Titre: Title:	A Novel Approach to Tightening Semidefinite Relaxations for Certain Combinatorial Problems
Auteur: Author:	Elspeth Adams
Date:	2015
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Adams, E. (2015). A Novel Approach to Tightening Semidefinite Relaxations for Certain Combinatorial Problems [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1904/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1904/
Directeurs de recherche: Advisors:	Miguel F. Anjos
Programme: Program:	Mathématiques de l'ingénieur

UNIVERSITÉ DE MONTRÉAL

A NOVEL APPROACH TO TIGHTENING SEMIDEFINITE RELAXATIONS FOR CERTAIN COMBINATORIAL PROBLEMS

ELSPETH ADAMS
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2015

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

A NOVEL APPROACH TO TIGHTENING SEMIDEFINITE RELAXATIONS FOR CERTAIN COMBINATORIAL PROBLEMS

présentée par : ADAMS Elspeth
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. LE DIGABEL Sébastien, Ph. D., président
M. ANJOS Miguel F., Ph. D., membre et directeur de recherche
M. PERRIER Michel, Ph. D., membre
Mme SOTIROV Renata, Ph. D., membre externe

DEDICATION

To all the voices that kept me sane...

ACKNOWLEDGEMENTS

First and foremost I would like to sincerely thank my supervisor Miguel Anjos for his support and guidance. It is hard to sum the past seven years up in a few words but you have helped me find the researcher within me and I'm infinitely grateful.

I also want to thank Franz Rendl and Angelika Wiegele for their insight and for inviting me to Klagenfurt for a week. I learned so much from our discussions and from seeing the problem from a different perspective.

To my family, your love, support and understanding with my unpredictable schedule is extremely appreciated. Juan Pablo, thank you for the adventures and the encouragement. Max, thank you for (literally) being by my side throughout this process.

To all my friends on the 5^{th} floor, thank you, merci and gracias. You're a great group of friends; the laughs, lunches, games, MATLAB tips and practice presentations made this thesis possible. I officially pass on my window seat to the next person in line.

Andrew, thank you for learning about triangle inequalities; spending many nights listening to code run; and for accepting a world you did not always understand. Words cannot describe how much your support, respect and unwavering belief in me and my math means to me. Knowing you are on my side makes everything easier.

Most importantly Andrew, thanks for using the word 'optimization' correctly!

RÉSUMÉ

Ce mémoire présente une nouvelle famille de coupes nommées contraintes polytopiques k-projection (kPPCs) qui peuvent être utilisées pour résoudre certains problèmes quadratiques binaires. Notamment les problèmes qui satisfont une propriété de projection pour les solutions réalisables sur un sous-graphe induit ont la même structure que les solutions faisables sur le graphe entier. Parmi ces problèmes se trouvent le problème max-cut et le problème d'ensemble stable (*stable set problem*).

Les coupes sont généralement des inégalités, cependant les kPPCs s'en distinguent par le fait qu'elles sont formées d'un ensemble d'inégalités. De plus, elle peuvent être définies pour un seul sous-graphe induit ou pour un ensemble de sous-graphes induits, et sont utilisées pour resserrer les relaxations en programmation semi-définie. Trois aspects des kPPCs sont examinés dans ce mémoire : une hiérarchie qui converge vers une formulation exacte, une formulation pour trouver la contrainte kPPC la plus violée, et l'amélioration de la borne supérieure (pour un problème de maximisation) d'une implémentation pratique de kPPCs pour le problème max-cut.

La relaxation SDP avec kPPCs forme une hiérarchie. Le $k^{\text{ème}}$ niveau de la hiérarchie est la relaxation SDP avec kPPCs pour tous les sous-graphes induits de taille k. Lorsque k augmente, l'intensité de la relaxation augmente également puisque $\text{CUT}_k \subseteq \text{CUT}_{k+1}$ où CUT_k est le polytope de coupe de taille k. Au $n^{\text{ème}}$ niveau, la formulation n'est plus une relaxation et rejoint exactement le problème d'origine CUT_n. Il existe $\binom{n}{k}$ sous-graphes induits uniques pour un graphe à n nœuds. Par conséquent, il n'est possible d'énumérer explicitement les niveaux de la hiérarchie que pour de petits exemples. Cependant, la force de la hiérarchie des kPPