C_i (u_{i1} + u_{i2})$
\[
\begin{align*}
\text{s.t. :} \\
u_{i1} + u_{i2} & \leq 1, \forall i \in V \quad (2.7.1) \\
u_{i1} + u_{j2} & \leq 1, \forall (i, j) \in E \quad (2.7.2) \\
u_{j1} + u_{i2} & \leq 1, \forall (i, j) \in E \quad (2.7.3) \\
1 & \leq u_{ik}(V) \leq b, \quad k = 1, 2 \quad (2.7.4) \\
u_{ik} & \geq 0, \forall i \in V, k = 1, 2 \quad (2.7.5) \\
u_{i1} & \text{ integer, } \forall i \in V \quad (2.7.6)
\end{align*}
\]

Let $u_{i1} = \begin{cases} 1 & \text{if } i \in A \\ 0 & \text{otherwise} \end{cases}$

Let $u_{i2} = \begin{cases} 1 & \text{if } i \in B \\ 0 & \text{otherwise} \end{cases}$

For $S \subset V$ and $k = 1, 2$, we denote by $u_k(S) = \sum_{i \in S} u_{ik}$ and $u(S) = u_1(S) + u_2(S)$.

Inequality (2.7.1) means that a vertex $i$ cannot be in $A$ and $B$. Constraints (2.7.2) and (2.7.3) imply that vertices of every edge $(i, j) \in E$ must be both either in $A$ or in $B$. Then, condition (2.7.4) prevents that neither $A$ nor $B$ is empty and the size of each subset $(A, B)$ must be less than $b$.

Many applications of VSP are mentioned in [20, 61, 70, 60], among them a problem of minimizing the work involved in solving systems of equations [60].

We detail below the class of symmetric facets of $P(G, b)$ [21]. A valid inequality of $P(G, b)$ is called symmetric if for all $i \in V$, the coefficients $u_{j1}$ and $u_{j2}$ are equal.

Let $G(V, E)$ be a simple undirected graph. $S \subseteq V$ such that $V \subseteq (S \cup N_S(G))$ is called a dominating set for $G$ or for $V$. A vertex $i \in V$ is universal if it is neighbor to evry $j \in V \setminus \{i\}$. The proposition 2.1 shows a relationship between vertex separator and connected dominators.

**Proposition 2.1 [21]** In a connected graph, any separator and any connected dominator have at least one vertex in common.

Let $P(G, b) = \text{conv}\{u \in B^{2n} : u \text{ satisfies } (2.7.1) - (2.7.6)\}$ be a vertex separator problem (VSP) polytope. A vertex $i$ is called regular, if there exists a separator
2.4. Further connections between the k-separator problem and other problems

$C \subset V \setminus \{i\}$ such that $C \cup \{i\}$ is also a separator. Proposition 2.2 gives us the conditions to have a full dimensional polytope.

**Proposition 2.2** [21] If every $i \in V$ is regular, then $P(G, b)$ is full dimensional.

Furthermore, the proposition 2.3 is a direct result from proposition 2.1.

**Proposition 2.3** [21] Let $S$ be a minimal connected dominator of $V$. Then

$$u(S) \leq |S| - 1 \quad (2.16)$$

is a valid inequality for $P(G, b)$.

A valid inequality $\alpha u \leq \alpha_0$ is maximal if there exists no valid inequality $\alpha' u \leq \alpha_0$ with $\alpha' \geq \alpha$ and $\alpha'_j > \alpha_j$ for some $j \in V$ [21]. In objective to prove in which case the inequality 2.16 is maximal for $P(G, b)$ let us introduce some definitions [21]. Let $S \subset V$ be a dominator of $V$. For $i \in S$,

$$P(i) := \{k \in V \setminus S : N_S(k) = i\}$$

is the set of pendent vertices of $i$ [21].

Let $S_D := \{i \in S : P(i) \neq \emptyset\}$, $S_Q := S \setminus S_D$, if $S_Q \neq \emptyset$ then every $i \in S_Q$ is an articulation point of $G(S)$ [21] (label 2.4.2). A forbidden vertex $v \in V \setminus S$ relative to a minimal connected dominator $S$ of $G$ is a node where $G(\{S \cup \{v\}\})$ has a non articulation point and $v$ is adjacent to every $j \in \bigcup_{i \in S} P(i)$ [21]. Now we present the proposition 2.4.

**Proposition 2.4** [21] The inequality 2.16, with $|S| \leq b$, is maximal if and only if $G$ has no forbidden vertices relative to $S$.

Let $F := \{u \in P(G, b) : u(S) = |S| - 1\}$ be a face of $P(G, b)$. $F$ is a facet of $P(G, b)$ if and only if for each equation $\alpha u = |S| - 1$, where $u \in F$ and have a coefficients $\alpha_{j_1}$ and $\alpha_{j_2}$ such that

$$\alpha_{j_1} = \alpha_{j_2} = \begin{cases} 1 & \text{if } j \in S \\ 0 & \text{otherwise (i.e., } j \in V \setminus S) \end{cases}$$

We need propositions 2.5 - 2.7 to prove proposition 2.9. But before giving this proposition we will present proposition 2.8 in order to show that if $G$ has a node
for which none of three conditions cited in propositions 2.5 - 2.7 is satisfied, then
inequality 2.16 is not a facet of $P(G, b)$

**Proposition 2.5** [21] If for some $v \in V \setminus S$, $G(S \cup \{v\})$ and $G(S)$ have a common
articulation point, then $\alpha_{v_1} = \alpha_{v_2} = 0$.

**Proposition 2.6** [21] If for some $v \in V \setminus S$ there exists $l \in P(i)$ for some $i \in S$
such that $(v, l) \notin E$, then $\alpha_{v_1} = \alpha_{v_2} = 0$

**Proposition 2.7** [21] If $v \in P(i)$ for some $i \in S$ such that $|P(i)| \geq 2$, then
$\alpha_{v_1} = \alpha_{v_2} = 0$.

Under consideration of some conditions the inequality 2.16 is not facet-defining
for $P(G, b)$, the proposition 2.8 shows us this result.

**Proposition 2.8** [21] Suppose there exists $v \in V \setminus S$ with the properties

- $G(S \cup \{v\})$ and $G(S)$ have no common articulation point
- $v$ is adjacent to every $j \in \bigcup_{k \in S} P(k)$
- $\{v\} = P(i)$ for some $i \in S$

Then the inequality 2.16 does not define a facet of $P(G, b)$.

Proposition 2.9 shows in which case (2.16) is a facet of $P(G, b)$. But before
presenting it, we give some definitions and notations taken from [21] where those
definitions are required for the presentation of proposition 2.9.

Let $S$ be a minimal connected dominator, with $S = S_D \cup S_Q$, where $S_D$ is defined
above in label 2.4.2 that is the unique minimal dominator contained in $S$, and
$S_Q$ which is defined above in label 2.4.2. A subset $S \subseteq V$ is called *orderly*, if
either $S_Q = \emptyset$, or else $S_D$ contains no articulation point of $G(S)$, and $S_Q$ can be
ordered into sequence $i_1, \ldots, i_q$, with the property that for $r = 1, \ldots, q$, $G(S \setminus \{i_r\})$
has exactly two components with vertex sets $S', S''$, such that $\{i_1, \ldots, i_{r-1}\} \subseteq S'$,
$\{i_{r+1}, \ldots, i_q\} \subseteq S''$ [21]. Let $s = |S|$, $d = |S_D|$ and $q = |S_Q|$ . Let $C_i$ is a separator
in $F$ with shores $A_i, B_i$, let $a_i = |A_i \cap S_D|$ and $b_i = |B_i \cap S_D|$. A separator $C_i$ is called
of type 1 if $S_i \{i\}$ is contained in a single shore, and of type 2 if $(S \setminus \{i\}) \subseteq A_i \cup B_i$,
with $A_i \cap S \neq \emptyset \neq B_i \cap S$ [21]. We have $a_i + b_i = d$ for the separators of type 2,
with $i \in S_Q$ [21]. A collection $C$ of type 2 separators is called representative if it contains exactly one member $C_i$ for each $i \in S_Q$ [21]. If the members of such a collection is ordered according to the rule $a_i \geq a_{i+1}(b_i \leq b_{i+1})$, $i = 1, \ldots, q - 1$, and

$$
a_1^{2k+1} = a_1 + a_3 + \ldots + a_{2k+1}
$$

$$
a_2^{2k} = a_2 + a_4 + \ldots + a_{2k}
$$

(2.17)

with $b_1^{2k+1}$ and $b_2^{2k}$ defined in the same way, then an orderly minimal connected dominator $S$ is called exceptional [21] if

1. $s$ is odd and

2. $S_Q \neq \emptyset$, and for any representative collection of type 2 separators

$$
a_1^{q-1} - a_2^q = \frac{(d-1)}{2}, \text{ if } q \text{ is even}
$$

$$
a_1^q - a_2^{q-1} = \frac{d}{2}, \text{ if } q \text{ is odd}
$$

**Proposition 2.9** [21] Let $S$ be a minimal connected dominator that is orderly, $|S| \leq b$, and assume that every $v \in V \setminus S$ satisfies at least one of the conditions stated in Propositions 2.5, 2.6 and 2.7. Then the inequality 2.16 defines a facet of $P(G, b)$ if and only if $S$ is not exceptional.

If $S$ is exceptional then we have the following proposition 2.10.

**Proposition 2.10** [21] Let $S$, with $S_Q = \{i\}$, be exceptional. Then the inequality

$$pu_1(S \setminus \{i\}) + (p - 1)u_2(S \setminus \{i\}) + (2p - 1)u_{i_2} \leq p(2p - 1)
$$

(2.18)

is valid for $P(G, b)$. Furthermore, 2.18 is a facet defining if and only if every $v \in V \setminus S$ satisfies at least one of the conditions of propositions 2.5, 2.6 and 2.7.

In the reference [21] we find more results of VSP polytope among this a class of asymmetric facets of $P(G, b)$ and some generalized inequalities for $P(G, b)$.

### 2.5 Conclusion

In this chapter we have exposed some related works to $k$-separator problem. Among these we mentioned many combinatorial optimization problems that have a relationship with $k$-separator problem. From primal-dual method to polyhedral approach,
Chapter 2. Related work

we have tried to give an introduction to each of them and applied them to problems that have a relation with $k$-separator problem. In the next chapter we will show many cases where $k$-separator problem can be solved in polynomial time.
Chapter 3

Polynomial cases

Contents

3.1 Introduction ........................................... 52
3.2 Graphs with bounded treewidth ......................... 53
  3.2.1 Basics ............................................. 53
  3.2.2 A dynamic-programming algorithm .................. 54
3.3 Paths, trees and cycles .................................. 56
3.4 \( mK_2 \)-free graphs .................................... 59
3.5 \( (G_1,G_2,G_3,P_6) \)-free graphs .......................... 60
3.6 Interval-filament, asteroidal triple-free and weakly chordal graphs ............................................ 64
3.7 Interval and circular-arc graphs ........................... 66
3.8 Conclusion ............................................... 70

3.1 Introduction

This chapter is devoted to polynomial cases of the \( k \)-separator problem. It starts with bounded treewidth graph, then paths trees and cycles. Many other graph classes where \( k \)-separator problem is polynomial-time solvable are studied, inter alia: \( mK_2 \)-free graphs, \( (G_1,G_2,G_3,P_6) \)-free graphs, interval-filament, asteroidal triple-free, weakly chordal, interval and circular-arc graphs. Concluding is made in the last section.
Chapter 3. Polynomial cases

3.2 Graphs with bounded treewidth

3.2.1 Basics

A tree-decomposition of $G$ is defined by a pair $(\mathcal{X}, T)$ where $\mathcal{X} = (X_t)_{t \in V(T)}$ is a set of vertex subsets of $G$ indexed by vertices of a tree $T$ satisfying the following:

(i) for each $v \in V(G)$, there is some $t \in V(T)$ such that $v \in X_t$;

(ii) for each edge $(u, v) \in E(G)$, there is some $t \in V(T)$ such that $u \in X_t$ and $v \in X_t$;

(iii) for each vertex $v \in V(G)$, if $v \in X_{t_1}$ and $v \in X_{t_2}$ then $v$ belongs to $X_t$ for each $t \in V(T)$ on the path between $t_1$ and $t_2$.

Property (iii) implies that the subgraph of $T$ induced by the vertices $t$ such that $X_t$ contains $v$ is a subtree. The width of the decomposition is given by $\max_{t \in V(T)} |X_t| - 1$. The treewidth of $G$ is the minimum width over all tree-decompositions of $G$.

We assume here that $G$ has a treewidth bounded by a constant $l$. It is well-known that computing the treewidth of a graph and a corresponding minimum-width tree-decomposition can be done in linear time (assuming that $l$ is constant) [6]. Many NP-hard optimization problems can be solved in polynomial time for bounded-treewidth graphs. The algorithms are generally based on dynamic programing and a tree-decomposition of the graph (see e.g., [71, 7, 78]).

A relatively general approach is proposed in [78] to solve vertex partitioning problems in bounded-treewidth graphs. Since the $k$-separator problem can be seen as a vertex partitioning where the partition is given by the $k$-separator and the remaining connected components, the approach of [78] might be used. However, the algorithm of [78] has a polynomial complexity only if the number of subsets of the partition is bounded by the logarithm of the size of the graph (Theorem 5.7 of [78]). Unfortunately, this does not hold for the $k$-separator problem. The approach of [71] also leads to a non-polynomial algorithm. Another extension is proposed in [7] but it is not clear for us how to express the $k$-separator problem in a compatible way with [7].

However, it is not difficult to derive a dynamic programming algorithm for our problem. It is described below for sake of completeness.
3.2. Graphs with bounded treewidth

3.2.2 A dynamic-programming algorithm

Similarly to most dynamic-programming algorithms for bounded-treewidth graphs, we are going to use the separator property induced by the tree-decomposition: given an edge \((t_1, t_2) \in E(T)\), let \(T_1\) and \(T_2\) be the connected components of \(T\) obtained by deletion of the edge, let \(Y_1 = \bigcup_{t \in V(T_1)} X_t\) and \(Y_2 = \bigcup_{t \in V(T_2)} X_t\), then \(X_{t_1} \cap X_{t_2}\) separates \(Y_1\) from \(Y_2\). In other words, \(Y_1 \cap Y_2 = X_{t_1} \cap X_{t_2}\) and each vertex of \(Y_1 \setminus Y_2\) is not adjacent to any vertex of \(Y_2 \setminus Y_1\). Let us consider \(T\) as a rooted tree (by choosing an arbitrary root) and let \(T_t\) be the subtree rooted at \(t \in V(T)\). Then, we can define \(Y_t\) as the union of all subsets indexed by the vertices of \(T_t\): \(Y_t = \bigcup_{t' \in V(T_t)} X_{t'}\). The main idea of this type of algorithms is to consider, for each \(t \in V(T)\), a table of partial solutions of the optimization problem that can be built by considering the tables of the children vertices of \(t\) (in the rooted tree). The validity of this table construction is based on the separator property induced by the tree-decomposition. The optimal solution of the problem is obtained at the root of the tree. This approach is useful in order to derive a polynomial-time algorithm to solve the problem when the size of the table of each \(t \in V(T)\) is polynomially bounded (for the case when the width of the tree decomposition is bounded by a constant).

Before describing how these tables are built, let us introduce one more concept. A nice tree-decomposition is a decomposition where each vertex \(t \in V(T)\) falls into one of the following categories:

- **Leaf**: \(t\) is a leaf of \(T\) and \(|X_t| = 1\).
- **Join**: \(t\) has exactly two children, say \(t'\) and \(t''\), and \(X_t = X_{t'} = X_{t''}\).
- **Introduce**: \(t\) has exactly one child, say \(t'\), and there is a vertex \(v \in V(G)\) such that \(X_t = X_{t'} \cup \{v\}\).
- **Forget**: \(t\) has exactly one child, say \(t'\), and there is a vertex \(v \in V(G)\) such that \(X_t = X_{t'} \setminus \{v\}\).

It is easy to show that given a tree-decomposition, one can transform it into a nice tree decomposition having the same width and a linear number of vertices. This can be done in polynomial time [43]. We will then assume that a nice tree-decomposition of minimum-width is known.
Chapter 3. Polynomial cases

For each \( t \in V(T) \), for each partition \((S_0, S_1, ..., S_j)\) of \( X_t \) where \( j \leq l \) and for each set of numbers \( l_1, ..., l_j \) with \( |S_i| \leq l_i \leq k \), let \( Z^*(t, S_0, S_1, ..., S_j, l_1, ..., l_j) \) be the weight of a minimum-weight \( k \)-separator of \( G(Y_t) \) where \( S_0 \) belongs to the \( k \)-separator while all vertices of \( S_i \) (\( 1 \leq i \leq j \)) belong to the same connected component of size \( l_i \). Observe that if \( S_i \) and \( S_{i'} \) (\( 1 \leq i < i' \)) are adjacent then \( Z^*(t, S_0, S_1, ..., S_j, l_1, ..., l_j) = \infty \) since \( S_i \) and \( S_{i'} \) should be merged into one connected component. We assume that the table of \( t \in V(T) \) contains an entry for each partition \((S_0, S_1, ..., S_j)\) with the associated numbers \( l_1, ..., l_j \) and the optimal cost \( Z^*(t, S_0, S_1, ..., S_j, l_1, ..., l_j) \). A \( k \)-separator of \( G(Y_t) \) achieving this cost (containing \( S_0 \)) can also be kept in the table of \( t \).

The number of entries of the table for each \( t \in V(T) \) is bounded by \( k^{l+1}(l + 1)^{l+1} \) which is polynomial. The optimum solution is obtained at the root vertex by considering the solution of minimum weight among all partial solutions at the root’s table.

To complete the algorithm description, we only have to show how to build the table of a vertex \( t \) knowing the tables of the children of \( t \). The case where \( t \) is a leaf is obvious. Let us consider the case of a Join vertex \( t \) with two children \( t' \) and \( t'' \). To obtain \( Z^*(t, S_0, S_1, ..., S_j, l_1, ..., l_j) \), it is clear that we should consider entries of type \( Z^*(t', S_0, S'_1, ..., S'_j, l'_1, ..., l'_j) \) and \( Z^*(t'', S_0, S''_1, ..., S''_j, l''_1, ..., l''_j) \) (with the same set \( S_0 \)). Consider two such partitions \((S_0, S'_1, ..., S'_j)\) and \((S_0, S''_1, ..., S''_j)\) of the same set \( X_t \). By merging \( S'_a \) and \( S''_b \) if they are adjacent, we get a new partition \((S_0, S_1, ..., S_j)\) of \( X_t \). Let \( S_i = (\cup_{a \in I} S'_a) \cup (\cup_{b \in J} S''_b) \) be one of the subsets of this new partition where \( I \subset \{1, ..., j'\} \) and \( J \subset \{1, ..., j''\} \) are the set of indexes of the merged subsets that led to \( S_i \). Then the size of the connected component containing \( S_i \) is given by \( l_i = |S_i| + \sum_{a \in I} (l'_a - |S'_a|) + \sum_{b \in J} (l''_b - |S''_b|) \). Observe that \( l'_a - |S'_a| \) represents the number of vertices of \( G(Y_{t'}) \) that belong to the same connected components as \( S'_a \) but not to \( X_{t'} = X_t \). These vertices do not belong to \( Y_{t''} \) by the separator property.

If the combination of \( Z^*(t', S_0, S'_1, ..., S'_j, l'_1, ..., l'_j) \) and \( Z^*(t'', S_0, S''_1, ..., S''_j, l''_1, ..., l''_j) \) leads to connected components of size less than or equal to \( k \) (i.e., all numbers \( l_i \) are \( \leq k \)), then we keep in the table of \( t \) the cost \( Z^*(t', S_0, S'_1, ..., S'_j, l'_1, ..., l'_j) + Z^*(t'', S_0, S''_1, ..., S''_j, l''_1, ..., l''_j) - \sum w_v \) obtained with the new partition and the corresponding numbers \( l_i \). Observe that the cost \( \sum w_v \) is subtracted since it is counted twice in \( Z^*(t', S_0, S'_1, ..., S'_j, l'_1, ..., l'_j) \) and \( Z^*(t'', S_0, S''_1, ..., S''_j, l''_1, ..., l''_j) \). Notice that since the same partition
3.3 Paths, trees and cycles

$(S_0, ..., S_j)$ with the corresponding sizes $(l_1, ..., l_j)$ might be obtained by different combinations of $Z^*(t', S_0, S_1', ..., S_j', l_1', ..., l_j')$ and $Z^*(t'', S_0, S_1', ..., S_j', l_1', ..., l_j')$, we should of course keep the one having the lowest cost.

As a conclusion, the table of each Join vertex $t$ can be built in polynomial time using the tables of its children $t'$ and $t''$.

Let us now assume that $t$ is an Introduce vertex whose unique child is $t'$ and let $v \in V(G)$ be the vertex such that $X_t = X_{t'} \cup \{v\}$. Let us consider the entry $Z^*(t, S_0, ..., S_j, l_1, ..., l_j)$ where $v \in S_0$. Then, $Z^*(t, S_0, ..., S_j, l_1, ..., l_j) = Z^*(t', S_0 \setminus \{v\}, S_1, ..., S_j, l_1, ..., l_j) + w_v$. The case where $v \notin S_0$ is slightly more complicated. Assume that $v \in S_{i_0}$ for some $i_0 \neq 0$. Since $v \notin Y_t$ (by property (iii) of tree-decompositions), the deletion of $v$ can split the component containing $S_{i_0}$ into several components and the set $S_{i_0} \setminus \{v\}$ will be partitionned into several subsets $A_1, ..., A_p$ where the vertices of $A_i$ belong to the same connected component of size $l_i'$. Observe that since $v$ is not adjacent to any vertex in $Y_t \setminus X_t$, we should have $l_{i_0} - 1 = \sum l_i'$. This clearly implies that

$$Z^*(t, S_0, ..., S_j, l_1, ..., l_j) = \min_{A_1, ..., A_p | l_0' = l_1' + \ldots + l_p', \sum_{i=1}^{p} l_i' = l_{i_0} - 1, \sum_{i=1}^{p} l_i' \leq k} Z^*(t', S_0, ..., S_{i_0-1}, A_1, ..., A_p, S_{i_0+1}, ..., S_j, l_1, ..., l_{i_0-1}, l_1', ..., l_p', l_{i_0+1}, ..., l_j').$$

The equation above shows again that the table of $t$ can be computed in polynomial time using the table of its child. The case where $v$ is a Forget vertex can be handled in a similar way.

**Proposition 3.1** The $k$-separator problem can be solved in polynomial time for graph with bounded treewidth. This holds even if $k$ is part of the input.

### 3.3 Paths, trees and cycles

A more specialized algorithm is described for paths, trees and cycles.

Let us start with the case where $G = (V, E)$ is a tree denoted $T$. Without loss of generality, we assume that the edges of $T$ are oriented such that $T$ can be seen as a tree rooted at an arbitrary vertex $r$. Let $N^+_v$ denote the set of children of a vertex $v$. The number of children of $v$ is denoted by $d^+_v = |N^+_v|$. The children of $v$ can be assumed to be arbitrarily ordered from the first to the last. Then $v_i$ corresponds to
Chapter 3. Polynomial cases

the $i$th child of $v$. Let $T_v$ be the subtree rooted at $v$. We also use $T^p_v$ to denote the subtree containing $v$ and the $p$ subtrees rooted at the first $p$ children of $v$. Observe that $T_v = T^d_v$.

We will describe a dynamic programming approach to compute a minimum-weight $k$-separator. Let $C_v$ be the weight of an optimal $k$-separator of the subtree $T_v$. The global optimum is of course obtained when $v = r$.

Let us also use $C^\text{in}_v$ to denote the cost of an optimal $k$-separator of $T_v$ under the conditions that $v$ belongs to this $k$-separator.

Observe that when $v$ belongs to the $k$-separator, the subtrees rooted at the children of $v$ become non connected together. In other words, one can write the following:

$$C^\text{in}_v = w_v + \sum_{y \in N^+_v} C_y.$$  \hfill (3.1)

For each number $i$ such that $1 \leq i \leq k$ and each vertex $v \in V$, let $C^\text{out}_v(i)$ be the weight of an optimal $k$-separator under the condition that $v$ does not belong to this separator and there is a connected component of $T_v$ of size exactly equal to $i$ containing $v$ and not belonging to the separator. In other words, we require here that after the deletion of the $k$-separator of $T_v$, $v$ remains in the graph and belongs to a component of size $i$.

Observe that $C_v$ is just given by:

$$C_v = \min \left\{ C^\text{in}_v, \min_{i=1,...,k} C^\text{out}_v(i) \right\}.$$  \hfill (3.2)

We need one more definition. For any vertex $v \in V$, any number $1 \leq i \leq k$ and any number $1 \leq p \leq d^+_v$, let $C^\text{out}_v(p, i)$ be the weight of an optimal $k$-separator of $T_v$ under the following conditions: $v$ does not belong to the separator; after the deletion of the $k$-separator $T^p_v$ contains a connected component of size $i$ including $v$. Observe that:

$$C^\text{out}_v(i) = C^\text{out}_v(d^+_v, i).$$  \hfill (3.3)

It is now easy to see that $C^\text{out}_v(p, i)$ can be expressed as follows:

$$C^\text{out}_v(p, i) = \min \left\{ C^\text{out}_v(p-1, i) + C^\text{in}_{v_p} \min_{j=1,...,i-1} C^\text{out}_v(p-1, i-j) + C^\text{out}_{v_p}(j) \right\}.$$  \hfill (3.4)
3.3. Paths, trees and cycles

In Equation (3.4), $C^{\text{out}}_{vp}(p-1,i) + C^{\text{in}}_{vp}$ represents the situation where $v_p$ (the $p^{th}$ child of $v$) belongs to the $k$-separator of $T_v$ while $T_{p-1}^v$ contains a connected component of size $i$ including $v$ after the removal of the $k$-separator. The second term $\min_{j=1,...,i-1} C^{\text{out}}_{vp}(p-1,i-j) + C^{\text{out}}_{vp}(j)$ clearly corresponds to the case where $v_p$ does not belong to the $k$-separator.

Equation (3.4) can be used in combination with equations (3.1), (3.2), (3.3) to compute all optimal weights. For a vertex $v$, quantities $C^{\text{out}}_{v}(p,i)$ can be computed only when the children of $v$ were already addressed. We should of course start by $p=1$ and increase it until reaching $d^+_v$.

Similarly to many dynamic programming algorithms related to trees, we start by the leaves of the tree, and we go up until we reach the root $r$.

To finish the description of the algorithm we should only observe that when $v$ is a leaf, then $C^{\text{in}}_v = w_v$, $C^{\text{out}}_v(1) = 0$ and $C_v = \min\{w_v,0\}$ while all other quantities are not defined (assumed to be infinite).

The number of quantities to be computed ($C^{\text{out}}_{v}(p,i)$, $C^{\text{in}}_v$, $C^{\text{out}}_v(i)$ and $C_v$) is about $O(nk)$ (using the fact that $\sum_{v \in V} d^+_v = |V| - 1 = n - 1$). The complexity of the algorithm is also easy to estimate. Observe that assuming that all children of a vertex $v$ are already addressed, the additional time required to compute all terms related to $v$ is $O(d^+_v k^2)$. A simple induction implies that the dynamic programming algorithm has a complexity of $O(nk^2)$.

Since paths are also trees, the algorithm described for trees can also be used.

The problem for cycles can also be solved using dynamic programming. If there are vertices with negative weights then they belong to the $k$-separator. By deleting these vertices we get a set of subpaths. Then we can use dynamic programming to solve the problem on each subpath. If all vertices have strictly positive weights and the size of the cycle is less than or equal to $k$, then the optimal separator is the empty set. Finally, if weights are positive and the size of the cycle is strictly greater than $k$, then we select a connected subset of $k+1$ vertices and we solve $k+1$ path problems by deleting from the cycle one vertex belonging to this subset.

**Proposition 3.2** The $k$-separator problem can be solved in polynomial time for trees and cycles. This holds even if $k$ is not constant.
Chapter 3. Polynomial cases

3.4 \( mK_2 \)-free graphs

Before presenting \( mK_2 \)-free graphs, let us introduce a construction \( G^* \) from \( G \) allowing to transform the \( k \)-separator problem into a maximum weight stable set problem.

Given a vertex-weighted graph \( G \), we build a vertex-weighted extended graph \( G^* = (V^*, E^*) \) as follows. Each subset of vertices \( S \subset V \) such that \( 1 \leq |S| \leq k \) and \( G(S) \) is connected, is represented by a vertex in \( G^* \). In others words, \( V^* = \{ S \subset V, |S| \leq k, G(S) \text{ is connected} \} \). The set of edges is defined as follows: \( E^* = \{(S, T), S \in V^*, T \in V^*, S \neq T, \text{ such that either } S \cap T \neq \emptyset, \text{ or } (u, v) \in E \text{ for some } u \in S \text{ and } v \in T \} \). Said another way, \( S \in V^* \) and \( T \in V^* \) are connected by an edge if the subsets of vertices of \( G \) they are representing either have a common vertex or contain two adjacent vertices. The weight of a vertex \( S \in V^* \) is defined by \( w_S = \sum_{v \in S} w_v \).

Let \( R \) be a maximum-weight stable set of \( G^* \). If two vertices \( S \in V^* \) and \( T \in V^* \) belong to this stable set \( R \), then \( S \cap T = \emptyset \) and there are no edges in \( G \) with one endvertex in \( S \) and another endvertex in \( T \). In other words, if we consider \( \cup_{S \in R} S \), we get a set of vertices in \( V \) inducing a subgraph where each connected component has a size less than or equal to \( k \). The complementary set of \( \cup_{S \in R} S \) in \( V \) is a \( k \)-separator for the graph \( G \). This graph construction can be seen as a generalization of a construction proposed by [81] for the dissociation problem \((k = 2)\).

Let us now assume that \( G \) does not contain an induced matching of size \( m \) where \( m \) is a constant. This is equivalent to say that \( G \) is \( mK_2 \)-free. It is shown in [90] that the dissociation problem is easy to solve in this case. Remember that the last is equivalent to the \( k \)-separator problem with \( k = 2 \). We generalize this result for any constant \( k \).

**Proposition 3.3** The \( k \)-separator problem can be solved in polynomial time for \( mK_2 \)-free graphs if we assume that \( m \) and \( k \) are constants.

**Proof:** we consider again the extended graph \( G^* \). Since \( k \) is a constant, \( G^* \) has a polynomial size. We know from [22] that the stable set problem can be solved in polynomial time if the graph is \( mK_2 \)-free. It is then enough to prove that \( G^* \) is \( mK_2 \)-free if \( G \) is \( mK_2 \)-free.

Suppose that \( G^* \) contains an induced matching of size \( m \). Consider an edge \((u, w)\)
3.5. \((G_1, G_2, G_3, P_6)\)-free graphs

Let \(G_1\) be the chair graph (or fork) obtained from the claw by a single subdivision of one if its edges. \(G_1\) is represented on the left of Figure 3.1. It is proved in [82] that the maximum weight stable set problem can be solved in polynomial time if the graph is \(G_1\)-free. Their result is an improvement of the classical result of [75, 58] related to claw-free graphs since the class of \(G_1\)-free graphs includes the class of claw-free graphs.

When \(k=2\), it is proved in [90] that the graph \(G^*\) is \(G_1\)-free if and only if \(G\) is \((G_1, G_2, G_3)\)-free where \(G_2\) and \(G_3\) are shown on Figure 3.1. We are going to extend this result when \(k \geq 3\). More precisely, we will prove that \(G^*\) is \(G_1\)-free if and only if \(G\) is \((G_1, G_2, G_3, P_6)\)-free graph where \(P_6\) is the simple path containing 6 vertices (shown on the right part of Figure 3.1).

**Proposition 3.4** Assuming that \(k \geq 3\), the extended graph \(G^*\) is \(G_1\)-free if and
only if the original graph $G$ is $(G_1, G_2, G_3, P_6)$-free.

**Proof:** It is easy to check that if $G$ contains one of the graphs $G_1$, $G_2$, $G_3$ and $P_6$ as an induced graph, then $G^*$ contains $G_1$. Let us do it for $P_6$. Assume that the vertices of $P_6$ are $\{1, 2, ..., 6\}$ and let $V_0 = \{2, 3, 4\}$, $V_1 = \{1\}$, $V_2 = \{3\}$, $V_3 = \{5\}$ and $V_4 = \{6\}$. Each subset $V_i$ ($i = 0, ..., 4$) induces a connected graph of $G$ with at most $k$ vertices. Considering $V_i$ ($i = 0, ..., 4$) as vertices of $G^*$, the graph induced by $\{V_0, V_1, V_2, V_3, V_4\}$ is clearly a chair. The same kind of constructions can be exhibited for $G_1$, $G_2$ and $G_3$.

Let us now assume that $G^*$ contains $G_1$. We should prove that $G$ necessarily contains one of the graphs $G_1$, $G_2$, $G_3$ and $P_6$ as an induced graph. Among all chairs included in $G^*$, consider a chair induced by $\{V_0, V_1, V_2, V_3, V_4\}$ such that $|V_0| + |V_1| + |V_2| + |V_3| + |V_4|$ is minimum. We assume that $V_0$ is the central vertex of the chair while $V_4$ is adjacent to $V_3$ and not adjacent to $V_0$.

Since $V_1$ only has to be connected to $V_0$ in $G^*$, there is no need for $V_1$ to contain more than one vertex. This holds also for $V_2$. Moreover, $V_4$ must be adjacent to $V_3$ and not adjacent to the rest of vertices. It is clear that $|V_4| = 1$ by the minimality assumption of $\sum_{i=0}^{t=4} |V_i|$. Let then $V_1 = \{v_1\}$, $V_2 = \{v_2\}$ and $V_4 = \{v_4\}$ where $v_1$, $v_2$ and $v_4$ are vertices of $G$.

Suppose that $|V_3| \geq 2$. If all vertices of $V_3$ are adjacent to $V_0$, then consider a vertex $a \in V_3$ that is also adjacent to $v_4$. Observe that we could choose $V_3 = \{a\}$ to still obtain an induced chair in $G^*$. This contradicts the minimality of $\sum_{i=0}^{t=4} |V_i|$. Let us now assume that there are vertices in $V_3$ that are not adjacent to $V_0$. Then, there are at least two adjacent vertices $a$ and $b$ of $V_3$ such that $a$ is adjacent to $V_0$ while $b$ is not adjacent to $V_0$. By taking $V_3 = \{a\}$ and $V_4 = \{b\}$ without changing $V_0$, $V_1$ and $V_2$, we clearly obtain a chair in $G^*$ violating the minimality condition. Consequently, we necessarily have $|V_3| = 1$. Similarly to the other subsets, $V_3$ is denoted by $\{v_3\}$ where $v_3 \in V$. The only subset $V_i$ that might have more than one vertex is $V_0$.

Notice that since $|V_4| = |V_3| = 1$ and $V_4$ is not adjacent to $V_0$, we should also have $V_3 \cap V_0 = \emptyset$.

We will now study all possible situations depending on $V_0$ and how it is connected to $V_1$, $V_2$ and $V_3$.

**Case 1.** If $|V_0| = 1$, then $G(V_0 \cup \{v_1, v_2, v_3, v_4\})$ is clearly a chair.
3.5. \((G_1, G_2, G_3, P_6)\)-free graphs

Case 2. Assume that \(G(V_0)\) contains a cycle. Since deleting any vertex of the cycle does not break the connectivity of \(G(V_0)\), the only reason that can prevent us to decrease the size of \(V_0\) is that for each vertex \(v\) of the cycle, there is a subset \(V_i (i \in \{1, 2, 3\})\) such that \(V_i\) is adjacent to \(v\) (and only to \(v\) in \(V_0\)). This clearly implies that the cycle is in fact a triangle. Moreover, by the minimality assumption, \(V_0\) does not contain vertices outside of the triangle. It is also clear that we cannot simultaneously have \(v_1 \in V_0\) and \(v_2 \in V_0\) since \(V_1\) and \(V_2\) are not adjacent in \(G^*\). Let us consider the two possible subcases.

Subcase 2.1. Assume that \(v_1 \in V_0\) and \(v_2 \notin V_0\), then the graph \(G(V_0 \cup \{v_2, v_3, v_4\})\) is isomorphic to \(G_3\).

Subcase 2.2. If neither \(v_1\) nor \(v_2\) belong to \(V_0\), then \(G(V_0 \cup \{v_2, v_3\})\) is isomorphic to \(G_2\).

Case 3. Assume that \(|V_0| = 2\). Let then \(V_0 = \{a, b\}\) where \(a\) and \(b\) are two adjacent vertices of \(G\). Notice that we neither have \(v_1 \in V_0\) nor \(v_2 \in V_0\). Indeed, if \(v_1 = a \in V_0\), then both \(v_2\) and \(v_3\) are adjacent to \(b\) (and not to \(a\)). Then, we could take \(V_0 = \{b\}\) without changing the other subsets to obtain a chair in \(G^*\). This contradicts the minimality assumption. Let us now consider all possible subcases. Observe that there is some symmetry between \(V_1\) and \(V_2\) which reduces the number of subcases to be studied.

Subcase 3.1. Assume that \(v_3\) is adjacent to both \(a\) and \(b\). This clearly implies that \(v_1 \notin V_0\) and \(v_2 \notin V_0\).

- Suppose that \(a\) is adjacent to both \(v_1\) and \(v_2\), then by taking \(V_0 = \{a\}\) without changing the other subsets, we still obtain a chair in \(G^*\). This contradicts the minimality assumption. Replacing \(a\) by \(b\) leads to the same conclusion.

- Let us now assume that \(a\) is only adjacent to \(v_1\) while \(b\) is only adjacent to \(v_2\). Then, \(G(V_0 \cup \{v_1, v_2, v_3, v_4\})\) is isomorphic to \(G_2\).

Subcase 3.2. Suppose that \(v_3\) is adjacent to \(a\) (and not to \(b\)). Remember that \(v_1 \notin V_0\) and \(v_2 \notin V_0\).

- Assume that both \(v_1\) and \(v_2\) are adjacent to \(a\). Then, by taking \(V_0 = \{a\}\), we still obtain a chair in \(G^*\) contradicting the minimality...
- Let us now assume that \( v_1 \) is only adjacent to \( b \) (and not to \( a \)). Then, if \( v_2 \) is only adjacent to \( a \), \( G(V_0 \cup \{v_1, v_2, v_3\}) \) is isomorphic to \( G_1 \). On the contrary, if \( v_2 \) is adjacent to both \( a \) and \( b \), then \( G(V_0 \cup \{v_1, v_2, v_3, v_4\}) \) is isomorphic to \( G_3 \). Finally, if \( v_2 \) is only adjacent to \( b \), then \( G(V_0 \cup \{v_1, v_2, v_3\}) \) is isomorphic to \( G_1 \).

**Case 4.** Let us now assume that \( 3 \leq |V_0| \leq k \) and \( G(V_0) \) is a tree having at least 3 leaves (vertices of degree 1 in the tree). The minimality assumption require that for each leaf, there is a subset \( V_i \) \((i \in \{1, 2, 3\})\) such that \( V_i \) is only adjacent to this leaf (otherwise, we could reduce the size of \( V_0 \) by delete this leaf). This implies that the number of leaves is exactly equal to 3. Let \( x_1 \) be the unique leaf adjacent to \( v_1 \). By taking \( V_1 = \{x_1\}, V_0 = V_0 \setminus \{x_1\} \) (observe that \( V_0 \) is still connected) without changing the other subsets, we obtain a chair in \( G^* \) contradicting the minimality assumption.

**Case 5.** Suppose that \( 3 \leq |V_0| \leq k \) and \( G(V_0) \) is a simple path. For each one of the two leaves, there is at least one subset \( V_i \) \((i \in \{1, 2, 3\})\) such that \( V_i \) is only adjacent to this leaf. For symmetry reasons, we can assume without loss of generality that \( v_1 \) is only adjacent to \( x_1 \) (so \( x_1 \) is a leaf). Observe that this implies that \( v_1 \notin V_0 \). Let \( x_2 \) (resp. \( x_3 \)) be a vertex of \( V_0 \) such that \( v_2 \) (resp. \( v_3 \)) is adjacent to \( x_2 \) (resp. \( x_3 \)). We will study all possible situations depending on the positions of \( x_2 \) and \( x_3 \).

**Subcase 5.1.** Assume that \( x_2 \in ]x_1, x_3[ \). Then \( x_3 \) is the second leaf of the path. By the minimality assumption, \( x_3 \) is not adjacent to \( V_0 \setminus \{x_3\} \) (otherwise \( |V_0| \) can be reduced by eliminating \( x_3 \)). Suppose that there is a vertex \( y \in [x_1, x_3[ \) such that \( v_2 \) and \( y \) are not adjacent. Then, but taking \( V_1 = \{y\}, V_0 = ]y, x_3[ \) without changing the other subsets, we get a chair in \( G^* \) contradicting the minimality assumption. Then, \( v_2 \) is adjacent to all vertices of \( [x_1, x_3[ \). In a similar way, one can prove that \( v_2 \) is adjacent to \( x_3 \). In other words, \( v_2 \) is adjacent to all vertices of \( V_0 \). One can see now that \( G(\{v_1, x_1, v_2, x_3, v_3, v_4\}) \) is isomorphic to \( P_6 \).

**Subcase 5.2.** Assume that \( x_2 = x_1 \). To avoid the previous subcase, we should also assume that \( v_2 \) is not adjacent to \( V_0 \setminus \{x_2 = x_1\} \). Remember that
3.6. Interval-filament, asteroidal triple-free and weakly chordal graphs

$v_1$ is only adjacent to $x_1$. It is also clear that if $v_3$ is adjacent to any vertex $y$ of $[x_1,x_3]$, then $|V_0|$ can be reduced by eliminating $x_3$. Let us then change the subsets $V_i$ as follows: take $V_4 = \{v_3\}$, $V_3 = \{x_3\}$, $V_0 = V_0 \setminus \{x_3\} = [x_1,x_3]$ without changing $V_1$ and $V_2$. It is easy to check that the graph induced by $\{V_0, V_1, V_2, V_3, V_4\}$ is a chair of $G^*$. This contradicts the minimality assumption.

**Subcase 5.3.** Assume that $x_3 = x_2$. To avoid the two previous subcases, we assume that $v_2$ is not adjacent to $V_0 \setminus \{x_2\}$. This implies that $v_2 \notin V_0$. Notice that $v_3$ is not adjacent to any vertex $y \in [x_1,x_2]$, because, if not, we could take $V_1 = \{y\}$, $V_0 = \{y,x_2\}$ without changing the other subsets to get a chair in $G^*$ contradicting the minimality assumption. Observe that the graph $G(\{v_4,v_3,x_1,v_1\})$ is a chair.

**Subcase 5.4.** Let us now assume that $x_3 \notin [x_1,x_2]$. By the minimality assumption, we deduce that $v_2$ is not adjacent to $V_0 \setminus \{x_2\}$. If $v_3$ is adjacent to $x_2$, then we get the previous subcase. Let us then assume that $v_3$ is not adjacent to $x_2$. By taking $V_2 = \{x_2\}$, $V_0 = V_0 \setminus \{x_2\}$ without changing the other subsets, we obtain a chair in $G^*$ contradicting the minimality assumption.

\[
\]

**Corollary 3.1** Assuming that $k$ is a constant $\geq 3$, the $k$-separator problem can be solved in polynomial time for $(G_1,G_2,G_3,P_6)$-free graphs.

**Proof:** If $k$ is a constant, then $G^*$ has a polynomial size. Using Proposition 3.4 and the algorithm of [82] to compute a maximum weight stable set problem, one can solve the $k$-separator problem in polynomial time. ■

3.6 Interval-filament, asteroidal triple-free and weakly chordal graphs

The results of this section are a direct consequence of the results of [39]. Given a graph $G$ and a family $\mathcal{H}$ of fixed connected graphs, a $\mathcal{H}$-packing of $G$ is a pairwise node-disjoint set of subgraphs of $G$, each isomorphic to a member of $\mathcal{H}$ [39]. If we add the requirement that each two subgraphs of the packing are not joined by
Chapter 3. Polynomial cases

edges, we get independent $H$-packings. To study this problem, a graph $\mathcal{H}(G)$ is introduced in [39]. Each subgraph of $G$ which is isomorphic to a member of $\mathcal{H}$ is represented by a vertex of $\mathcal{H}(G)$, and two vertices are adjacent if the two subgraphs either intersect or are joined by an edge.

Consider a collection of intervals on a line $L$. Suppose that for each interval, we are given a curve above the line, connecting the endpoints of the interval, and remaining within the limits of the interval. An interval-filament graph (see figure 3.2) is the intersection graph of such a collection of intervals [29]. Computing a maximum weight stable set in interval-filament graph can be done in polynomial time [29]. It is proved in [39] that if $G$ is an interval-filament graph, then $\mathcal{H}(G)$ is also an interval-filament graph. In other words, the class of interval-filament graphs is closed under the operation $G \rightarrow \mathcal{H}(G)$. Notice that the class of interval-filament graphs includes polygon-circle graphs and cocomparability graphs.

The same was also proved in [39] for the class of weakly chordal graphs [33] (graphs such that neither the graph nor its complement contain an induced cycle on 5 or more vertices, see figure 3.4) and the class of asteroidal triple-free graphs (graphs not containing an asteroidal triple defined as a stable set of 3 vertices such

Figure 3.2: Example of an interval-filament graph

Figure 3.3: Asteroidal triple-free graph

\begin{align*}
&c1 = (y1; x2; x3; x4; x5; y9); c2 = (y3; x1; y5); c3 = (y6; x3; x4; y8); c4 = (y2; x6; y4); c5 = (y7; x7; x8; y8).
&a(v) = c1 \cup c2; a(e) = c2; a(u) = c4; a(t) = c3; a(r) = c3 \cup c5.
\end{align*}
that between each pair of vertices of this triple, there is path connecting them and avoiding the neighborhood of the third vertex, see figure 3.3). We know from [30] that the maximum weight stable set problem can be solved in polynomial time for asteroidal triple-free graphs. The same holds for weakly chordal graphs (see, e.g., [38]).

Let us now go back to our $k$-separator problem and let us slightly change the definition of $\mathcal{H}$ by allowing it to depend on $G$. More precisely, let $\mathcal{H}$ be the set of all connected subgraphs of $G$ containing at most $k$ vertices. Then, $\mathcal{H}(G)$ is exactly our graph $G^*$. Consequently, the results of [39] can be directly applied here to deduce that the problem is easy to solve. We only have to ensure that the size of $G^* = \mathcal{H}(G)$ is polynomially bounded. This of course occurs if $k$ is a constant.

**Proposition 3.5** Assuming that $k$ is a constant, the $k$-separator problem can be solved in polynomial time for interval-filament, asteroidal triple-free and weakly chordal graphs.

### 3.7 Interval and circular-arc graphs

Interval graphs are graphs where a vertex corresponds to an interval and an edge $(u, v)$ exists if there is a non-empty intersection between the intervals represented by $u$ and $v$, see figure 3.5. We prove below that the $k$-separator problem is easy to solve for interval graphs.

Interval graphs are obviously interval-filament and are also chordal. So the results of Section 3.6 can be applied here to deduce that the $k$-separator problem can be solved in polynomial-time for this class of graphs. However, in Section 3.6, $k$ is required to be constant. This was necessary to get a graph $G^*$ with a polynomial size. We will prove in this section that the problem is easy to solve even if $k$ is part...
Chapter 3. Polynomial cases

Figure 3.5: Example of an interval and circular-arc graphs

Given a graph $G$, one can check in linear time if the graph is an interval graph and provide a family $\mathcal{I}$ of intervals such the graph is the intersection graph of the family \cite{52}. We can obviously assume that for each pair of intervals $[a, b]$ and $[c, d]$ of $\mathcal{I}$, the endpoints are different ($a \neq b \neq c \neq d$). Let $[a_w, b_w]$ be the interval related to vertex $w \in V$. Then, $\mathcal{I} = \{[a_w, b_w] : w \in V\}$.

When $G^*$ is built, the number of vertices can be non-polynomial. However, since each vertex $v^*$ of $G^*$ corresponds to a connected graph of $G$, and each vertex $w$ of $G$ corresponds to an interval, one can associate to $v^*$ the union of the intervals $\bigcup_{w \in v^*} [a_w, b_w]$. The connectivity of the subgraph related to $v^*$ clearly implies that $\bigcup_{w \in v^*} [a_w, b_w]$ is an interval. Two vertices $v^*$ and $u^*$ are adjacent in $G^*$ if and only if the two intervals associated with $v^*$ and $u^*$ intersect: $\bigcup_{w \in v^*} [a_w, b_w] \cap \bigcup_{w \in u^*} [a_w, b_w] \neq \emptyset$.

While the number of vertices of $G^*$ can be non-polynomial, the number of intervals that can be obtained as a union of intervals of $\mathcal{I}$ is polynomial (quadratic). In other words, for an interval $[x, y]$ where $x$ and $y$ belong to $\bigcup_{w \in V} \{a_w, b_w\}$, we might have many vertices $v^*$ for which $\bigcup_{w \in v^*} [a_w, b_w] = [x, y]$. However, a stable set in $G^*$ cannot simultaneously contain $v^*$ and $u^*$ if $\bigcup_{w \in v^*} [a_w, b_w] = \bigcup_{w \in u^*} [a_w, b_w]$ since $u^*$ and $v^*$ are adjacent in $G^*$.

It becomes now clear that instead of building $G^*$, we should consider a more restricted graph $G^{**}$, where all vertices $v^*$ having the same $\bigcup_{w \in v^*} [a_w, b_w] = [x, y]$ are represented by only one vertex $v_{[x,y]}$. Two vertices $v_{[x,y]}$ and $v_{[a,b]}$ are adjacent
3.7. Interval and circular-arc graphs

if they intersect. The graph $G^{**}$ obviously has a polynomial size.

In order to transform the maximum weight stable set problem in $G^*$ into a maximum weight stable set problem in $G^{**}$, we have to define the weight of a vertex $v_{[x,y]}$ of $G^{**}$.

Observe that $v_{[x,y]}$ exists if the interval $[x,y]$ is the exact union of at most $k$ intervals of $\mathcal{I}$. Since a weight $w_v$ is associated with each interval $[a_v,b_v] \in \mathcal{I}$, the weight of $v_{[x,y]}$ is given by the maximum weight of at most $k$ intervals of $\mathcal{I}$ whose union is equal to $[x,y]$. More precisely, for each interval $[x,y]$ where $x$ and $y$ belong to $\bigcup_{v \in V \setminus \{a_v,b_v\}} \mathcal{I}$, we should solve the problem

$$\max_{A \subset V \mid |A| \leq k, [x,y] = \bigcup_{v \in A} [a_v,b_v]} \sum_{v \in A} w_v. \quad (3.5)$$

If (3.5) does not have a solution, then $[x,y]$ is not represented by a vertex in $G^{**}$. Otherwise, the weight of $v_{[x,y]}$ is equal to the maximum objective value of (3.5). We show below that (3.5) can be solved in polynomial time. For an interval $[a,b] \in \mathcal{I}$, we will use $w_{[a,b]}$ to denote the weight $w_v$ of the vertex $v \in V$ representing this interval.

**Lemma 3.1** Problem (3.5) can be solved in polynomial time by dynamic programming.

**Proof:** First, observe that all intervals $[a_v,b_v] \in \mathcal{I}$ that are not included in $[x,y]$ can be eliminated when we are solving (3.5). Let $S = \{c_1 = x, c_2, ..., c_r = y\}$ be the set of endpoints of the intervals included in $[x,y]$: $S = \bigcup_{[a_v,b_v] \subseteq [x,y]} \{a_v,b_v\}$. The sequence $(c_i)_{1 \leq i \leq r}$ is an increasing one. Notice that we can assume that $c_1 = x$ and $c_r = y$, since otherwise problem (3.5) does not have a solution.

The cardinality of $S$ denoted by $r$ is of course less than $2|V|$. Let $O \subset \{1, ..., r\}$ be the subset of indexes $j$ such that there exists $v \in V$ satisfying $[a_v,b_v] \subseteq [x,y]$ and $a_v = c_j$. In this case, let $j + \delta(j)$ be the index such $b_v = c_{j+\delta(j)}$. Thus, if $j \in O$, then $1 \leq \delta(j) \leq r - j$.

Figure 3.6 illustrates the definitions where we have $r = 16$, $O = \{1, 2, 3, 6, 7, 8, 11, 14\}$, $\delta(1) = 4, \delta(2) = 8, \delta(3) = 1, \delta(6) = 3$, etc. Assume that $k = 6$ and suppose that the optimal solution of problem (3.5) is given by the intervals represented by thick arrows in Figure 3.6. The intervals belonging to the optimal solution can be numbered.
Chapter 3. Polynomial cases

Figure 3.6: On the dynamic programming approach to solve problem (3.5) according to the order of their starting points. In Figure 3.6, they are numbered from 1 to 6. Let us, for example, consider the 3 first intervals belonging to the optimal solution. According to Figure 3.6, these 3 intervals cover the interval \([c_1, c_{10}]\). To reach \(y = c_{16}\), it is clear that we should at least cover the interval \([c_{10}, c_{16}]\) using intervals starting after \(c_4\) (because the third interval starts at \(c_3\)). Since we already used 3 intervals to reach \(c_{10}\), we should use at most \(k - 3 = 3\) intervals to reach \(y = c_{16}\). Intervals numbered from 4 to 6 necessarily constitute an optimal solution of the problem that consists in covering \([c_{10}, c_{16}]\) by no more than 3 intervals starting after \(c_4\).

The simple observation made above directly leads to a dynamic programming approach. To make things more precise, let us introduce further notation. Let \(i_0\) and \(i_1\) be two integer numbers such that \(1 \leq i_0 \leq i_1 \leq r\). For any integer number \(1 \leq l \leq k\), let \(f(i_0, i_1, l)\) be the maximum weight that we can have to cover the interval \([c_{i_1}, y]\) using at most \(l\) intervals among \(\{[a_w, b_w] : w \in V, c_{i_0} \leq a_w, b_w \leq y\}\).

If \(i_1 < r\), it is clear that to cover \([c_{i_1}, y]\), we need at least one interval belonging to \(\{[a_w, b_w] : w \in V, c_{i_0} \leq a_w, b_w \leq y\}\). This clearly leads to the following induction formula:

\[
f(i_0, i_1 < r, l) = \max_{j \in O : i_0 \leq j \leq i_1} w_{[c_j, c_{j + \delta(j)}]} + f(j + 1, \max(j + \delta(j), i_1), l - 1). \tag{3.6}
\]

If \(O \cap \{i_0, i_0 + 1, ..., i_1 < r\} = \emptyset\), then \(f(i_0, i_1 < r, l) = -\infty\). If \(l = 0\), we also have \(f(i_0, i_1 < r, 0) = -\infty\).

If \(i_1 = r\), then \(y = c_r\) is already reached. The induction formula is then given by:

\[
f(i_0, r, l) = \max_{j \in O : i_0 \leq j \leq r} w_{[c_j, c_{j + \delta(j)}]} + f(j + 1, r, l - 1). \tag{3.7}
\]
Problem (3.5) is solved by computing \( f(1, 1, k) \). The complexity of the dynamic programming algorithm is obviously given by \( O(kn^3) \).

**Proposition 3.6** The \( k \)-separator problem can be solved in polynomial time for interval graphs. This holds even if \( k \) is not constant.

**Proof:** We already observed that the size of the graph \( G^{**} \) is polynomially bounded. Since problem (3.5) can be solved in polynomial time, the weight of each vertex of \( G^{**} \) is easy to compute. Then, we only have to solve the maximum weight stable set problem in \( G^{**} \). Using the fact that this problem is easy to solve for interval graphs, concludes the proof.

Circular-arc graphs are a simple generalization of interval graphs. They are defined by the intersection graphs of a set of arcs on the circle. The previous proposition and the algorithm described in the proof of Lemma 3.1 can be generalized in an obvious way.

**Proposition 3.7** The \( k \)-separator problem can be solved in polynomial time for arc-circular graphs. This holds even if \( k \) is not constant.

### 3.8 Conclusion

Many cases where the \( k \)-separator problem can be solved in polynomial time are shown in this chapter. In the next chapter many integer formulations of the \( k \)-separator problem will be investigated.
Chapter 4

Integer Formulations

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>71</td>
</tr>
<tr>
<td>4.2</td>
<td>Basic formulation</td>
<td>72</td>
</tr>
<tr>
<td>4.3</td>
<td>Stable set formulations</td>
<td>72</td>
</tr>
<tr>
<td>4.4</td>
<td>Metric formulations</td>
<td>73</td>
</tr>
<tr>
<td>4.5</td>
<td>Projected metric formulation</td>
<td>75</td>
</tr>
<tr>
<td>4.6</td>
<td>Partitioning formulations</td>
<td>77</td>
</tr>
<tr>
<td>4.7</td>
<td>Conclusion</td>
<td>79</td>
</tr>
</tbody>
</table>

4.1 Introduction

This chapter is organized as follows. In the section 4.2, we present a basic formulation in the space of original variables indexed on the vertices of graph G. We study, in the section 4.3, a stable set formulation based on a graph transformation. The section 4.4 focuses on a metric formulation containing among these, triangle inequalities. A projected metric formulation is presented in section 4.5. The section 4.6 presents partitioning formulations containing variables associated with a partition of the vertex set. We conclude in the last section.
4.2 Basic formulation

Let $S$ be a subset of vertices such that $|S| = k + 1$ and $G(S)$ is connected. Then, the following inequality is obviously valid for $S_k(G)$.

$$\sum_{v \in S} x_v \geq 1.$$  (4.1)

The $k$-separator problem can be formulated as the following integer program:

$$\text{min} \sum_{v \in V} w_v x_v$$
$$\sum_{v \in S} x_v \geq 1, \quad \forall S \subset V, |S| = k + 1, G(S) \text{connected}$$
$$x_v \in \{0, 1\}, \forall v \in V$$

Let $LP1$ denote the linear relaxation of $IP1$. We will see in Section 6.2 that inequalities (4.1) are generally difficult to separate when $k$ is part of the input.

4.3 Stable set formulations

These formulations are based on the $G^*$ construction of Section 3.4. Remember that $V^* = \{S \subset V : |S| \leq k, G(S) \text{ is connected}\}$ and $E^* = \{(S, T) : S \in V^*, T \in V^*, S \neq T, \text{ such that either } S \cap T \neq \emptyset, \text{ or } (u, v) \in E \text{ for some } u \in S \text{ and } v \in T\}$. The connection with the stable set problem made in Section 3.4 directly leads to the following formulation.

$$\text{IP2} \begin{cases} 
\min \sum_{v \in V} w_v x_v \\
x_v = 1 - \sum_{S \in V^*, v \in S} y_S, \forall v \in V \\
y_S \in \{0, 1\}, \forall S \in V^* \\
y_S + y_T \leq 1, \forall S \in V^*, T \in V^*, (S, T) \in E^* 
\end{cases}$$

Let $Q_v = \{S \in V^* : v \in S\}$. One can add to $IP2$ the obvious valid inequalities $\sum_{S \in Q_v \cup Q_w} y_S \leq 1, \forall (v, w) \in E$. The number of these inequalities is $(|E|)$. This leads
to formulation $IP_3$.

$$IP_3 \begin{cases} \min \sum_{v \in V} w_v x_v \\ x_v = 1 - \sum_{S \in Q_v} y_S, \forall v \in V \\ \sum_{S \in Q_v \cup Q_w} y_S \leq 1, \forall (v, w) \in E \\ y_S \in \{0, 1\}, \forall S \in V^* \end{cases}$$

Let $LP_3$ denote the linear relaxation of $IP_3$. Let $F_1$ (resp. $F_3$) be the set of feasible solutions of $LP_1$ (resp. $LP_3$) with respect to variables $(x_v)_{v \in V}$.

Proposition 4.1 The following inclusion holds: $F_3 \subseteq F_1$.

Proof: Let $(x, y)$ stand for a feasible solution of $LP_3$. Let $C$ denote a connected component of size $k + 1$ in the original graph $G = (V, E)$. So we have: $\sum_{v \in C} x_v = k + 1 - \sum_{v \in C} \sum_{S \in Q_v} y_S$. Notice that in the last expression each variable $y_S$ such that $S$ has a nonempty intersection with $C$ occurs exactly $|S \cap C|$ times.

Let $T$ denote a spanning tree of $C$ (in the original graph) and consider the following quantity: $\sum_{(v, w) \in T} \sum_{S \in Q_v \cup Q_w} y_S$. Notice that in the last expression, the number of times a variable $y_S$ occurs is equal to the number of edges of $T$ that intersect with $S$, and thus is larger than or equal to $|S \cap C|$. From this we deduce that $\sum_{v \in C} \sum_{S \in Q_v} y_S \leq \sum_{(v, w) \in T} \sum_{S \in Q_v \cup Q_w} y_S$. Moreover, using the feasibility of $(x, y)$, we can write that $\sum_{(v, w) \in T} \sum_{S \in Q_v \cup Q_w} y_S \leq k$.

Combining the two previous inequalities leads to $\sum_{v \in C} \sum_{S \in Q_v} y_S \leq k$. Consequently, inequality $\sum_{v \in C} x_v \geq 1$ holds. In other words, $x$ is a feasible solution of $LP_1$. ■

4.4 Metric formulations

A metric formulation is proposed in [53]. In addition to variables $(x_i)_{i \in V}$, we consider a variable $x_{ij}$ for each pair of vertices $\{i, j\}$ to indicate whether $i$ and $j$ belong to the same component. More precisely, $x_{ij}$ is equal to 0 if they are in the same component. Then triangle inequalities are clearly valid. Moreover, to express the fact that a connected component does not contain more than $k$ vertices, we can add the constraints $\sum_{j \in V \setminus \{i\}} x_{ij} \geq n - k$, $\forall i \in V$. Finally, we must add constraints to impose that if two adjacent vertices are not in the $k$-separator, then they belong
4.4. Metric formulations

to the same component: \( x_i + x_j - x_{ij} \geq 0, \forall (i, j) \in E \). The formulation is given below.

\[
\begin{align*}
\text{IP4} & \quad \min \sum_{v \in V} w_v x_v \\
& \quad x_{ij} \leq x_{ik} + x_{jk}, \forall i, j, k \in V \\
& \quad \sum_{j \in V \setminus \{i\}} x_{ij} \geq n - k, \forall i \in V \\
& \quad x_i + x_j - x_{ij} \geq 0, \forall (i, j) \in E \\
& \quad 0 \leq x_{ij} \leq 1, \forall i, j \in V \\
& \quad x_i \in \{0, 1\}, \forall i \in V
\end{align*}
\]

Observe that the \( x_{ij} \) variables are not required to be integer. In fact, as noticed in [53], relaxing the integrity constraint of \( x_{ij} \) variables does not modify the solution of \( \text{IP4} \). The polytope related to formulation \( \text{IP4} \) is studied in [53] and many valid inequalities and facets are presented there. Since some of these inequalities are also valid for \( S_k(G) \), they will be presented in Section 5.4.

Let us present a new way to strengthen the linear relaxation of \( \text{IP4} \). First, constraint \( \sum_{j \in V \setminus \{i\}} x_{ij} \geq n - k \) can obviously be strengthened into \( \sum_{j \in V \setminus \{i\}} x_{ij} \geq n - k + (k - 1)x_i \).

Let \( p \) be any simple path joining \( i \) and \( j \). Remember that \( x(p) \) denotes the sum of \( x_v \) values for all vertices belonging to \( p \) (including \( i \) and \( j \)). It is clear that \( x(p) \geq x_{ij} \) is a valid inequality: if \( x(p) = 0 \), then all vertices of \( p \) do not belong to the \( k \)-separator, so they will be in the same component implying that \( x_{ij} = 0 \). Let \( \text{IP5} \) be the obtained integer formulation and let \( \text{LP5} \) be its linear relaxation.

\[
\begin{align*}
\text{LP5} & \quad \min \sum_{v \in V} w_v x_v \\
& \quad \sum_{j \in V \setminus \{i\}} x_{ij} \geq n - k + (k - 1)x_i, \forall i \in V \\
& \quad x(p) - x_{ij} \geq 0, \forall i, j \in V, \ p \in P_{ij} \\
& \quad 0 \leq x_{ij} \leq 1, \forall i, j \in V \\
& \quad 0 \leq x_i \leq 1, \forall i \in V
\end{align*}
\]

Observe that we do not consider triangle inequalities in \( \text{LP5} \). In fact, it is clear that there is nothing against taking \( x_{ij} = \min \left( 1, \min_{p \in P_{ij}} x(p) \right) \). Triangle inequalities are then naturally satisfied. In other words, adding triangle inequalities does not improve the relaxation.

Notice that constraints \( x(p) - x_{ij} \geq 0 \) can be separated by computing shortest
paths. We will show in Section 4.5 that formulation \( LP5 \) can be easily projected on the space of \( x_i \) variables.

Constraints \( x(p) - x_{ij} \geq 0 \) can also be induced by adding for each pair of vertices a variable \( y_{ij} \) representing the length of the shortest path between \( i \) and \( j \) in sense of \( x_v \) values. Then, we should write that \( y_{ij} = x_i + x_j \) if \( i \) and \( j \) are adjacent, and \( y_{ij} \leq x_i + y_{kj} \) if \( (i, k) \in E \). For more clearness, we give below the new compact linear formulation.

\[
LP6 \begin{cases}
\min \sum_{v \in V} w_v x_v \\
\sum_{j \in V \setminus \{i\}} x_{ij} \geq n - k + (k - 1)x_i, \forall i \in V \\
y_{ij} = x_i + x_j, \forall (i, j) \in E \\
y_{ij} \leq x_i + y_{kj}, \forall i, j \in V, (i, k) \in E \\
y_{ij} - x_{ij} \geq 0, \forall i, j \in V \\
0 \leq x_{ij} \leq 1, 0 \leq y_{ij}, \forall i, j \in V \\
0 \leq x_i \leq 1, \forall i \in V
\end{cases}
\]

\( LP5 \) and \( LP6 \) are obviously equivalent.

4.5 Projected metric formulation

Let \( S \) be a set of vertices with \( |S| \geq k \) and let \( i \in S \). For each \( j \in S \), let \( p_{ij} \in P_{ij} \) be a path joining \( i \) and \( j \). Notice that \( p_{ij} \setminus \{i\} \) is a path joining \( j \) and the neighbor of \( i \) in \( p_{ij} \). Consider the following inequality

\[
(|S| + 1 - k)(1 - x_i) \leq \sum_{j \in S} x(p_{ij} \setminus \{i\}). \tag{4.2}
\]

Lemma 4.1 Inequalities (4.2) are valid for \( S_k(G) \).

Proof: if \( x_i = 1 \), then inequality (4.2) obviously holds. Let us now assume that \( x_i = 0 \). This is equivalent to say that \( i \) does not belong to the \( k \)-separator. Let \( j \in S \). If there is a path \( p_{ij} \) such that \( x(p_{ij} \setminus \{i\}) = 0 \), then \( j \) and \( i \) belong to the same connected component after the removal of the \( k \)-separator. The number of such vertices is less than or equal to \( k - 1 \) since \( i \) is already in the component. In other words, we necessarily have \( (|S| + 1 - k) \leq \sum_{j \in S} x(p_{ij} \setminus \{i\}) \). ■
4.5. Projected metric formulation

We will show in Section 6.2 that inequalities (4.2) can be separated in polynomial time.

Let us now consider a formulation based on inequalities (4.2).

\[
IP7 \begin{cases}
\min \sum_{v \in V} w_v x_v \\
\sum_{v \in S'} x_v = (|S| + 1 - k)(1 - x_i), \forall i \in V, S \subseteq V \setminus \{i\}, |S| \geq k; p_{ij} \in P_{ij}, \forall j \in S
\end{cases}
\]

\[x_v \in \{0, 1\}, \forall v \in V\]

**Lemma 4.2** Formulation \(IP7\) is exact.

**Proof:** The solution of \(IP7\) is integer. Since we already proved the validity of inequalities (4.2), we do not eliminate the incidence vector of any \(k\)-separator. To prove the exactness of \(IP7\), it is enough to prove that \(\sum_{v \in S'} x_v \geq 1\) for any subset \(S' \subseteq V\) with \(|S'| = k + 1\) and \(G(S')\) connected. Let us consider such a subset \(S'\) and let \(i\) be any vertex of \(S'\). For each \(j \in S = S' \setminus \{i\}\), let \(p_{ij}\) be a path joining \(i\) and \(j\) and contained in \(G(S')\) (this is possible by the connectivity of \(G(S')\)). If \(x_i = 1\), then \(\sum_{v \in S'} x_v \geq 1\) is clearly satisfied. Let us now assume that \(x_i = 0\). Inequality (4.2) in addition to the integrity constraint imply that \(x(p_{ij} \setminus \{i\}) \geq 1\) for some \(j \in S\). Since all vertices of \(p_{ij} \setminus \{i\}\) are inside \(S'\), this leads to \(\sum_{v \in S'} x_v \geq 1\).

Let \(LP7\) be the linear relaxation of \(IP7\).

**Proposition 4.2** Formulation \(LP7\) is equivalent to formulations \(LP5\) and \(LP6\). It is then stronger than formulation \(LP4\).

**Proof:** We know that \(LP5\) and \(LP6\) are equivalent. They both dominate \(LP4\). Let us then prove that \(LP7\) is equivalent to \(LP5\).

Let \(x^*\) be an optimal solution of \(LP7\). Let \(x^*_{ij} = \min \left(1, \min_{p \in P_{ij}} x^*(p)\right)\) for \(i \in V, j \in V \setminus \{i\}\). Then, inequalities \(x(p) - x_{ij} \geq 0\) are naturally satisfied for any path \(p \in P_{ij}\).

Let \(i\) be any vertex and let \(S\) be the subset of vertices such that \(x^*_{ij} < 1\). Thus, if \(j \in S\), there exists a path \(p_{ij} \in P_{ij}\) such that \(x^*(p_{ij}) = x^*_{ij}\). Using the fact that \(x^*\) is a solution of \(LP7\), one can write that \((|S| + 1 - k)(1 - x^*_{ij}) \leq \sum_{j \in S} x^*(p_{ij} \setminus \{i\}) = \sum_{j \in S} (x^*_{ij} - x^*_{i})\). Consequently, we have \(\sum_{j \in S} x^*_{ij} \geq (|S| + 1 - k) + (k - 1)x^*_{i}\). Moreover, we know by the definition of \(S\) that \(x^*_{ij} = 1\) if \(j \in S\). Then, the last inequality is
equivalent to \( \sum_{j \in V \setminus \{i\}} x^*_{ij} \geq (n-k) + (k-1)x^*_i \). Consequently, all constraints of LP5 are satisfied.

Let us now prove the opposite sense by considering an optimal solution of LP5 defined by \((x^*_i)_{i \in V}\) and \((x^*_{ij})_{i,j \in V}\). Let \(i\) be an arbitrary vertex, \(S\) be a subset of vertices of \(V \setminus \{i\}\) of cardinality at least \(k\), and \(p_{ij} \in P_{ij}\) be an arbitrary path joining \(i\) and \(j\) for each \(j \in S\). We aim to prove that inequality (4.2) is satisfied. Since we are considering a solution of LP5, we can write that \(\sum_{j \in S} x^*(p_{ij}) \geq \sum_{j \in S} x^*_{ij} = \sum_{j \in V \setminus \{i\}} x^*_{ij} - \sum_{j \in V \setminus \{i\} \cup S} x^*_{ij}\). Using the fact that \(\sum_{j \in V \setminus \{i\}} x^*_{ij} \geq (n-k) + (k-1)x^*_i\), and \(\sum_{j \in V \setminus \{i\} \cup S} x^*_{ij} \leq n - 1 - |S|\), the previous inequality becomes \(\sum_{j \in S} x^*(p_{ij}) \geq (k-1)x^*_i + (|S| + 1 - k)\) which is exactly inequality (4.2).

Notice that it is not difficult to show that there is not any general domination result relating LP7 and LP1 (no formulation is dominating the other one in general). This can also be deduced from the results of Chapter 6 where we see that inequalities (4.2) and inequalities (4.1) induce facets under some conditions.

### 4.6 Partitioning formulations

Another natural formulation for the \(k\)-separator problem can be inspired from partitioning or clustering problems [44, 53]. Let \(B\) be an upper bound of the number of connected components that will be obtained after the removal of the \(k\)-separator. \(B\) can be, for example, equal to \(n\). Components are then numbered from 1 to \(B\). A variable \(z_{ib}\) is considered for each vertex \(i\) and each component \(b \in \{1, ..., B\}\). \(z_{ib}\) will be equal to 1 if \(i\) belongs to component \(b\). The formulation follows.

\[
\text{IP8} \begin{cases}
\min \sum_{i \in V} w_i x_i \\
x_i + \sum_{b=1}^{B} z_{ib} = 1, \forall i \in V \\
\sum_{i \in V} z_{ib} \leq k, \forall b \in \{1, ..., B\} \\
z_{ib} + z_{jb'} \leq 1, \forall (i,j) \in E, b, b' \in \{1, ..., B\}, b \neq b' \\
x_i \in \{0, 1\}, \quad z_{ib} \in \{0, 1\}, \forall i \in V, b \in \{1, ..., B\}
\end{cases}
\]

The first set of constraints expresses the fact that a vertex \(i\) either belongs to the \(k\)-separator \((x_i = 1)\) or to one of the remaining components. The second set of constraints allows to bound the size of each component, while constraints
4.6. Partitioning formulations

$z_{ib} + z_{jb'} \leq 1$ guarantee that adjacent vertices do not belong to different components.

In fact, constraints $z_{ib} + z_{jb'} \leq 1$ are dominated by constraints $z_{ib} - z_{jb} \leq x_j$ for any $(i, j) \in E$ and $b \in \{1, ..., B\}$. Another compact exact formulation can then be obtained.

$$\text{Objective: } \min \sum_{i \in V} w_i x_i$$

$$x_i + \sum_{b=1}^{B} z_{ib} = 1, \ \forall i \in V$$

$$\sum_{i \in V} z_{ib} \leq k, \ \forall b \in \{1, ..., B\}$$

$$z_{ib} - z_{jb} \leq x_j, \ \forall (i, j) \in E, b \in \{1, ..., B\}$$

$$x_i \in \{0, 1\}, \ z_{ib} \in \{0, 1\}, \ \forall i \in V, b \in \{1, ..., B\}$$

A further improvement is obtained by considering a subset $U \subset \{1, ..., B\}$, its complement $\overline{U}$ and two adjacent vertices $i$ and $j$:

$$\sum_{b \in U} z_{ib} + \sum_{b \in \overline{U}} z_{jb} \leq 1, \ \forall (i, j) \in E. \quad (4.3)$$

We will see in Section 6.2 that inequalities (4.3) can be separated in polynomial time.

Let us now go back to the definition of $B$. Remember that $B$ is an upper bound of the number of remaining components after the removal of the $k$-separator. It is clear that we should take $B$ as small as possible to improve the quality of the relaxations. The maximum number of connected components that can be obtained after the deletion of a $k$-separator is in fact exactly equal to the maximum size of a stable set in $G^*$ (defined in Section 3.4). We show below that this number is in fact exactly equal to the maximum size of a stable set in $G$.

**Proposition 4.3** The lowest upper bound $B$ that can be considered in formulations $\text{IP8}$ and $\text{IP9}$ is equal to the maximum size of a stable set in $G$

**Proof:** Since $G$ is included in $G^*$, the maximum size of a stable set in $G^*$ is larger than or equal to the maximum size of a stable set in $G$. Consider a stable set of $G^*$. Each vertex $S \in V^*$ belonging to the stable set corresponds to a subset of vertices of $G$. Let us pick an arbitrary vertex $v_S$ from each $S$. Consider any pair of vertices $S$ and $S'$ of the stable set. $v_S$ and $v_{S'}$ are necessarily not adjacent in $G$ since $S$ and $S'$ are not adjacent in $G^*$. This clearly implies that $G$ contains a stable set whose size is equal to the size of the stable set of $G^*$. ■
4.7 Conclusion

Metric and partition with others formulations are studied in this chapter. A polyhedral study of the convex hull of the feasible region of such integer formulations are reported in the next chapter with several families of facet-defining inequalities.
Chapter 5

Polyhedral study of $S_k(G)$

Contents

5.1 Introduction ................................................. 80
5.2 Some general properties ................................. 81
5.3 The path and cycle cases ............................... 83
5.4 Valid inequalities for $S_k(G)$ ............................ 86
   5.4.1 Hitting set inequalities .......................... 86
   5.4.2 Connectivity inequalities ....................... 87
   5.4.3 Cycle inequalities ................................. 88
   5.4.4 Wheel inequalities ................................. 90
   5.4.5 Antiweb inequalities ............................... 91
   5.4.6 Generalized projected metric inequalities ..... 93
5.5 Conclusion .................................................... 95

5.1 Introduction

This chapter is dedicated to a polyhedral study of $S_k(G)$. We will first present some properties of this polytope. Then we will focus on the path and cycle cases. Finally, several valid inequalities will be presented and studied.
5.2 Some general properties

Proposition 5.1 The $k$-separator polytope $S_k(G)$ is full-dimensional.

Proof: The incidence vectors of the following $k$-separators are affinely independent: $V$ and $V \setminus \{w\}, \forall w \in V$. ■

Proposition 5.2 If $k \geq 2$, the trivial inequalities $x_v \leq 1$ and $x_v \geq 0$, for all $v \in V$ are facet defining for $S_k(G)$.

Proof: Given $v \in V$, the incidence vectors of the following $k$-separators are affinely independent and all saturate the inequality $x_v \geq 0$: $V \setminus \{v\}$ and $V \setminus \{v, w\}, \forall w \in V \setminus \{v\}$.

Given $v \in V$, the incidence vectors of the following $k$-separators are affinely independent and all saturate the inequality $x_v \leq 1$: $V$ and $V \setminus \{w\}, \forall w \in V \setminus \{v\}$. ■

Proposition 5.3 If $a^t x \geq \alpha$ denotes a facet defining inequality for $S_k(G)$ different from the trivial inequalities (i.e. $0 \leq x_v \leq 1$, for all $v \in V$) then necessarily $a_v \geq 0, \forall v \in V$ and $\alpha > 0$.

Proof:

[Necessity of the condition $a_v \geq 0, \forall v \in V$]. Assume there exists some node $w \in V$ with $a_w < 0$. As the inequality $a^t x \geq \alpha$ is facet defining and different from $-x_w \geq -1$, there exists a $k$-separator $Z \subseteq V \setminus \{w\}$ whose incidence vector saturates the constraint. But adding the node $w$ to $Z$ we still get a $k$-separator but its incidence vector violates the inequality, hence a contradiction with the validity of the constraint.

[Necessity of the condition $\alpha > 0$]. From the former we have $a_v \geq 0, \forall v \in V$. Since all the vectors in the $k$-separator polytope satisfy $x_v \geq 0$ and $a^t x \geq \alpha$ is facet defining, different from $x_v \geq 0$, it follows that necessarily $\alpha > 0$. ■

Proposition 5.4 The support of any facet of $S_k(G)$ necessarily corresponds to a connected component of $G$ having at least $k + 1$ nodes.

Proof: Let $a^t x \geq \alpha$ denote a facet-defining inequality for $S_k(G)$. From Proposition 5.3 $\alpha > 0$. This implies that the subgraph $G_\alpha$ of $G$ which is induced by the set
of nodes \( V_a \) corresponding to the support of the vector \( a \) contains a connected component with size at least \( k+1 \). (For, if \( G_a \) would not contain a component with size at least \( k+1 \), considering the set of nodes \( V \setminus V_a \) it corresponds to a \( k \)-separator, thus a contradiction with the validity of the inequality).

Assume that the subgraph \( G_a \) contains a component \( C \) with size at most \( k \). Then any \( \k \)-separator \( S \) whose incidence vector saturates the constraint \( a^t x \geq \alpha \) should verify \( v_w = 0, \forall w \in C \). So the face defined by the inequality \( a^t x \geq \alpha \) would be contained in faces defined by trivial inequalities: a contradiction.

Finally assume that \( G_a \) contains two components \( C_1, C_2 \) with size at least \( k+1 \). Let \( S \) and \( S' \) denote two \( \k \)-separators whose incidence vectors saturate the inequality \( a^t x \geq \alpha \), with \( \sum_{v \in S \cap C_1} a_v = k_1 \) and \( \sum_{v \in S' \cap C_1} a_v = k_1' \). Assuming \( k_1 > k_1' \) (the case \( k_1 < k_1' \) can be treated similarly) we have \( \sum_{v \in S \setminus C_1} a_v < \sum_{v \in S' \setminus C_1} a_v \). Now consider the node set \( W = (S \setminus C_1) \cup (S' \cap C_1) \cup (V \setminus V_a) \). \( W \) is a \( \k \)-separator satisfying \( \sum_{v \in W} a_v < \sum_{v \in S} a_v \), thus contradicting the validity of the inequality \( a^t x \geq \alpha \).

It follows that the inequality \( a^t x \geq \alpha \) is redundant with respect to inequalities associated with each connected component of \( G_a \) (which are of the form \( \sum_{v \in C} a_v x_v \geq k_c \), where \( k_c \) denotes the value \( \sum_{v \in S \cap C} a_v \), with \( S \) as defined above). \( \blacksquare \)

The following proposition characterizes when a facet defining inequality for \( S_k(G) \) is also facet defining for \( S_k(G') \), where \( G' \) is obtained from \( G \) by adding a vertex (and a set of edges between this additional vertex and vertices in \( G \)).

**Proposition 5.5** Let \( a^t x \geq b \) define a facet of the \( \k \)-separator polytope \( S_k(G) \) with \( G = (V, E) \). Let \( G' = (V' = V \cup \{v\}, E') \) denote a graph obtained from \( G \) by adding a node \( v \) and some edges of the form \( vw, w \in V \). Then the inequality \( a^t x \geq b \) defines a facet of \( S_k(G') \) iff there exists a \( \k \)-separator \( S \subseteq V \) in \( G' \) whose incidence vector \( \chi(S) \in \mathbb{R}^{|V|} \) satisfies \( a^t \chi(S) = b \).

**Proof:** Notice firstly that if \( a^t x \geq b \) defines a facet of the \( \k \)-separator polytope \( S_k(G) \) then it is valid for \( S_k(G') \). (For, if \( S \subseteq V' \) is a \( \k \)-separator in \( G' \) then \( S \cap V \) is a \( \k \)-separator in \( G \)).

[\( \Rightarrow \)] By contradiction. Assuming such a \( \k \)-separator \( S \subseteq V \) does not exist, then the face of \( S_k(G') \) that is defined by \( a^t x \geq b \) is contained in the one defined by \( x_v \leq 1 \) and hence it cannot define a facet of \( S_k(G') \).

[\( \Leftarrow \)] Given a set of \( |V| \) \( \k \)-separators \( S_1, \ldots, S_{|V|} \) in \( G \) whose incidence vectors (in \( \mathbb{R}^{|V|} \)) are affinely independent the sets \( (S_i')_{i=1}^{|V|} \) with \( S'_i = S_i \cup \{v\}, \forall i \in \{1, \ldots, |V|\} \)
are \( k \)-separators in \( G' \). And adding to this latter set of vectors the incidence vector \( \chi(S_q) \in \mathbb{R}^{|V'|} \), for some arbitrarily chosen index \( q \in \{1, \ldots, |V|\} \), we get a set of \( |V'| \) incidence vectors of \( k \)-separators in \( G' \) that are affinely independent. ■

**Proposition 5.6** Let \( a'x \geq b \), \( x \in \mathbb{R}^{|V|} \) define a facet \( F \) for \( S_k(G) \). Given \( x \in \mathbb{R}^{|V|} \), let \( x' \in \mathbb{R}^{|V'|} \) denote the restriction of \( x \) to the entries corresponding to nodes in the node subset \( V' = V \setminus \{v\} \), for some node \( v \in V \) such that \( a_v = 0 \). Then the set \( F' = \{x' \in S_k(G') : x \in F\} \) contains \( |V'| \) affinely independent incidence vectors of \( k \)-separators in \( G' \): the subgraph of \( G \) that is induced by the node set \( V' \).

**Proof:** Let \( A \in \{0,1\}^{m \times |V|} \) be a matrix whose rows correspond to all the incidence vectors of \( k \)-separators in \( G \) that are in \( F = \{x \in S_k(G) : a'x = b\} \). Since \( F \) is facet defining for \( S_k(G) \) and \( S_k(G) \) is full-dimensional, \( A \) contains \(|V|\) affinely independent rows. Let \( A' \) be obtained from \( A \) by dropping the column corresponding to node \( v \) and assume \( A' \) contains at most \(|V| - 2\) affinely independent rows. This implies that \( F' = \{x \in S_k(G') : a'x = b\} \) is not a facet of \( S_k(G') \). Thus, there exists an inequality \( c'x \geq d \) that is valid for \( S_k(G') \) and such that \( F' \subseteq F'' = \{x \in S_k(G') : c'x = d\} \) with \((c, d)\) that is not a scalar multiple of \((a, b)\) (i.e. there does not exist \( \alpha > 0 \) with \((c, d) = \alpha(a, b)\)). Notice that since \( c'x \geq d \) is valid for \( S_k(G) \), \( F' \subseteq F'' \), we have \( F \subseteq \{x \in S_k(G) : c'x = d\} \). This namely implies \( c'x \geq d \) is facet defining for \( S_k(G) \), so that we must have \((c, d) = \alpha(a, b)\) for some scalar \( \alpha > 0 \), a contradiction. ■

**Corollary 5.1** Let the inequality \( a'x \geq b \) be facet inducing for \( S_k(G) \), \( G = (V, E) \). Then it is also facet defining for \( S_k(G') \) where \( G' = (V', E') \) denotes the subgraph of \( G \) that is induced by the node set \( V' \subseteq V \) and such that \( a_v = 0, \forall v \in V \setminus V' \).

**Proof:** Iterative application of Proposition 5.6, removing nodes of \( G \) that are not in \( V' \). ■

### 5.3 The path and cycle cases

Let us assume that \( G = (V, E) \) is a path where \( V = \{v_1, v_2, \ldots, v_n\} \) and \( E = \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\} \). The connected components of size \( k + 1 \) considered in \( LP1 \) are denoted by \( S_1, \ldots, S_1, \ldots S_{n-k} \), where \( S_i = \{v_i, v_{i+1}, \ldots, v_{i+k}\} \). The constraints of \( LP1 \) related to large connected components can be written in the matrix form \( Ax \geq 1 \), where the \( i^{th} \) row of \( A \) corresponds to the incidence vector of \( S_i \).
5.3. The path and cycle cases

**Proposition 5.7** If $G$ is a path, then formulation $LP_1$ is exact (the extreme solutions of $LP_1$ are optimal $k$-separators).

**Proof:** We can write $LP_1$ as $\{\min wx : Ax \geq 1, 0 \leq x_v \leq 1, v \in V\}$. Considering the matrix $B$ where the first $n-k$ rows correspond to matrix $A$, the next $n$ rows correspond to the identity matrix of dimension $n$ and the last $n$ rows correspond to the opposite of the identity matrix. It is clear that all constraints of $LP_1$ can be summarized in the form $Bx \geq b$ where $b$ is an integer vector. Using the fact that interval matrices are totally unimodular, we deduce that $B$ is also totally unimodular \[4, 76\]. This terminates the proof. ■

Knowing that the problem can be solved in polynomial-time for trees, one may look for a polyhedral description in this case. In fact, inequalities \((4.1)\) considered in \((IP1)\) can be separated in polynomial time using the algorithm proposed in \[63\] to find a maximum-weight connected subgraph of a given size when the graph is a tree. However, these inequalities are generally not sufficient to describe the convex hull of the incidence vectors of $k$-separators. A complete description of $S_k(G)$ in the tree case is still an open question.

Let us now assume that $G = (V, E)$ is a cycle where $V = \{v_1, v_2, ..., v_n\}$ and $E = \{(v_1, v_2), ..., (v_i, v_{i+1}), ..., (v_{n-1}, v_n), (v_n, v_1)\}$.

It is clear that if we know that $x_{v_i} = 1$ for some vertex $v_i$ (i.e., $v_i$ belongs to the $k$-separator), then any $k$-separator of the cycle should contain a $k$-separator of the path $V \setminus v_i$. Using Proposition 5.7, we deduce that a minimum-weight $k$-separator containing vertex $v_i$ can be computed by solving the following linear program $LP_i$.

\[
LP_i \begin{cases} 
\min \sum_{v \in V} w_v x_v \\
\text{s.t.:} \\
\sum_{v \in S} x_v \geq 1 \ \forall |S| = k + 1, \ G(S) \text{ connected and } v_{i_0} \notin S \\
0 \leq x_v \leq 1 \ \forall v \in V \\
x_{v_i} = 1
\end{cases}
\]

Let $P_i$ be the polytope corresponding to the feasible region of the formulation $LP_i$. Then, $P_i = \text{conv}\{\chi(S) \in \{0, 1\}^n, S \text{ is a } k\text{-separator containing } v_i\}$. Let $T$ be an arbitrary subset of vertices such that $|T| = k + 1$ and $G(T)$ is connected. Since at least one vertex of $T$ belongs to the $k$-separator, we can write that $S_k(G) = \text{conv}\{\bigcup_{v_i \in T} P_i\}$.
Chapter 5. Polyhedral study of $S_k(G)$

Using the projection result of Balas [5], we get the following equivalent formulation for the k-vertex separator problem when the graph is a cycle:

$$\begin{align*}
\min & \sum_{v \in V} w_v x_v \\
\text{s.t.} & \quad x = \sum_{i \in T} z_i^j \\
& \quad 0 \leq z_j^i \leq z_i^i \quad \forall v_i \in T, v_j \in V \\
& \quad \sum_{v_j \in S} z_j^i \geq z_i^i \quad \forall |S| = k + 1, G(S) \text{ connected, and } v_i \in T \setminus S \\
& \quad \sum_{v_i \in T} z_i^i = 1
\end{align*}$$

In the formulation above, $z_i^j$ is a vector of dimension $n$ whose components are given by $z_j^i$ for $v_j \in V$.

Let us now focus on some special cases. Let $C_k$ denote a cycle with length $k$.

**Proposition 5.8** $S_k(C_{k+1}) = \{ x \in [0,1]^{k+1} : \sum_i x_i \geq 1 \}$

**Proof:** Constraint matrix is TU. ■

The formulations for $S_k(C_{k+1})$ and $S_k(P_{k+1})$ are the same (where $P_{k+1}$ stands for an elementary path obtained from $C_{k+1}$ by removing an edge).

**Proposition 5.9** $S_k(C_{k+2}) = \{ x \in [0,1]^{k+2} : \sum_i x_i \geq 2 \}$

**Proof:** Constraint matrix is TU. ■

Note that differently from the case of $C_{k+1}$, the formulations of $S_k(C_{k+2})$ and $S_k(P_{k+2})$ do not coincide (since the constraint $\sum_i x_i \geq 2$ is not valid for $S_k(P_{k+2})$).

For the particular case of $C_5$ and $k = 2$ the following proposition shows that the addition of the constraints on all paths with length $k + 1$ provides an exact formulation.

**Proposition 5.10** $S_2(C_5) = \{ x \in [0,1]^{k+1} : \sum_i x_i \geq 2, \sum_{i \in p} x_i \geq 1, \forall p \in P_3 \}$, where $P_3$ stands for the set of all the paths on $C_5$ with length 3.

**Proof:** Let $a^t x \geq \alpha$ denote a facet defining inequality for $S_2(C_5)$ which is different from the trivial inequalities. From Proposition 5.3, we have $a_v \geq 0, \forall v \in V$ and $\alpha > 0$. We consider two cases.

**Case 1:** there exists a node $w \in V$ with $a_w = 0$. From Corollary 5.1 the inequality $a^t x \geq \alpha$ must be facet defining for $S_2(P_4)$ where $P_4$ stands for the graph.
5.4 Valid inequalities for $S_k(G)$

5.4.1 Hitting set inequalities

Hitting set inequalities are the basic inequalities (4.1).

**Proposition 5.11** Let $S$ be a subset of vertices such that $G(S)$ is connected and $|S| = k + 1$. Then the inequality $\sum_{v \in S} x_v \geq 1$ defines a facet of $S_k(G)$ if each vertex $w \in V \setminus S$ is adjacent to at most 1 vertex in $S$.

**Proof:** Let us build $n$ affinely independent vectors related to $k$-separators and saturating inequality (4.1). For each vertex $w \in \overline{S}$, we consider a $k$-separator incidence vector where $x_w = 0$, $x_v = 1$ for each $v \in \overline{S}$ ($v \neq w$), and $x_v = 0$ for each $v \in S$ except for one vertex $v$ that is a neighbor of $w$ (if $w$ does not have neighbors, the vertex $v$ such that $x_v = 1$ is chosen arbitrarily in $S$). In this way, we obtain $n - |S|$ vectors saturating (4.1). The remaining $|S|$ vectors are built as follows. For each vertex $v \in S$, we consider the vector where $x_v = 1$, $x_w = 0$ for $w \in S \setminus \{v\}$, and $x_w = 1$ for any $w \in \overline{S}$.

It is now easy to see that the $n$ vectors correspond to $k$-separators and are affinely independent. First, for each $v \in \overline{S}$, there is only one vector such that $x_v = 0$. Second, among the last $|S|$ vectors, for each $v \in S$ there is only one vector such that $x_v = 1$. These two observations immediately lead to the affine independence of the $n$ vectors. ■
Chapter 5. Polyhedral study of $S_k(G)$

If $G$ is a tree and $S$ is a subtree of $G$, then each vertex $v \in S$ has at most one neighbor in $G$. This leads to the following obvious corollary.

Corollary 5.2 If $G$ is a tree, then each inequality (4.1) induces a facet of $S_k(G)$.

5.4.2 Connectivity inequalities

Proposition 5.12 Let $S \subset V$ be a subset of vertices such that $|S| \geq k + q$, $q \geq 1$ and $G(S)$ is $q$-node-connected. Then the following inequality is valid for $S_k(G)$:

$$\sum_{v \in S} x_v \geq q$$  \hfill (5.1)

Proof: It is clear that by removing less than $k$ vertices from $S$, the remaining subgraph is still connected and it contains at least $k + 1$ vertices. This immediately implies that $\sum_{v \in S} x_v \geq q$ is valid. ■

Notice that inequalities (5.1) were also considered in [53] but in a more restricted form (it is required in [53] that $|S| = k + q$).

Let us focus on the special case where $|S| = k + q$. The next result can be seen as a generalization of Proposition 5.11 where $q$ was equal to 1.

Proposition 5.13 Let $S \subset V$ be a subset of vertices such that $|S| = k + q$, $q \geq 1$ and $G(S)$ is $q$-node-connected. Then inequality (5.1) induces a facet of $S_k(G)$ if each vertex $w \in V \setminus S$ is adjacent to at most $q$ vertices in $S$.

Proof: The proof is very similar to the proof of Proposition 5.11. We build $n$ affinely independent vectors related to $k$-separators and saturating inequality (5.1). For each vertex $w \in S$, we consider a $k$-separator incidence vector defined as follows. We have $x_w = 0$ and $x_v = 1$ for each $v \in S$ ($v \neq w$). We select a subset of vertices $S_w \subset S$ of size $q$ containing all neighbors of $w$ in $S$. Then $x_v = 1$ for any $v \in S_w$ and $x_v = 0$ for $v \in S \setminus S_w$. In this way, we obtain $n - |S|$ vectors saturating (5.1). The remaining $|S|$ vectors are built as follows. Observe that $\sum_{v \in S} x_v \geq q$ induces a facet of $S_k(G(S))$. This is easy to check: assume that all $k$-separators saturating inequality (5.1) satisfy the equality $\sum_{v \in S} \alpha_v x_v = \beta$, then by considering two $k$-separators containing the same subset of vertices of size $q - 1$ and differing in only one vertex, we show that $\alpha_v = \alpha_v'$ for any vertices $v, v'$ of $S$. This implies that inequality (5.1) induces a facet of $S_k(G(S))$. Then, it is possible to find $|S|$ $k$-separators (of $G(S)$) whose incidence
vectors are affinely independent and contained in the face induced by (5.1). These
$k$-separators of $G(S)$ can be extended to $k$-separators of $G$ by taking $x_w = 1$ for any
$w \in \overline{S}$.
It is now easy to see that the $n$ vectors correspond to $k$-separators and are affinely
independent. ■

If we consider the more restrictive assumption: $G(S)$ is $q + 1$-node-connected,
then the condition related to the number of neighbors in $S$ becomes necessary and
sufficient to obtain a facet.

Proposition 5.14 Let $S \subset V$ be a subset of vertices such that $|S| = k + q, q \geq 1$
and $G(S)$ is $q + 1$-node-connected. Then inequality (5.1) induces a facet of $S_k(G)$ if
and only if each vertex $w \in V \setminus S$ is adjacent to at most $q$ vertices in $S$.

Proof: If each vertex $w \in V \setminus S$ is adjacent to at most $q$ vertices in $S$, we know
from the previous proposition that (5.1) induces a facet of $S_k(G)$. Let us now prove
that this condition is necessary. Suppose that there is a vertex $w \in V \setminus S$ adjacent
to at least $q + 1$ vertices in $S$. By removing any subset of nodes of size $q$ from
$S$, we still obtain a connected component of size $k$ (by $q + 1$-node-connectivity of
$G(S)$). The vertex $w$ has necessarily at least one neighbor in the remaining part
of $S$. This implies that whenever $\sum_{v \in S} x_v = q$, we should have $x_w = 1$. In other
words, inequality (5.1) does not induce a facet of $S_k(G)$. ■

Notice that the case where $G(S)$ is a clique was considered in [53]. Then the
previous proposition can be seen as a generalization of the clique case.

5.4.3 Cycle inequalities

Proposition 5.15 Let $S$ be a subset of vertices such that $|S| \geq k + 1$ and $G(S)$ is
an elementary cycle, then the following inequality is valid for $S_k(G)$:

$$\sum_{v \in S} x_v \geq \left\lceil \frac{|S|}{k+1} \right\rceil.$$  (5.2)

Proof: By writing inequality (4.1) for each subset $S' \subset S$ for which $G(S')$ is
connected and adding up all of them, we obtain the inequality $(k+1) \sum_{v \in S} x_v \geq |S|.$
Inequality (5.2) follows by simple rounding. ■

Notice that when $|S| = k + 1$, Proposition 5.14 can be applied with $q = 1$ to
know under which conditions inequality (5.2) induces a facet. Let us then focus on
Chapter 5. Polyhedral study of $S_k(G)$

the case where $|S| > k + 1$. It is clear that if $|S| \equiv 0[k+1]$, then (5.2) is just the sum of inequalities of type (4.1).

**Proposition 5.16** Let $S$ be a subset of vertices such that $|S| > k + 1$, $|S|$ is not a multiple of $k + 1$, and $G(S)$ is an elementary cycle, then inequality (5.2) induces a facet of $S_k(G)$ if and only if for each vertex $w \in \overline{S}$, there is a $k$-separator of size $\lceil \frac{|S|}{k+1} \rceil$ in $G(S \cup \{w\})$.

**Proof:** The existence of a $k$-separator of size $\lceil \frac{|S|}{k+1} \rceil$ in $G(S \cup \{w\})$ is equivalent to say that there are $k$-separators saturating (5.2) and not containing $w$. If there is a vertex $w \in \overline{S}$ for which there are no such $k$-separators, then equality $\sum_{v \in S} x_v = \lceil \frac{|S|}{k+1} \rceil$ implies that $x_w = 1$. We deduce that the existence of such $k$-separators is a necessary condition to get a facet.

Let us now assume that the condition is satisfied and let us build $n$ affinely independent vectors related to $k$-separators and saturating inequality (5.2).

For each vertex $w \in \overline{S}$, we consider a $k$-separator incidence vector defined as follows. We have $x_w = 0$ and $x_v = 1$ for each $v \in \overline{S}$ ($v \neq w$). We select a subset of vertices $S_w \subset S$ of size $\lceil \frac{|S|}{k+1} \rceil$ corresponding with a $k$-separator of $G(S \cup \{w\})$. Then $x_v = 1$ for any $v \in S_w$ and $x_v = 0$ for $v \in S \setminus S_w$. In this way, we obtain $n - |S|$ vectors saturating (5.2).

To build the remaining $|S|$ vectors, we should first prove that $\sum_{v \in S} x_v \geq \lceil \frac{|S|}{k+1} \rceil$ induces a facet of $S_k(G(S))$. Assume that all $k$-separators saturating inequality (5.2) satisfy the equality $\sum_{v \in S} v_v = \beta$. Given any $k$-separator of $G(S)$ of size $\lceil \frac{|S|}{k+1} \rceil$, there is at least one vertex $v$ belonging to the $k$-separator such that the next vertex $v'$ belonging to the $k$-separator (when we go through the cycle in the clockwise direction) is situated at a distance less than or equal to $k - 1$. In other words, there is a subset $S' \subset S$ of size less than or equal to $k$, containing $v$ and $v'$ where both $v$ and $v'$ belong to the $k$-separator. This is true because $|S|$ is not a multiple of $k + 1$.

It is now clear that if we replace $v$ by the vertex $v''$ preceding $v$ in the cycle (in clockwise direction), we still obtain a $k$-separator of $G(S)$ of size $\lceil \frac{|S|}{k+1} \rceil$. By writing that both the initial and the modified $k$-separators satisfy equality $\sum_{v \in S} v_v = \beta$, we get that $v_v = v_v''$ where $v$ and $v''$ are adjacent on the cycle. This obviously implies that all coefficients $v_v$ are equal. This is enough to say that inequality (5.2) induces a facet of $S_k(G(S))$.

We are now ready to build the remaining $|S|$ $k$-separators. First, we consider $|S|$
5.4. Valid inequalities for $S_k(G)$

$k$-separators (of $G(S)$) whose incidence vectors are affinely independent and contained in the face induced by (5.2). These $k$-separators of $G(S)$ can be extended to $k$-separators of $G$ by taking $x_w = 1$ for any $w \in \overline{S}$.

It is now easy to see that the $n$ vectors are affinely independent. ■

The existence of a $k$-separator of size $\lceil \frac{|S|}{k+1} \rceil$ in $G(S \cup \{w\})$ mentioned in Proposition 5.16 does not appear to be very explicit. While it is possible to find an explicit equivalent condition related to how the neighbors of a $w \in G \setminus S$ are located in $S$, we will only give a sufficient condition (to keep the size of the paper under control).

Let $v_{w}^1, \ldots, v_{w}^r$ be the neighbors of $w$ in $S$. We assume that they are encountered in the order $v_{w}^1, \ldots, v_{w}^r$ when one goes through the cycle in the clockwise sense. Let $h(v_i^w, v_{i+1}^w)$ be the number of vertices located between $v_i^w$ and $v_{i+1}^w$ (when going from $v_i^w$ to $v_{i+1}^w$ in the same sense and not counting $v_i^w$ and $v_{i+1}^w$). We will consider that $v_{i+1}^w = v_i^w$.

**Proposition 5.17** Let $S$ be a subset of vertices such that $|S| > k + 1$, $|S|$ is not a multiple of $k + 1$, and $G(S)$ is an elementary cycle, then inequality (5.2) induces a facet of $S_k(G)$ if for each vertex $w \in \overline{S}$ having $r$ adjacent vertices $v_{w}^1, \ldots, v_{w}^r$ in $S$, the following equality holds:

$$r + \sum_{i=1}^{r} \left\lfloor \frac{h(v_i^w, v_{i+1}^w)}{k+1} \right\rfloor = \left\lceil \frac{|S|}{k+1} \right\rceil.$$

**Proof**: Observe that if $w$ does not have neighbors in $S$, then the existence of a $k$-separator of size $\lceil \frac{|S|}{k+1} \rceil$ in $G(S \cup \{w\})$ is obviously guaranteed. Assume that $w$ has $r$ neighbors in $S$. Then $r + \sum_{i=1}^{r} \left\lfloor \frac{h(v_i^w, v_{i+1}^w)}{k+1} \right\rfloor$ represents the size of a $k$-separator including the $r$ neighbors in addition to a minimum number of vertices that must be removed to disconnect the connected components of size $k + 1$ located between $v_i^w$ and $v_{i+1}^w$ ($1 \leq i \leq r$). If $r + \sum_{i=1}^{r} \left\lfloor \frac{h(v_i^w, v_{i+1}^w)}{k+1} \right\rfloor = \lceil \frac{|S|}{k+1} \rceil$, then this $k$-separator has a minimum size. ■

5.4.4 Wheel inequalities

Let $W$ denote a wheel (contained in $G$) having for rim the cycle $C = (V_C, E_C)$ and hub $v_0$ (i.e., $W$ is the subgraph of $G$ whose node set is $V_C \cup \{v_0\}$ and edge set is
Chapter 5. Polyhedral study of $S_k(G)$

$E_C \cup (v_0, r, \forall r \in V_C)$. The wheel inequality is defined below.

$$\sum_{v \in V_C} x_v + (|V_C| - \left\lfloor \frac{|V_C|}{k+1} \right\rfloor - k + 1)x_{v_0} \geq |V_C| - k + 1.$$  \hspace{1cm} (5.3)

**Proposition 5.18** Inequality (5.3) is valid for $S_k(G)$. Whenever the cycle inequality (5.2) obtained with $S = V_C$ defines a facet of $S_k(G \setminus \{v_0\})$, inequality (5.3) induces a facet of $S_k(G)$.

**Proof:** Inequality (5.3) can be obtained by maximum lifting of (5.2). ■

The results of Section 5.4.3 can be directly used to get conditions under which inequality (5.3) induces a facet of $S_k(G)$.

### 5.4.5 Antiweb inequalities

Let $AW(r, q)$ with $r, q \in \mathbb{N}$, denote a graph (also called antiweb) consisting of a cycle $C$ with length $r$: $C = (v_1, v_2, \ldots, v_r)$ and all edges of the form $(v_i, v_j)$ if the distance between $v_i$ and $v_j$ on $C$ is at most $q$.

**Proposition 5.19** If $AW(r, q)$ with $r \geq k + q$ is a subgraph of $G$, then the following inequality is valid for $S_k(G)$

$$\sum_{v \in AW(r, q)} x_v \geq \frac{rq}{k + q}.$$ \hspace{1cm} (5.4)

**Proof:** Let $U$ denote the set of vertices in a $k$-separator of $G$. Let $T$ denote the nodes of $AW(r, q)$ that are not contained in $U$.

Consider first the case when the subgraph $G'$ of $G$ that is induced by $V \setminus U$ has $p$ distinct connected components $B_1, \ldots, B_p$ intersecting $AW(r, q)$, with $p \geq 2$.

Considering one component $B_i$, for some $i \in \{1, \ldots, p\}$, it is a simple observation (considering the nodes of $B_i$ in $C$ sequentially, for some arbitrarily fixed order on $C$) that at least one node, denoted $\overline{v}_i \in B_i \cap C$ satisfies the following:

- the first node which follows $\overline{v}_i$ in $C$ (for the chosen order on $C$) belongs to $U$, and

- the first node which follows $\overline{v}_i$ in $C$ (for the chosen order on $C$) and does not belong to $U$ belongs to a different connected component $B_j$, $j \in \{1, \ldots, p\}, j \neq i$.  

91
So the first $q$ nodes following $v_i$ on $C$ (for the chosen order) necessarily belong to $U$. And as this holds for each connected component $B_i, i \in \{1, \ldots, p\}$, we have $|U \cap C| \geq qp$.

And since each connected component has at most $k$ nodes, the number of nodes $r$ satisfies $r \leq kp + |U \cap C|$.

Combining the last two inequalities and rounding leads to $|U \cap C| \geq \frac{rq}{k+q}$.

Finally, for the case $p \leq 1$, using the inequalities $|U \cap C| \geq q, r \leq k + |U \cap C|$ and then rounding, leads to the proposition. ■

**Proposition 5.20** Let $G = AW(n, q)$ with $n = r(k+q) - z, 1 \leq z \leq \frac{k^2}{q}, q \geq 2,$ and $n, q$ relatively prime integers. Then the inequality (5.4) is facet-defining for $S_k(G)$.

**Proof:** Let $\sum_{v \in AW(n,q)} \alpha_v x_v \geq \beta$ be an inequality defining a facet $F$ of $S_k(G)$ and such that $F$ contains all the incidence vectors of the $k$-separators saturating inequality (5.4). We show that $\alpha_v = \alpha_w$ for all $v, w \in V$, thus implying that the inequality (5.4) is facet-defining for $S_k(G)$.

Let $U$ denote the node set of a $k$-separator saturating inequality (5.4). From the assumptions on $n$ and $z$ we deduce $|U| = rq$.

Let $B_i$ denote a connected component of the graph $G(V \setminus U)$ with cardinality at least 2. Then $B_i$ can be described by a sequence of nodes in $C$: $(v_1, \ldots, v_p)$ such that two consecutive nodes in the sequence are at distance at most $q$ on $C$ from each other. This namely implies that all the nodes in the cycle $C$ from $v_1$ to $v_p$ must belong to either $U$ or $B_i$ (i.e. they cannot belong to another connected component of $G(V \setminus U)$). From this observation it follows that if $G(V \setminus U)$ consists of $l$ connected components then the $k$-separator $|U|$ must contain at least $lq$ nodes. We mentioned above that for a separator saturating inequality (5.4) we have $|U| = rq$, thus implying that $G(V \setminus U)$ has at most $r$ connected components.

And from the assumption on $n$, it follows that $G(V \setminus U)$ must consist of (at least, and so) exactly $r$ connected components. This shows that $G(U)$ consists of $r$ paths contained in $C$, each having length $q$ (and consequently each connected component of the graph $G(V \setminus U)$ consists of a path contained in $C$ having length at most $k - 1$). Since the graph $G(V \setminus U)$ has $rk - z$ nodes and $r$ connected components, there exists at least one connected component with cardinality at most $k - 1$ and there is at most one connected component consisting of a single node.
Chapter 5. Polyhedral study of $S_k(G)$

Let $B_0, \ldots, B_{r-1}$ denote the connected components of $G(V \setminus U)$ that appear sequentially on $C$ (for some arbitrarily set order) and let $B_l$, $l \in \{0, \ldots, r-1\}$ denote one such component with minimum cardinality. Consider now the nodes $w_1$: the last node of $B_{l-1}$ (taking indices modulo $r$) and $w_2$: the node of $U$ that is located after $w_1$ on $C$ (for the chosen order) and is a neighbor of the first node of $B_l$. (So the $q$ nodes which follow $w_1$ on $C$ belong to $U$ and $w_2$ is the last one of these nodes in $U$). Then the node set $U' := U \setminus \{w_2\} \cup \{w_1\}$ is a $k$-separator with the same cardinality as $U$, and we can deduce $\alpha_{w_1} = \alpha_{w_2}$.

We may then proceed in this manner until the component $B_i$ of $G(V \setminus U)$ to which we added a node attains a cardinality of value $k$. We may then consider the component $B_{i-1}$ in place of $B_i$... and so on, leading to the equations $\alpha_{v_i} = \alpha_{v_i+q}$, with $C = (v_0, \ldots, v_{n-1})$, taking indices modulo $n$. Since $n$ and $q$ are relatively prime integers we deduce the equations $\alpha_v = \alpha_w, \forall v, w \in V$.

5.4.6 Generalized projected metric inequalities

Remember that the projected metric inequalities (4.2) are given by $(|S|+1-k)(1-x_i) \leq \sum_{j \in S} x(p_{ij} \setminus \{i\})$ where $i \in V$. They can be generalized as follows. Vertex $i$ is replaced by a subset of vertices $A \subset V$ such that $G(A)$ is connected and $|A| \leq k$. Let $S$ be a subset of vertices $A \subset V$ such that $S \cap A = \emptyset$ and $|S| + |A| \geq k + 1$. For each vertex $j \in S$, let $p_{Aj}$ be a path connecting $j$ to one vertex from $A$. The internal vertices of $p_{Aj}$ can be assumed to be in $A$. Similarly to Section 4.5, $p_{Aj} \setminus A$ denotes the path connecting $j$ to the last vertex of $p_{Aj}$ not belonging to $A$. It is then easy to see that the following inequalities are valid:

$$((|S| + |A| - k)(1 - x(A)) \leq \sum_{j \in S} x(p_{Aj} \setminus A).$$

Inequalities (5.5) can be written in a different way by making two observations. First, the paths $p_{Aj}$ for $j \in S$ should be shortest paths from $j$ to $A$ (with respect to vertex weights $(x_v)_{v \in V}$). Second, we can assume that each path $p_{Aj} \setminus A$ is included in $S$ since otherwise inequality (5.5) can be strengthened by deleting $j$ from $S$, adding to $S$ a vertex $l$ from $p_{Aj} \setminus A$ and replacing $p_{Aj}$ by the subpath of $p_{Aj}$ connecting $l$ to $A$. These two observations imply that we can assume that $\bigcup_{j \in S} p_{Aj}$ is in fact the disjoint union of some trees rooted at vertices in $A$. All vertices of each rooted tree
valid inequalities for $S_k(G)$

Figure 5.1: Illustration of inequality (5.6)

(except the root) belong to $S$. Figure 5.1 illustrates the situation.

Observe that in the sum $\sum_{j \in S} x(p_{A_j \setminus A})$, the variable $x_v$ related to vertex $v \in S$ appears as many times as the number of vertices in the subtree rooted at $v$. Let us use $d'_v$ to denote this number. Then inequality (5.5) can be written as follows:

$$\left(|S| + |A| - k\right)(1 - x(A)) \leq \sum_{v \in S} d'_v x_v$$  \hspace{1cm} (5.6)

where $A \subset V$ and $S \subset \overline{A}$ such that: $G(A)$ is connected, $|A| \leq k$, and $|S| + |A| \geq k + 1$.

In the situation depicted by Figure 5.1, inequality (5.6) can be written as follows:

$$\left(|S| + |A| - k\right)(1 - x_a - x_b - x_c) \leq (x_f + x_g + x_h + x_j + x_k) + 2x_d + 3(x_e + x_i).$$

If we consider the special case where $S \subset N(A)$, then inequality (5.6) becomes

$$\left(|S| + |A| - k\right)(1 - x(A)) \leq x(S).$$ \hspace{1cm} (5.7)

Since the exhibition of all cases where inequality (5.6) induces a facet requires some tedious proofs, we will only focus on a special case of inequality (5.7).

**Proposition 5.21** Let $A = \{i\}$ and $S \subset N(A)$ be such that $|S| \geq k + 1$, and $G(S \cup \{j\})$ does not contain a connected component of size greater than or equal to $k + 1$ for any $j \in S \cup \overline{A}$. Then inequality (5.7) induces a facet of $S_k(G)$.

**Proof:** Assume that all $k$-separators saturating inequality (5.7) satisfy the equality
Chapter 5. Polyhedral study of $S_k(G)$

Among these $k$-separator, we can select the separator defined by $S$. Since we assumed that $G(S \cup \{j\})$ does not contain a connected component of size greater than or equal to $k + 1$ for any $j \in S \cup \bar{A}$, we still obtain a $k$-separator if we eliminate from $S$ a vertex $j \in S \cup \bar{A}$. This clearly implies that $\alpha_j = 0$ for any $j \in S \cup \bar{A}$.

By considering the union of any subset of $S$ of size $|S| + |A| - k$ with $S \cup \bar{A}$ we still get a $k$-separator whose incidence vector satisfies (5.7) with equality. Since the choice of the subset of $S$ of size $|S| + |A| - k$ is arbitrary, we deduce that $\alpha_v = \alpha_w$ for any $v$ and $w$ belonging to $S$. In other words, equality $\sum_{v \in V} \alpha_v x_v = \beta$ can be written as $\alpha_S x(S) + x(A) = \beta$ (remember that we assumed that $|A| = 1$).

Considering again the $k$-separator $S$ leads to $\beta = 1$. Also by considering a $k$-separator given by the union of a subset of $S$ of size $|S| + |A| - k$ with $S \cup \bar{A}$, we deduce that $\beta = \alpha_S(|S| + |A| - k)$. Consequently, equality $\sum_{v \in V} \alpha_v x_v = \beta$ is proportional to $(|S| + |A| - k)(1 - x(A)) = x(S)$. ■

5.5 Conclusion

Several classes of valid inequalities have been investigated, along with conditions under which some of them are facet defining for the $k$-separator polytope. The next chapter is dedicated to computational results of some formulations studied in this chapter and the previous one.
Chapter 6

Computational Results

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>96</td>
</tr>
<tr>
<td>6.2</td>
<td>Cutting-plane algorithms</td>
<td>97</td>
</tr>
<tr>
<td>6.2.1</td>
<td>A cutting-plane algorithm related to the stable set formulation $IP_2$</td>
<td>97</td>
</tr>
<tr>
<td>6.2.2</td>
<td>A cutting-plane algorithm related to the partitioning formulation $IP_9$</td>
<td>97</td>
</tr>
<tr>
<td>6.2.3</td>
<td>A cutting-plane algorithm related to formulations $IP_1$ and $IP_7$</td>
<td>98</td>
</tr>
<tr>
<td>6.3</td>
<td>Experimental results</td>
<td>99</td>
</tr>
<tr>
<td>6.4</td>
<td>Conclusion</td>
<td>105</td>
</tr>
</tbody>
</table>

6.1 Introduction

In this chapter many cutting-plane algorithms based on branch-and-cut and branch-and-bound are implemented. The first algorithm is related to the stable set formulation $IP_2$ by using a branch-and-bound method. The second algorithm is based on the partitioning formulation $IP_9$ using a branch-and-cut approach. The last implemented algorithm is related to formulations $IP_1$ and $IP_7$ using a branch-and-cut concept. A depth analysis of the results is also included in this chapter.
6.2 Cutting-plane algorithms

Three cutting-plane algorithms will be compared. The first one is related to the stable set formulation $\text{IP}_2$. The second one is related to the partitioning formulation $\text{IP}_9$. The third cutting plane algorithm is related to both basic and projected metric formulations ($\text{IP}_1$ and $\text{IP}_7$). Thanks to Proposition 4.2, we do not need to consider formulations $\text{IP}_4$, $\text{IP}_5$ and $\text{IP}_6$. Moreover, many valid inequalities including (4.1) and those of Section 5.4 can be used to strengthen the linear relaxation $\text{LP}_7$.

6.2.1 A cutting-plane algorithm related to the stable set formulation $\text{IP}_2$

The linear relaxation $\text{LP}_2$ can naturally be strengthened using some of the valid inequalities of the stable set polytope including odd-cycle inequalities, clique inequalities, etc. (see, e.g., [76]). Gerards and Schrijver [3] gave a polynomial-time separation algorithm for odd-cycle inequalities. In our implementation, only odd-cycle inequalities and clique inequalities are considered. Clique inequalities are separated using a basic greedy algorithm.

More valid inequalities could be considered. However, the size of formulation $\text{IP}_2$ becomes huge when $k$ increases. Then we can not expect a cutting-plane algorithm related to $\text{IP}_2$ to be competitive with the two next cutting-plane algorithms.

6.2.2 A cutting-plane algorithm related to the partitioning formulation $\text{IP}_9$

We consider the linear relaxation $\text{LP}_9$. Only inequalities (4.3) are iteratively added to improve the relaxation $\text{LP}_9$. The separation of these inequalities can obviously be done in polynomial time. For each edge $(i, j)$, we should take $U = \{b \in \{1, \ldots, B\} : z_{ib} \geq z_{jb}\}$. Then we only have to check if the inequality is violated.

Proposition 6.1 Inequalities (4.3) can be separated in polynomial time.

We have used a heuristic method to compute $B$. We solve $\text{LP}_2$ and then we round the max stable set (obj) value to the ceiling one($\lceil \text{obj} \rceil$).
6.2. Cutting-plane algorithms

6.2.3 A cutting-plane algorithm related to formulations \( IP_1 \) and \( IP_7 \)

We implemented a cutting-plane algorithm based on inequalities (4.1) and (4.2) in addition to some of the valid inequalities presented in chapter 5 (see section 5.4).

Separation of the hitting-set inequalities (4.1) is NP-hard even if all vertex-weights belong to \( \{0,1\} \) when \( k \) is part of the input [63]. If \( k \) is constant, the separation is obviously easy. It is also known that a maximum-weight connected subgraph of size \( k + 1 \) can be computed easily if the graph is a tree [63]. In the general case, we use a simple algorithm to separate inequalities (4.1) by building a connected component of size \( k \) in a greedy way: add to the component the neighbor having the largest \( x \)-value until the size of the component reaches \( k \). If the weight of the component is less than 1, we add the corresponding inequality.

Inequalities (4.2) can be separated in polynomial time as shown below.

**Proposition 6.2** There exists a polynomial-time algorithm to separate inequalities (4.2)

**Proof:** We give here the algorithm. The subset \( S \) is initially empty. For each vertex \( i \in V \), we first compute the shortest path \( p_{ij} \) for each \( j \neq i \). Then, we put in \( S \) the \( k \) closest vertices to \( i \). We also add to \( S \) all vertices \( j \ni S \) for which \( x(p_{ij} \setminus \{i\}) < (1 - x_i) \). If \( (|S| + 1 - k)(1 - x_i) > \sum_{j \in S} x(p_{ij} \setminus \{i\}) \), then we add the violated inequalities. The procedure is repeated for each vertex \( i \). The complexity of the algorithm is obviously polynomial. ■

Observe that inequalities (5.6) (equivalent to (5.5)) can also be separated in polynomial-time if the size of \( A \) is bounded by a constant. This happens for example if \( k \) is bounded by a constant. The separation algorithm is similar to the one presented above to separate inequalities (4.2). We only need to enumerate all subsets \( A \) of size at most \( k \) for which \( G(A) \) is connected.

**Corollary 6.1** Inequalities (5.6) can be separated in polynomial-time if \( k \) is upper-bounded by a constant.
6.3 Experimental results

We present numerical experiments obtained using many integer programs and instances. First integer program used is $IP_2$. Remember that $(IP_2)$ is based on a stable set formulation in an extended graph. Then all valid inequalities for the stable set problem can be used to strengthen the linear relaxation of $(IP_2)$. However, in our implementation we only focused on odd-cycle inequalities and clique inequalities. Odd-cycle inequalities are separated in polynomial time using the algorithm by [3] (see also 2.3.4), while clique inequalities are generated using a basic greedy heuristic. After a cutting-plane phase based on these two families of valid inequalities, a branch-and-bound follows using the default parameters of the Cplex solver. Results are reported in the Table 6.1. Graphs are generated randomly with different densities, orders (number of vertices) and costs are also uniformly distributed in the interval $[0, 100]$ for the tables, Table 6.1 and Table 6.2. For each instance in table 6.1 and table 6.2, we provide the following information: Name is the name of instance, the number of vertices is $|V|$, the number of edges denoted by $|E|$, 

| No.   | $|V|$ | $|E|$ | $k$ | Time  | Iter | Cuts | O.LP  | O.IP  | Gap  |
|-------|------|------|-----|-------|------|------|-------|-------|------|
| RanG1 | 10   | 12   | 2   | 00:03 | 3    | 7    | 106   | 107   | 1    |
| RanG2 | 10   | 6    | 2   | 00:03 | 2    | 2    | 74    | 74    | 0    |
| RanG3 | 20   | 39   | 2   | 00:07 | 4    | 32   | 273   | 349   | 22   |
| RanG4 | 20   | 20   | 2   | 00:03 | 3    | 8    | 203   | 203   | 0    |
| RanG5 | 30   | 23   | 2   | 00:02 | 3    | 13   | 200,33| 230   | 13   |
| RanG6 | 40   | 40   | 2   | 00:07 | 5    | 44   | 303,1 | 429   | 29   |
| RanG7 | 50   | 62   | 2   | 00:16 | 4    | 72   | 408,07| 492   | 17   |
| RanG8 | 60   | 90   | 2   | 00:43 | 5    | 90   | 618,04| 743   | 17   |
| RanG9 | 80   | 159  | 2   | 03:40 | 5    | 139  | 822,95| 1239  | 34   |
| RanG10| 100  | 249  | 2   | 13:15 | 13   | 296  | 1043,42| 2150  | 51   |
| RanG11| 10   | 15   | 3   | 00:03 | 4    | 12   | 35    | 63    | 44   |
| RanG12| 20   | 39   | 3   | 00:51 | 2    | 11   | 93,65 | 274   | 66   |
| RanG13| 30   | 23   | 3   | 00:05 | 3    | 23   | 63,06 | 143   | 56   |
| RanG14| 40   | 79   | 3   | 00:19 | 3    | 54   | 486,2 | 734   | 34   |
| RanG15| 50   | 62   | 3   | 03:50 | 4    | 63   | 186,1 | 420   | 56   |
| RanG16| 100  | 249  | 3   | 00:00 | 67   | 827  | 432,24| 1724  | 75   |
| RanG17| 10   | 10   | 4   | 00:04 | 3    | 7    | 29,8  | 44    | 32   |
| RanG18| 20   | 20   | 4   | 00:40 | 5    | 10   | 46    | 90    | 49   |
| RanG19| 40   | 40   | 5   | 04:10 | 4    | 34   | 9,14  | 165   | 94   |

Table 6.1: Compact Formulation $(IP_2)$ applied to random graphs
the maximum size of each component is $k$, $\text{Time}$ is the total time in minutes and seconds spent in the cutting-plane and the branch-and-bound or branch-and-cut phases, the $O.I.P$ is the best solution found before one hour. In table 6.1: the gap defined by 0 if the optimum value ($O.I.P$) found before one hour, otherwise \( \text{Gap} = (O.I.P - O.LP)/O.I.P \times 100 \), the cost $O.LP$ at the end of the cutting-plane phase, the number of generated odd cycle inequalities denoted by $Cuts$ and the number of iterations in the cutting-plane phase is $Iter$. Let $IP1$ and $IP7$ denote the formulations $IP1$ and $IP7$. In table 6.2 we have five different columns from Table 6.1 for the $IP1$ and $IP7$. The first is $\text{HitSet}$ corresponding to Hitting Set inequalities (4.1). The second is $\text{ProjMetric}$ for the Projected metric inequalities (4.2). The third, $B&C$ denotes the cuts added by Cplex solver. $N#Iter$ is the number of iterations one iteration consists of one round of adding violated inequalities then reoptimizing the $IP1$ or $IP9$ and $Nodes$ is the number of nodes in the branch-and-cut tree.

We implemented two branch-and-cut algorithms for $k$-separator problem applied to $IP1$ and $IP7$ (resp. $IP9$). In the cutting-phase we add in the first step the Cplex solver cuts in the $IP1$ and $IP7$ (resp. $IP9$), then we add the violated constraints for each class of the valid inequalities. Precisely, we add in second step the $\text{HitSet}$ to $IP1$ and $IP7$ (resp. $IP9$), where $\text{Compl}$ is the constraints (4.3)). In the last step we add the $\text{ProjMetric}$ to $IP1$ and $IP7$ and reoptimize $IP1$ and $IP7$ (resp. $IP9$) in each iteration. The separation of these inequalities are mentioned before in 6.2.3 (resp. 6.2.2). The branching phase of our two branch-and-cuts algorithms is done as follows: we branch on variable $x_i (i \in V)$ for which \( \{x_i, 1 - x_i\} \) is maximal. If $x_i < 0.5$ we first examine the branch corresponding to $x_i = 0$, if $x_i \geq 0.5$ we start with the case $x_i = 1$.

The program was written in C++. All experiments were conducted on a computer with a processor "Intel Core 2 Quad CPU Q6600" of frequency 2.4 Ghz, and a RAM size of 3.25 Gbytes running on Windows operating system.

While the problem is solved very quickly by using IP2, one can observe that the gap between the linear-programming bound at the end of the cutting-plane phase and the optimum can be very large even for problems of medium size. This suggests the necessity of adding other valid inequalities. Moreover, when $k$ increases, the size of the extended graph and thus the formulation $LP2$ increases very quickly. In other words, to solve problems with larger values of $k$ and $|V|$, it seems that we should try to strengthen formulation $LP1$ rather than $LP2$. We can observe that
Chapter 6. Computational Results

| Name   | $|V|$ | $|E|$ | $k$ | Time | $N$ $\#$.Iter | Nodes | HitSet | ProjMetric | $B$ & $C$ | $O.IP$ |
|--------|-----|-----|-----|------|--------------|-------|--------|------------|---------|--------|
| RanG1  | 10  | 12  | 2   | 00:00| 3            | 1     | 24     | 87         | 211     | 107    |
| RanG2  | 10  | 6   | 2   | 00:00| 2            | 2     | 71     | 73         | 320     | 74     |
| RanG3  | 20  | 39  | 2   | 00:01| 3            | 3     | 61     | 167        | 337     | 349    |
| RanG4  | 20  | 20  | 2   | 00:00| 4            | 9     | 53     | 89         | 271     | 203    |
| RanG5  | 30  | 23  | 2   | 00:00| 7            | 17    | 38     | 128        | 187     | 230    |
| RanG6  | 40  | 40  | 2   | 00:01| 9            | 11    | 86     | 180        | 253     | 429    |
| RanG7  | 50  | 62  | 2   | 00:02| 12           | 15    | 67     | 201        | 462     | 492    |
| RanG8  | 60  | 90  | 2   | 00:04| 15           | 7     | 89     | 345        | 545     | 743    |
| RanG9  | 80  | 159 | 2   | 00:15| 23           | 9     | 112    | 492        | 844     | 1239   |
| RanG10 | 100 | 249 | 2   | 03:18| 32           | 92    | 417    | 920        | 3467    | 2150   |
| RanG11 | 10  | 15  | 3   | 00:00| 5            | 4     | 26     | 89         | 163     | 63     |
| RanG12 | 20  | 39  | 3   | 00:01| 6            | 5     | 43     | 180        | 420     | 274    |
| RanG13 | 30  | 23  | 3   | 00:01| 8            | 8     | 52     | 122        | 289     | 143    |
| RanG14 | 40  | 79  | 3   | 00:01| 9            | 10    | 86     | 183        | 537     | 734    |
| RanG15 | 50  | 62  | 3   | 00:01| 11           | 8     | 43     | 241        | 372     | 420    |
| RanG16 | 100 | 249 | 3   | 03:51| 45           | 112   | 693    | 1378       | 8472    | 1829   |
| RanG17 | 10  | 10  | 4   | 00:00| 5            | 4     | 21     | 150        | 358     | 44     |
| RanG18 | 20  | 20  | 4   | 00:02| 4            | 3     | 39     | 120        | 339     | 90     |
| RanG19 | 40  | 40  | 5   | 00:23| 3            | 13    | 59     | 298        | 632     | 165    |

Table 6.2: IP1 and IP7 formulations applied to random graphs

when we strengthen $IP1$ the difference between both formulations is considerable.

Now we look at the results of another instances. Precisely MIPLIB and NETLIB libraries instances (available at this reference [1]). For these instances we use the same parameter values as [53]. In tables 6.4 and 6.6 we have $Compl$, $B$ and $T.B$. $Compl$ corresponding to the inequalities (4.3). $B$ denotes the number of partitions and $T.B$ is a time spent for computing $B$.

The performances of our two branch-and-cut algorithms on these instances are given in the tables, Table 6.3, Table 6.4, Table 6.5 and Table 6.6 . The results of these tables show that the branch-and-cut approach based upon $IP1&IP7$ and $IP9$ is a robust method to solve instances for the MIPLIB and NETLIB libraries. We can see that all instances, 28 MIPLIB instances and 18 NETLIB instances, are solved exactly in less than one hour i.e. with gap equal to zero, whereas 23 MIPLIB instances and 15 NETLIB instances are solved in the case of [53]. For example, we solved some instances like share2b or stein15 whereas in [53] we don’t have the optimum value.
### 6.3. Experimental results

| Name  | \( |V| \) | \( |E| \) | \( k \) | \( T.B \) | \( Time \) | \( N\#\text{Iter} \) | \( Nodes \) | \( HitSet \) | \( ProjMetric \) | \( B&C \) | \( O.I.P \) |
|-------|--------|--------|------|-------|--------|--------------|--------|--------|--------------|--------|--------|
| afiro | 20     | 20     | 5    | 00:00 | 53     | 40           | 26     | 85     | 126          | 2      | 16     |
| fit1d | 24     | 228    | 6    | 00:01 | 61     | 62           | 103    | 198    | 320          | 16     | 18     |
| fit2d | 25     | 279    | 7    | 00:01 | 73     | 98           | 54     | 73     | 143          | 11     | 18     |
| sc50b | 28     | 110    | 7    | 00:03 | 645    | 660          | 362    | 679    | 3562         | 11     | 11     |
| sc50a | 29     | 95     | 8    | 00:01 | 523    | 689          | 89     | 216    | 753          | 8      | 8      |
| kb2   | 39     | 330    | 10   | 00:01 | 78     | 178          | 95     | 139    | 432          | 14     | 14     |
| vtpbase | 51   | 354    | 13   | 00:11 | 524    | 836          | 317    | 528    | 3585         | 14     | 14     |
| bore3d | 52    | 615    | 13   | 00:08 | 531    | 639          | 429    | 832    | 2754         | 23     | 23     |
| scsd1 | 77     | 202    | 20   | 00:07 | 574    | 498          | 520    | 708    | 3302         | 8      | 8      |
| share2b | 93   | 619    | 24   | 00:40 | 442    | 643          | 734    | 920    | 4720         | 8      | 8      |
| seba  | 2      | 0      | 1    | 00:00 | 1      | 1            | 0      | 0      | 0            | 0      | 0      |
| addlittle | 53  | 239    | 14   | 00:32 | 1289   | 1784         | 611    | 1293   | 8390         | 10     | 10     |
| blend | 54     | 548    | 14   | 00:21 | 456    | 559          | 532    | 840    | 3924         | 20     | 20     |
| recipe | 55     | 129    | 14   | 00:01 | 40     | 19           | 21     | 49     | 67           | 0      | 0      |
| scagr7 | 58     | 661    | 15   | 00:37 | 671    | 847          | 828    | 891    | 5274         | 21     | 21     |
| sc105 | 59     | 356    | 15   | 00:23 | 889    | 1083         | 782    | 945    | 6007         | 16     | 16     |
| stocfor1 | 62  | 272    | 16   | 00:14 | 241    | 187          | 159    | 362    | 930          | 10     | 10     |
| beaconfd | 90   | 1199   | 23   | 03:15 | 3261   | 3782         | 924    | 2749   | 15649        | 26     | 26     |

**Table 6.3:** IP1 and IP7 formulations applied to NETLIB instances

| Name  | \( |V| \) | \( |E| \) | \( k \) | \( B \) | \( T.B \) | \( Time \) | \( N\#\text{Iter} \) | \( Nodes \) | \( Compl \) | \( B&C \) | \( O.I.P \) |
|-------|--------|--------|------|-------|-------|--------|--------------|--------|--------|--------|--------|
| afiro | 20     | 20     | 5    | 11    | 00:02 | 00:00 | 44           | 43     | 21     | 43     | 2      |
| fit1d | 24     | 228    | 6    | 4     | 00:01 | 00:01 | 48           | 70     | 62     | 138    | 16     |
| fit2d | 25     | 279    | 7    | 3     | 00:00 | 00:01 | 62           | 103    | 43     | 60     | 18     |
| sc50b | 28     | 110    | 7    | 4     | 00:01 | 00:04 | 518          | 664    | 509    | 1548   | 11     |
| sc50a | 29     | 95     | 8    | 3     | 00:02 | 00:02 | 415          | 663    | 225    | 413    | 8      |
| kb2   | 39     | 330    | 10   | 4     | 00:01 | 00:02 | 76           | 134    | 73     | 148    | 14     |
| vtpbase | 51   | 354    | 13   | 4     | 00:04 | 00:13 | 534          | 903    | 604    | 1596   | 14     |
| bore3d | 52    | 615    | 13   | 3     | 00:01 | 00:11 | 427          | 748    | 224    | 1275   | 23     |
| scsd1 | 77     | 202    | 20   | 5     | 00:07 | 00:11 | 493          | 523    | 655    | 1473   | 8      |
| share2b | 93   | 619    | 24   | 5     | 00:10 | 00:42 | 398          | 514    | 1757   | 1188   | 8      |
| seba  | 2      | 0      | 1    | 2     | 00:00 | 00:00 | 1            | 1      | 0      | 0      | 0      |
| addlittle | 53  | 239    | 14   | 6     | 00:05 | 00:49 | 1170         | 1161   | 2657   | 3504   | 10     |
| blend | 54     | 548    | 14   | 5     | 00:03 | 00:29 | 432          | 538    | 1000   | 1290   | 20     |
| recipe | 55     | 129    | 14   | 5     | 00:03 | 00:01 | 11           | 7      | 2      | 10     | 0      |
| scagr7 | 58     | 661    | 15   | 4     | 00:03 | 00:39 | 584          | 789    | 1399   | 1746   | 21     |
| sc105 | 59     | 356    | 15   | 6     | 00:05 | 00:37 | 776          | 1029   | 1659   | 1548   | 16     |
| stocfor1 | 62  | 272    | 16   | 13    | 00:08 | 00:25 | 187          | 212    | 575    | 185    | 10     |
| beaconfd | 90   | 1199   | 23   | 3     | 00:09 | 02:36 | 2596         | 4624   | 889    | 10374  | 26     |

**Table 6.4:** IP9 formulation applied to NETLIB instances

102
### Chapter 6. Computational Results

| Name     | $|V|$ | $|E|$ | $k$ | $Time$ | $N$\#Iter | Nodes | HitSet | ProjMetric | $B&C$ | O.I.P |
|----------|------|------|-----|------|----------|-------|--------|-----------|-------|------|
| mod008   | 6    | 15   | 4   | 00:00| 9        | 2     | 2      | 7         | 9     | 2    |
| p0040    | 13   | 30   | 7   | 00:00| 31       | 1     | 12     | 21        | 35    | 3    |
| flugpl   | 16   | 28   | 9   | 00:00| 5        | 1     | 1      | 4         | 6     | 1    |
| p0033    | 15   | 40   | 8   | 00:00| 15       | 7     | 3      | 9         | 17    | 3    |
| gt1      | 15   | 46   | 8   | 00:00| 11       | 4     | 1      | 6         | 8     | 5    |
| stein9   | 13   | 66   | 7   | 00:00| 19       | 13    | 5      | 7         | 19    | 6    |
| rgn      | 24   | 75   | 13  | 00:00| 28       | 21    | 7      | 11        | 27    | 5    |
| sample2  | 45   | 97   | 24  | 00:00| 9        | 3     | 1      | 6         | 11    | 4    |
| enigma   | 21   | 118  | 12  | 00:00| 10       | 9     | 1      | 3         | 6     | 9    |
| mod014   | 74   | 127  | 39  | 00:00| 12       | 16    | 2      | 8         | 13    | 2    |
| mod013   | 62   | 144  | 33  | 00:00| 8        | 7     | 1      | 9         | 16    | 6    |
| lseu     | 28   | 136  | 15  | 00:00| 26       | 23    | 3      | 11        | 23    | 7    |
| stein15  | 36   | 350  | 19  | 00:02| 834      | 1437  | 28     | 174       | 810   | 17   |
| misc01   | 54   | 929  | 29  | 00:05| 401      | 720   | 11     | 84        | 382   | 23   |
| lp4l     | 85   | 1644 | 45  | 09:00| 27456    | 50167 | 537    | 3528      | 27465 | 35   |
| l152lav  | 97   | 1866 | 51  | 01:25| 2671     | 4567  | 156    | 209       | 2649  | 35   |
| kbb05250 | 100  | 1323 | 53  | 00:01| 9        | 5     | 1      | 4         | 9     | 24   |
| misc03   | 96   | 2894 | 51  | 24:34| 38756    | 72901 | 378    | 6389      | 37830 | 43   |
| bm23     | 20   | 190  | 11  | 00:00| 19       | 21    | 2      | 3         | 13    | 9    |
| air01    | 23   | 137  | 13  | 00:00| 23       | 5     | 1      | 6         | 10    | 2    |
| pipex    | 25   | 153  | 14  | 00:01| 9        | 9     | 1      | 4         | 7     | 9    |
| gt2      | 28   | 173  | 15  | 00:00| 25       | 35    | 4      | 13        | 25    | 11   |
| sentoy   | 30   | 435  | 16  | 00:01| 37       | 58    | 3      | 18        | 33    | 14   |
| air02    | 50   | 1126 | 27  | 00:01| 65       | 102   | 7      | 27        | 59    | 21   |
| bell5    | 87   | 226  | 46  | 00:01| 32       | 27    | 5      | 19        | 42    | 4    |
| p0291    | 92   | 521  | 49  | 00:00| 30       | 32    | 4      | 28        | 49    | 7    |
| harp2    | 100  | 1225 | 53  | 00:01| 52       | 41    | 8      | 36        | 62    | 17   |
| misc02   | 43   | 454  | 23  | 00:01| 123      | 199   | 9      | 63        | 126   | 14   |

Table 6.5: IP1 and IP7 formulations applied to MIPLIB instances
6.3. Experimental results

| Name    | $|V|$ | $|E|$ | $k$ | $B$ | Time  | $N\#\text{Iter}$ | Nodes | Compl | $B\&C$ | $O/IP$ |
|---------|-----|-----|-----|-----|------|-------------------|-------|-------|--------|--------|
| mod008  | 6   | 15  | 4   | 2   | 00:00| 00:00             | 8     | 3     | 2      | 4      | 2      |
| p0040   | 13  | 30  | 7   | 2   | 00:01| 00:00             | 26    | 1     | 6      | 18     | 3      |
| flugpl  | 16  | 28  | 9   | 2   | 00:00| 00:00             | 3     | 1     | 0      | 1      | 1      |
| p0033   | 15  | 40  | 8   | 3   | 00:00| 00:00             | 13    | 9     | 3      | 8      | 3      |
| gt1     | 15  | 46  | 8   | 2   | 00:00| 00:00             | 7     | 5     | 0      | 5      | 5      |
| stein9  | 13  | 66  | 7   | 2   | 00:01| 00:00             | 17    | 13    | 5      | 10     | 6      |
| rgn     | 24  | 75  | 13  | 2   | 00:01| 00:00             | 23    | 27    | 8      | 13     | 5      |
| sample2 | 45  | 97  | 24  | 3   | 00:02| 00:00             | 7     | 4     | 1      | 5      | 4      |
| enigma  | 21  | 118 | 12  | 2   | 00:01| 00:00             | 7     | 8     | 0      | 5      | 9      |
| mod014  | 74  | 127 | 39  | 2   | 00:02| 00:00             | 11    | 8     | 3      | 6      | 2      |
| mod013  | 62  | 144 | 33  | 2   | 00:02| 00:00             | 7     | 4     | 1      | 4      | 6      |
| lseu    | 28  | 136 | 15  | 2   | 00:02| 00:01             | 20    | 19    | 6      | 12     | 7      |
| stein15 | 36  | 350 | 19  | 2   | 00:03| 00:04             | 711   | 1365  | 120    | 589    | 17     |
| misc01  | 54  | 929 | 29  | 2   | 00:05| 00:08             | 364   | 701   | 87     | 276    | 23     |
| lp4l    | 85  | 1644| 45  | 2   | 00:11| 12:00             | 26343 | 49767 | 5343   | 20998  | 35     |
| li52lav | 97  | 1866| 51  | 2   | 00:09| 01:39             | 2535  | 4656  | 410    | 2123   | 35     |
| kkh05250| 100 | 1323| 53  | 2   | 00:06| 00:01             | 4     | 3     | 0      | 3      | 24     |
| misc03  | 96  | 2894| 51  | 2   | 00:14| 30:56             | 37525 | 73725 | 2212   | 35312  | 43     |
| bm23    | 20  | 190 | 11  | 2   | 00:01| 00:00             | 13    | 19    | 4      | 7      | 9      |
| air01   | 23  | 137 | 13  | 2   | 00:02| 00:00             | 17    | 3     | 6      | 9      | 2      |
| pipex   | 25  | 153 | 14  | 2   | 00:00| 00:01             | 8     | 8     | 2      | 4      | 9      |
| gt2     | 28  | 173 | 15  | 2   | 00:02| 00:01             | 20    | 33    | 6      | 13     | 11     |
| sentoy  | 30  | 435 | 16  | 2   | 00:03| 00:01             | 29    | 50    | 5      | 22     | 14     |
| air02   | 50  | 1126| 27  | 2   | 00:12| 00:03             | 57    | 94    | 12     | 43     | 21     |
| bell5   | 87  | 226 | 46  | 2   | 00:04| 00:01             | 21    | 23    | 8      | 11     | 4      |
| p0291   | 92  | 521 | 49  | 3   | 00:05| 00:00             | 21    | 31    | 7      | 12     | 7      |
| harp2   | 100 | 1225| 53  | 2   | 00:12| 00:02             | 42    | 30    | 12     | 16     | 17     |
| misc02  | 43  | 454 | 23  | 2   | 00:08| 00:01             | 108   | 188   | 27     | 80     | 14     |

Table 6.6: IP9 formulation applied to MIPLIB instances
6.4 Conclusion

In this chapter some formulations are evaluated and some branch and cut algorithms are also presented. We study before concluding this thesis some approximation algorithms.
Chapter 7

Approximations

Contents

7.1 Introduction ................................................. 106
7.2 Approximation algorithms ............................... 106
  7.2.1 LP-Based approximation algorithms ............. 106
  7.2.2 Primal Dual approach ............................... 107
  7.2.3 Greedy approximation algorithm ................. 108
7.3 Inapproximability ......................................... 108
7.4 Conclusion ................................................. 109

7.1 Introduction

This chapter is devoted to approximability. The first algorithm is based on linear programming. The second algorithm uses a greedy method. Inapproximability is also studied in this chapter.

7.2 Approximation algorithms

7.2.1 LP-Based approximation algorithms

The first approximation algorithm we give relies on the linear relaxation ($LP_1$) of the integer program ($IP_1$) introduced in Section 4.2. Notice that the separation of
inequalities “\( \sum_{v \in S} x_v \geq 1, \forall S \subset V, |S| = k + 1, G(S)\) connected ” in \((LP1)\) is NP-hard even if all vertex-weights belong to \(\{0, 1\}\) when \(k\) is part of the input [63]. If \(k\) is constant, the separation is obviously easy. It is also known that a maximum-weight connected subgraph of size \(k + 1\) can be computed easily if the graph is a tree [63].

An LP-based approximation algorithm 9 is obtained by generalizing the basic approximation algorithm for the vertex cover problem.

A connected subgraph \(G(S)\) is said to be large if \(|S| \geq k + 1\).

**Algorithm 9** LP-based Approximation Algorithm

1. **Input:** A vertex-weighted undirected graph \(G = (V, E, w)\) and an integer \(k\).
2. **Output:** A \(k\)-separator \(S\).
3. Solve \((LP1)\) and let \(x\) be an optimal solution of \((LP1)\).
4. Set \(S := \emptyset\).
5. **while** \(G(V \setminus S)\) contains large connected components **do**
6. Select \(R \subset V \setminus S\) such that \(|R| = k + 1\) and \(G(R)\) connected.
7. Select \(v \in R\) such that \(x_v\) is maximum and set \(S := S \cup \{v\}\).
8. **end while**

**Proposition 7.1** The LP-based approximation algorithm (Algorithm 9) is a \((k+1)\)-approximation algorithm.

**Proof:** Since \(\sum_{y \in R} x_y \geq 1\) for each subset \(R \subset V \setminus S\) where \(|R| = k+1\) and \(G(R)\) is connected, the vertex \(v\) (maximizing \(x_v\) inside \(R\)) necessarily satisfies \(x_v \geq \frac{1}{k+1}\). Adding \(v\) to \(S\) is equivalent to rounding \(x_v\) to 1. The final solution is clearly a \(k\)-separator. The weight of this \(k\)-separator is not more than \(k + 1\) times the weight of the fractional solution \(x\) (since in the rounding procedure, \(x_v\) is multiplied by at most \(k + 1\)). Since the weight of the fractional solution \(x\) is a lower bound of the optimal weight, we deduce that we have a \((k + 1)\)-approximation. ■

### 7.2.2 Primal Dual approach

Observe that the algorithm described above is a polynomial-time algorithm if we assume that \(k\) is bounded by a constant. This is necessary to guarantee that the size of \((LP1)\) is polynomial. The primal-dual approach (see, e.g., [86]) leads also to a \((k + 1)\)-approximation. In fact, the \(k\)-separator problem is a special-case of the hitting set problem where we want to hit large connected components.
7.2.3 Greedy approximation algorithm

If all vertex weights are equal to 1, then there is another simple \((k+1)\)-approximation algorithm (Algorithm 10).

\textbf{Algorithm 10} Greedy Approximation Algorithm

1: \textbf{Input}: A graph \(G = (V, E)\) and an integer \(k\).
2: \textbf{Output}: A \(k\)-separator \(S\).
3: Set \(S := \emptyset\).
4: \textbf{while} \(G(V \setminus S)\) contains large connected components \textbf{do}
5: \hspace{1em} Select \(R \subset V \setminus S\) such that \(|R| = k + 1\) and \(G(R)\) is connected.
6: \hspace{1em} \(S := S \cup R\).
7: \textbf{end while}

\textbf{Proposition 7.2} For the case when all vertex weights are equal to 1, the greedy algorithm (Algorithm 10) is a \((k + 1)\)-approximation algorithm for the \(k\)-separator problem.

\textbf{Proof}: Since the algorithm stops only when there are no large connected components, the final set \(S\) is a \(k\)-separator. Each subset \(R\) selected in any iteration is a large connected component. Then, we know that any optimal \(k\)-separator should contain at least one vertex from this subset \(R\). Since we put all vertices of \(R\) in \(S\) and \(|R| = k + 1\), the size of \(S\) cannot be more than \(k + 1\) times the size of an optimal \(k\)-separator. \(\blacksquare\)

The greedy algorithm obviously has a polynomial time complexity even if \(k\) is part of the input.

7.3 Inapproximability

Notice that we should not expect much better approximation algorithms since it is shown that the vertex cover (corresponding with \(k = 1\)) cannot be approximated within a factor of 1.3606 \cite{35} unless \(P = NP\). It is even hard to approximate it within a factor less than 2 if the unique games conjecture is true \cite{73}.

Finally, since computing a minimum-weight \(k\)-separator is equivalent to maximizing the weight of the vertices that are not in the \(k\)-separator, one can also study the approximability of the maximization problem. Let us call this problem the maximum \(k\)-coseparator problem. For \(k = 2\), it is shown in \cite{90} that this problem cannot
be approximated within a factor of $|V|^{1/2-\varepsilon}$ for any constant $\varepsilon > 0$. We extend their results for any $k$ using the same reduction technique.

**Proposition 7.3** Assuming that $P \neq NP$, the maximum $k$-coseparator problem cannot be approximated in polynomial time within a factor of $(\frac{|V|}{k})^{1-\varepsilon}$ for any constant $\varepsilon > 0$.

**Proof:** Let us focus on instances of the $k$-coseparator problem with unit weights and let $\text{cosep}_k(G)$ denote the maximum size of the complement of a $k$-separator of a graph $G$. Consider an instance of the maximum stable set problem given by a graph $G = (V, E)$. Build a new graph $G' = (V', E')$ by $k$ duplications of each vertex $v \in V$ (each vertex $v \in V$ is replaced by $k$ vertices $v_1, \ldots, v_k$), and adding edges $(u_i, v_j)$ for $1 \leq i \leq j \leq k$ when $(u, v) \in E$. It is easy to see that $\text{cosep}_k(G') = k\alpha(G)$ where $\alpha(G)$ is the size of a maximum stable set of $G$. Indeed, a stable set of size $\alpha(G)$ directly leads to a $k$-coseparator of size $k\alpha(G)$ by replacing each vertex of the stable set by its $k$ duplicate vertices. Moreover, given a maximum size $k$-coseparator of $G'$, if two adjacent vertices $u_i$ and $v_j$ belong to the same connected component, then there is at least an index $l$ ($1 \leq l \neq i \leq k$) such that $u_l$ does not belong to the $k$-coseparator (otherwise the size of the connected component will be strictly greater than $k$). By deleting $v_j$ and adding $u_l$, we get a new $k$-coseparator of maximum size. By repeating this process, we should obtain a $k$-coseparator where each connected component contains exactly the $k$ duplicate vertices $v_1, \ldots, v_k$ of some vertex $v \in V$. By considering the vertices $v$ whose duplicate vertices are inside the $k$-coseparator, we get a stable set of $G$ whose size is $1/k$ times the size of the $k$-coseparator of $G'$. Consequently, $\text{cosep}_k(G') = k\alpha(G)$. Using the fact that the maximum stable set of $G$ cannot be approximated within $|V|^{1-\varepsilon}$ [32, 93], we deduce that the maximum $k$-coseparator of $G'$ cannot be approximated within $(\frac{|V'|}{k})^{1-\varepsilon}$ in polynomial time unless $P = NP$. ■

### 7.4 Conclusion

Approximation algorithms are presented above. Inapproximability is also studied. For big instances, a heuristic approach must be considered.
Chapter 8

Conclusion

In this thesis we presented and studied the $k$-separator problem which consists, given some vertex-weighted graph $G = (V, E)$, in determining a minimum-weight set of vertices $S \subseteq V$ such that no component in the subgraph induced by $V \setminus S$ has size strictly larger than $k$. Connections with other classical combinatorial optimization problems have been established and cases when the problem is easy to solve (i.e. polynomial time solvable) have been identified and methods for such cases, proposed. A polyhedral study has then been undertaken, leading to many valid inequalities that may be used to strengthen formulations of the problem. Part of these inequalities have been integrated in different cutting-plane algorithms that have been applied on a wide range of instances. These evaluations illustrate the potential advantages and limits for some of the different formulations presented throughout this work. Then different formulations of the problem and relaxations have also been studied and compared. Particularly, with respect to linear relaxations. Some approximation results and algorithms have been demonstrated.

A matter for future research work is the exhibition of some new classes of problems for which better approximation algorithms can be provided. Completing the polyhedral description when the problem can be solved in polynomial-time (such as for trees) deserves further research.

A heuristic algorithm is required in the case of big instances for the future works. The graphs related to social network are known to be huge. Applying the results obtained in this thesis for these graphs can be useful to detect communities.
Appendix A

Résumé du manuscrit de thèse en français

Contents

A.1 Introduction .................................................. 111
A.2 Problématique .................................................. 112
A.3 Propositions .................................................. 112
   A.3.1 Approches Polyédrales ................................. 113
      A.3.1.1 Cas polynomiaux ................................. 113
      A.3.1.2 Formulations ................................. 116
A.3.2 Algorithmes d’approximation ......................... 121
A.4 Conclusion & Perspective ................................. 121

A.1 Introduction

Nous assistons depuis des années dans le contexte de réseaux sociaux au développe-
ment des approches décentralisées de la gestion avec une volonté d’assurer une cer-
taine capacité à s’auto-configurer et à résister aux pannes et aux variations de la
topologie du réseau. L’approche la plus répandue consiste à former des clusters avec
éventuellement plusieurs niveaux et à essayer d’organiser les communications d’une
manière hiérarchique et dynamique.

Nous pouvons signaler le fait que les réseaux sociaux se modélisent généralement par
A.2. Problématique

un graphe (orienté ou non selon le cas) où les liens modélisent des échanges ou des points communs entre les membres du réseau (ou groupe). La détection des communautés à l’intérieur d’un réseau social est un sujet qui a fait couler beaucoup d’encre [8]. L’idée de base consiste à voir les communautés comme des sous-graphes denses. Plusieurs algorithmes ont été développés pour y parvenir incluant le très classique algorithme des $k$-moyennes et différents algorithmes de partitionnement de graphes. Les points communs entre les réseaux de télécommunications et les réseaux sociaux sont très nombreux. Le point qui nous intéresse le plus ici est le fait que dans les deux cas, on essaye de voir (ou de construire) le réseau comme des clusters (des sous-graphes denses). Dans la section suivante A.2 on présente le thème principal de la thèse. La section A.3 décrit les approches développées. Enfin nous essayons de tirer les conclusions qui s’imposent et nous proposons quelques perspectives dans la section A.4.

A.2 Problématique

L’objectif de cette thèse est la généralisation d’un problème connu de la théorie de graphes et son étude en caractérisant les cas où le problème est polynomial ou approximable avec un bon rapport. Le problème à étudier consiste plus précisément en la construction d’algorithmes afin de déterminer le nombre minimum de nœuds qu’il faut enlever à un réseau (ou graphe) pour que toutes les composantes connexes restantes contiennent chacune au plus $k$-somments. Ce problème on l’appelle Problème de $k$-Séparateur et on désigne par $k$-séparateur le sous-ensemble recherché. Il est une généralisation du Vertex Cover qui correspond au cas $k = 1$ (nombre minimum de sommets intersectant toutes les arêtes du graphe).

A.3 Propositions

Nous travaillons sur deux volets à savoir les méthodes exactes basées sur les approches polyédrales et les algorithmes d’approximation avec garantie de performance. Avant de présenter les approches adoptées, nous introduisons quelques notations. Notons par $G = (V, E, w)$ un graphe non orienté dont les sommets sont pondérés. Étant donné un sous-ensemble $S \subset V$, $\chi^{(S)} \in \{0, 1\}^n$ désigne le vecteur d’incidence de $S$. $P_k$ est l’enveloppe convexe pour tous les $k$-séparateurs. On note
Appendix A. Résumé du manuscrit de thèse en français

par $G(S)$ le sous-graphe induit par $S \subset V$.

**A.3.1 Approches Polyédrales**

Elles consistent à déterminer un système d’inégalités linéaires décrivant l’enveloppe convexe de tous les $k$-séparateurs.

**A.3.1.1 Cas polynomiaux**

**A.3.1.1.a Graphes avec largeur arborescente bornée**

Une décomposition arborescente de $G$ est définie par un couple $(X, T)$ où $X = (X_t)_{t \in V(T)}$ est une famille de sous-ensembles de sommets de $V$ étiquetés par les sommets d’un arbre $T$, tels que:

(i) pour chaque $v \in V(G)$, il existe $t \in V(T)$ tel que $v \in X_t$;

(ii) pour chaque arête $(u, v)(G)$, il existe $t \in V(T)$ tel que $u \in X_t$ et $v \in X_t$;

(iii) pour chaque sommet $v \in V(G)$, si $v \in X_{t_1}$ et $v \in X_{t_2}$ alors $v$ appartient à $X_t$ pour chaque $t \in V(T)$ sur le chemin entre $t_1$ et $t_2$.

Propriété (iii) implique que le sous-graphe de $T$ induit par les sommets $t$ tel que $X_t$ contient $v$ est un sous-arbre. La largeur de la décomposition est donnée par $\max_{t \in V(T)} |X_t| - 1$. La largeur arborescente ($treenwidth$) de $G$ est la largeur minimale sur toutes les décompositions arborescentes de $G$.

Nous supposons ici que $G$ a une largeur arborescente bornée par une constante $l$. Le calcul de la largeur arborescente de $G$ peut être fait en temps polynomial (en supposant que $l$ est constante) [6]. Beaucoup de problèmes d’optimisation NP-complets peuvent être résolus en temps polynomial pour les graphes de largeur arborescente bornée. Les algorithmes utilisés sont généralement basés sur la programmation dynamique et une décomposition arborescente du graphe (plus de détails dans les références [71, 7, 78]). Une approche générale est proposée dans [78] pour résoudre les problèmes de partitionnement dans les graphes de largeur arborescente bornée. Vu que notre problème, c.à.d. le problème de $k$-séparateur peut être considéré comme un problème de partitionnement où les partitions sont données par le $k$-séparateur et les composantes connexes restantes, l’approche de [78] peut être utilisée dans notre cas.
A.3. Propositions

Proposition A.1 Le problème du \( k \)-séparateur peut être résolu en temps polynomial dans le cas de chemins, de cycles et plus généralement des graphes avec largeur arborescente bornée même si \( k \) est paramètre de l’entrée.

A.3.1.1.b Graphes sans couplage \( mK_2 \) induit

Avant de présenter les graphes sans couplage \( mK_2 \) induit, nous introduisons une construction d’un graphe étendu \( H \) à partir du graphe \( G \) permettant de transformer le problème du \( k \)-séparateur à un problème de stable de poids maximal. L’idée consiste à créer un graphe étendu \( H = (V(H), E(H)) \) à partir du graphe \( G \). Chaque sous-ensemble de sommets \( S \subset V \) tel que \( 1 \leq |S| \leq k \) et \( G(S) \) est connexe est représenté par un sommet dans \( H \). \( V(H) = \{S \subset V, |S| \leq k, G(S) \text{ est connexe}\} \).

Les arêtes sont définies comme suit : \( E(H) = \{(S, T), S \in V(H), T \in V(H), S \neq T, \text{ tel que } S \cap T \neq \emptyset, \text{ ou } (u, v) \in E \text{ avec } u \in S \text{ et } v \in T\} \). Autrement dit, \( S \in V(H) \) et \( T \in V(H) \) sont reliés par une arête si les sous ensembles de sommets de \( G \) qui correspondent à \( S \) et \( T \), comportent un sommet commun ou contiennent deux sommets adjacents. Le poids du sommet \( S \in V(H) \) est égal à \( w_S = \sum_{v \in S} w_v \).

Notons par \( R \) le stable de poids maximum de \( H \). Si deux sommets \( S \in V(H) \) et \( T \in V(H) \) appartiennent à ce stable \( R \), alors \( S \cap T = \emptyset \) ne contient pas une arête dans \( G \) avec une extrémité dans \( S \) et l’autre extrémité dans \( T \). Autrement dit, si on considère \( \cup_{S \in R} S \), on obtient un ensemble de sommets dont chaque composante connexe est de taille inférieure ou est égale à \( k \). Le complémentaire de \( \cup_{S \in R} S \) est un \( k \)-séparateur.

Cette construction du graphe étendu peut être considérée comme une généralisation de la construction proposée dans [81] pour le problème de la dissociation \((k = 2)\).

Supposons maintenant que \( G \) ne contient pas un couplage \( m \) induit où \( m \) est une constante. Cela est équivalent à dire que \( G \) est sans couplage \( mK_2 \). Dans ce cas le problème de dissociation est facile à résoudre comme il est prouvé dans [90]. Et comme le problème de dissociation est un cas particuliers du problème de \( k \)-séparateur \((k = 2)\), nous généralisons ce résultat pour toute constante \( k \).

Proposition A.2 Le problème de \( k \)-séparateur peut être résolu en temps polynomial pour les graphes sans les couplages \( mK_2 \) induit dans le cas où \( m \) et \( k \) sont des constantes.
Appendix A. Résumé du manuscrit de thèse en français

A.3.1.1.c Graphes sans \((G_1, G_2, G_3, P_6)\) induits

Soit \(G_1\) le graphe de chaise (ou fourchette) représenté sur la gauche de la figure 3.1. Dans [82] il est démontré que le problème de stable de poids maximal est résolu en temps polynomial si le graphe est ne contient pas \(G_1\). Lorsque \(k = 2\), il est prouvé dans [90] que le graphe étendu \(H\) ne contient pas \(G_1\) si et seulement si \(G\) ne contient pas les graphes \((G_1, G_2, G_3)\) où \(G_2\) et \(G_3\) sont présentés sur la figure 3.1. Nous allons étendre ce résultat lorsque \(k \geq 3\). Plus précisément, nous montrons que \(H\) ne contient pas \(G_1\) si et seulement si \(G\) ne contient pas \((G_1, G_2, G_3, P_6)\) où \(P_6\) est un chemin contenant 6 sommets (figurant sur la partie droite de la figure 3.1).

**Proposition A.3** Soit \(k \geq 3\), le graphe étendu \(H\) ne contient pas le graphe \(G_1\) si et seulement si le graphe \(G\) ne contient pas les graphes \((G_1, G_2, G_3, P_6)\).

**Corollary A.1** En supposant que \(k\) est une constante \(\geq 3\), le problème du \(k\)-séparateur peut être résolu en temps polynomial si le graphe \(G\) ne contient pas les graphes \((G_1, G_2, G_3, P_6)\).

A.3.1.1.d Graphes de type Interval-filaments, Graphes sans asteroidal triple et Graphes de type weakly chordal

Les résultats de cette section sont une conséquence directe des résultats de [39]. Considérons une collection \(L\) d’intervalles sur une ligne. Supposons que, pour chaque intervalle, on donne une courbe au-dessus de la ligne reliant les extrémités de l’intervalle, et en restant dans les limites de l’intervalle. Un graphe est de type interval-filament s’il est défini par l’intersection d’une telle collection d’intervalles [29] (voir la figure 3.2). Le problème de stable max dans le cas de graphes de type "interval filament" se résout en temps polynomial [29]. Le même résultat a été également prouvé dans [39] pour la classe de graphes de type weakly chordal [33] ( graphe tel que ni le graphe en soi-même ni son complémentaire contiennent un cycle induit de taille 5 ou plus) et la classe de graphes sans asteroidal-triple (graphes qui ne contenant pas de stable de taille 3 de telle sorte que, entre chaque paire de sommets de ce triplet, il existe un chemin qui les relie, et en évitant le voisinage du troisième sommet).

**Proposition A.4** Soit \(k\) une constante, le problème de \(k\)-séparateur peut être résolu en temps polynomial pour les graphes de type Interval-filament, graphes sans
A.3. Propositions

asteroidal-triple et les graphes de type weakly-chordal.

A.3.1.1.e Graphes à intervalles et arc-circulaire

Les graphes à intervalles sont des graphes où chaque sommet correspond à un intervalle et une arête \((u, v)\) existe s’il y a une intersection non vide entre les intervalles représentés par \(u\) et \(v\) (voir figure 3.5.a). Nous avons montré dans cette thèse que le problème du \(k\)-séparateur est facile à résoudre pour les graphes d’intervalles.

**Proposition A.5** Le problème du \(k\)-séparateur peut être résolu en temps polynomial pour les graphes d’intervalles même si \(k\) n’est pas une constante.

Les graphes arc circulaires sont une généralisation simple de graphes d’intervalles. Par définition il s’agit des graphes d’intersection d’un ensemble d’arcs sur un cercle (voir figure 3.5.b).

**Proposition A.6** Le problème de \(k\)-séparateur peut être résolu en temps polynomial pour les graphes arc circulaires même si \(k\) n’est pas une constante.

A.3.1.2 Formulations

A.3.1.2.a Formulation de base

Soit \(S\) un sous-ensemble de sommets tels que \(|S| = k + 1\) et \(G(S)\) est connexe. L’inégalité suivante est évidemment valable pour \(S_k(G)\).

\[
\sum_{v \in S} x_v \geq 1. \tag{A.1}
\]

Le problème du \(k\)-séparateur peut être formulé comme le programme entier suivant:

\[
\text{IP1} \left\{ \begin{array}{ll}
\min & \sum_{v \in V} w_v x_v \\
\text{s.t.} & \sum_{v \in S} x_v \geq 1, \forall S \subset V, |S| = k + 1, G(S) \text{est connexe} \\
x_v \in \{0, 1\}, \forall v \in V
\end{array} \right.
\]

Notons par \(LP1\) la relaxation linéaire de \(IP1\).
Appendix A. Résumé du manuscrit de thèse en français

A.3.1.2.b Formulation de Stable de poids maximal

Cette formulation est basée sur le graphe étendu $H = (V(H), E(H))$ de la construction de la section A.3.1.1.b. Comme $V(H) = \{S \subset V, |S| \leq k, G(S) est connexe\}$ et $E(H) = \{(S, T), S \in V(H), T \in V(H), S \neq T, tel que S \cap T \neq \emptyset, ou (u, v) \in E avec u \in S et v \in T\}$. Le lien entre le problème de $k$-séparateur et le stable de poids maximal est déjà fait dans la section A.3.1.1.b, ce qui nous donne la formulation suivante.

\[
\begin{align*}
\min & \sum_{v \in V} w_v x_v \\
\text{s.t.} & x_v = 1 - \sum_{S \in V(H), v \in S} y_S \quad \forall v \in V \\
y_S & \in \{0, 1\} \quad \forall S \in V(H) \\
y_S + y_T & \leq 1 \quad \forall S \in V(H), T \in V(H), (S, T) \in E(H)
\end{align*}
\]

Cette formulation peut être renforcée par des inégalités de cycles impairs, cliques, etc.[76].

Soit $Q_v = \{S \in V(H) : v \in S\}$. On peut ajouter à de $IP2$ les inégalités valides suivantes : $\sum_{S \in Q_v \cup Q_w} y_S \leq 1, \forall (v, w) \in E$. Le nombre de ces inégalités est polynomial ($|E|$). Cela nous donne la formulation $IP3$.

\[
\begin{align*}
\min & \sum_{v \in V} w_v x_v \\
x_v = 1 - \sum_{S \in Q_v} y_S, \forall v \in V \\
& \sum_{S \in Q_v \cup Q_w} y_S \leq 1, \forall (v, w) \in E \\
y_S & \in \{0, 1\}, \forall S \in V^*
\end{align*}
\]

Notons par $LP3$ la relaxation linéaire de $IP3$. Soit $F1$ (resp. $F3$) l’ensemble des solutions possibles de $LP1$ (resp. $LP3$) par rapport aux variables $(x_v)_{v \in V}$.

**Proposition A.1** L’inclusion suivante est vérifiée : $F3 \subseteq F1$.

A.3.1.2.c Formulation métrique

Une formulation métrique est proposée dans [53]. En plus des variables $(x_i)_{i \in V}$. Considérons la variable $x_{ij}$ qui indique pour chaque paire de sommets $\{i, j\}$ si $i$ et $j$ appartiennent à la même composante ou non. Plus précisément, $x_{ij}$ est égal à
A.3. Propositions

0 si elles sont dans la même composante connexe. On peut voir que les inégalités triangulaires sont valides. Pour exprimer le fait qu’une composante connexe ne contient pas plus de $k$ sommets, nous pouvons ajouter les contraintes $\sum_{j \in V \setminus \{i\}} x_{ij} \geq n-k, \forall i \in V$. Enfin, il faut ajouter les contraintes qui imposent que si deux sommets sont adjacents et qu’ils ne sont pas dans le $k$-séparateur alors, ils appartiennent à la même composante : $x_i + x_j - x_{ij} \geq 0, \forall (i, j) \in E$. La formulation est donnée ci-dessous.

**IP4**

\[
\begin{align*}
\text{min} \sum_{v \in V} w_v x_v \\
x_{ij} &\leq x_{ik} + x_{jk}, \forall i, j, k \in V \\
\sum_{j \in V \setminus \{i\}} x_{ij} &\geq n-k, \forall i \in V \\
x_i + x_j - x_{ij} &\geq 0, \forall (i, j) \in E \\
x_i &\in \{0, 1\}, \forall i \in V
\end{align*}
\]

Nous présentons ci-dessous une nouvelle formulation qui renforce la relaxation linéaire de **IP4**.

**LP5**

\[
\begin{align*}
\text{min} \sum_{v \in V} w_v x_v \\
\sum_{j \in V \setminus \{i\}} x_{ij} &\geq n-k + (k-1)x_i, \forall i \in V \\
x(p) - x_{ij} &\geq 0, \forall i, j \in V, \ p \in P_{ij} \\
0 &\leq x_{ij} \leq 1, \forall i, j \in V \\
0 &\leq x_i \leq 1, \forall i \in V
\end{align*}
\]

Une autre formulation compacte est présentée ci-dessous.

**LP6**

\[
\begin{align*}
\text{min} \sum_{v \in V} w_v x_v \\
\sum_{j \in V \setminus \{i\}} x_{ij} &\geq n-k + (k-1)x_i, \forall i \in V \\
y_{ij} &\leq x_i + x_j, \forall (i, j) \in E \\
y_{ij} &\leq x_i + y_{jk}, \forall i, j \in V, \ (i, k) \in E \\
y_{ij} - x_{ij} &\geq 0, \forall i, j \in V \\
0 &\leq x_{ij} \leq 1, 0 \leq y_{ij}, \forall i, j \in V \\
0 &\leq x_i \leq 1, \forall i \in V
\end{align*}
\]

**LP5** et **LP6** sont équivalentes.
Appendix A. Résumé du manuscrit de thèse en français

A.3.1.2.d Formulation métrique projetée

Soit $S$ un ensemble de sommets avec $|S| \geq k$ et soit $i \in S$. Pour chaque $j \in S$, notons par $p_{ij} \in P_{ij}$ un chemin joignant $i$ et $j$. Considérons l’inégalité suivante

$$(|S| + 1 - k)(1 - x_i) \leq \sum_{j \in S} x(p_{ij} \setminus \{i\}). \quad (A.2)$$

**Lemma A.1** Les inégalités (A.2) sont valides pour $S_k(G)$.

Considérons maintenant la formulation basée sur les inégalités (A.2).

$$\begin{align*}
\text{IP}_7 \left\{ \begin{array}{l}
\min \sum_{v \in V} w_v x_v \\
(|S| + 1 - k)(1 - x_i) \leq \sum_{j \in S} x(p_{ij} \setminus \{i\}), \quad \forall i \in V, \ S \subset V \setminus \{i\}, \ |S| \geq k; \ p_{ij} \in P_{ij}, \forall j \in S \\
x_v \in \{0, 1\}, \forall v \in V
\end{array} \right.
\end{align*}$$

**Lemma A.2** La Formulation IP$_7$ est exacte.

Notons par LP$_7$ la relaxation linéaire de IP$_7$.

**Proposition A.2** La formulation LP$_7$ est équivalente aux formulations LP$_5$ et LP$_6$. Elle est même plus forte que la formulation LP$_4$.

Il n’est pas difficile de montrer qu’il n’y a pas de domination relative entre LP$_7$ et LP$_1$ (aucune formulation ne domine l’autre en général).

A.3.1.2.e Formulation de partitionnement

Une autre formulation pour le problème de $k$-séparateur peut être inspirée du problème de partitionnement ou regroupement [44, 53]. Soit $B$ une borne supérieure du nombre de composantes connexes qui seront obtenues après la suppression du $k$-séparateur. $B$ peut être, par exemple, égal à $n$. Les composantes sont alors numérotées de 1 à $B$. Une variable $z_{ib}$ est définie pour chaque sommet $i$ et chaque composante $b \in \{1, \ldots, B\}$. $z_{ib}$ sera égal à 1 si $i$ appartient à la composante $b$. La
La première famille de contraintes exprime le fait qu'un sommet $i$ est soit dans le $k$-séparateur ($x_i = 1$) ou dans l'une des composantes connexes. La seconde famille d'inégalités permet de limiter la taille de chaque composante connexe à $k$, tandis que les contraintes $z_{ib} + z_{jb'} \leq 1$ assurent que les sommets adjacents appartiennent aux même composantes.

En fait, les contraintes $z_{ib} + z_{jb'} \leq 1$ sont dominées par les inégalités $z_{ib} - z_{jb} \leq x_j$, $\forall (i,j) \in E$ et $b \in \{1, ..., B\}$. Une autre formulation exacte et compacte peut alors être obtenue.

Une amélioration est obtenue en considérant un sous-ensemble $U \subset \{1, ..., B\}$, son complémentaire $\overline{U}$ et deux sommets adjacents $i$ et $j$:

$$\sum_{b \in U} z_{ib} + \sum_{b \in \overline{U}} z_{jb} \leq 1, \ \forall (i,j) \in E. \quad (A.3)$$

Proposition A.3 La borne supérieure minimale $B$ qui peut être considérée dans les formulations $IP8$ et $IP9$ est égale à la taille maximale du stable dans $G$. 

120
Appendix A. Résumé du manuscrit de thèse en français

A.3.2 Algorithmes d’approximation

Le premier algorithme est une généralisation d’un autre algorithme d’approximation du vertex cover.

Algorithm 11 Algorithm d’approximation basé sur la programmation linéaire

1: Entrée: un graphe $G = (V, E, w)$ dont les sommets sont pondérés et un entier $k$.
2: Sortie: un $k$-séparateur $S$.
3: résoudre $(LP1)$ et soit $x$ la solution optimale de $LP1$.
4: $S = \emptyset$.
5: TANTQUE $G(V \setminus S)$ contient une composante large FAIRE
6: Choisir $R \subset V \setminus S$ tel que: $|R| = k + 1$ et $G(R)$ connexe.
7: Choisir $v \in R$ tel que: $x_v$ est maximum et on pose $S = S \cup \{v\}$.
8: FINTANTQUE

En utilisant la méthode primale-duale [86] nous avons un autre algorithme d’approximation.

Algorithm 12 Algorithme glouton

1: Entrée: un graphe $G = (V, E, w)$ dont les sommets sont pondérés et un entier $k$.
2: Sortie: un $k$-séparateur $S$.
3: $S = \emptyset$.
4: TANTQUE $G(V \setminus S)$ contient une composante large FAIRE
5: Choisir $R \subset V \setminus S$ tel que: $|R| = k + 1$ et $G(R)$ connexe.
6: $S = S \cup R$.
7: FINTANTQUE

Notons que les deux algorithmes sont $(k + 1)$-approchés.

A.4 Conclusion & Perspective

Dans cette thèse nous avons présenté le problème de $k$-Séparateur. Il consiste à trouver le sous-ensemble de sommets de poids minimal à supprimer dans un graphe non orienté dont les sommets sont pondérés afin d’obtenir des sous-ensembles connexes de taille inférieure ou égale à un entier positif $k$ donné. Une étude polyédrale a été faite, conduisant à de nombreuses inégalités valides qui peuvent être utilisées pour renforcer les différentes formulations linéaires du problème. Une partie de ces inégalités a été implémentée dans des algorithmes de type branch-and-cut et ces algorithmes ont été appliqués sur une large variété d’instances. Les différentes formulations du problème et relaxations ont également été étudiées et comparées.
A.4. Conclusion & Perspective

Des cas où le problème peut être résolu en temps polynomial sont présentés. Des algorithmes d’approximation avec garantie de performance ont été exposés.

Enfin, appliquer les résultats obtenus dans cette thèse pour le problème de \( k \)-séparateur à des cas réels tel que, les réseaux sociaux par exemple nous semble bénéfique si on prend le problème de détection des communautés dans ceux-ci.
Bibliography


Bibliography


129


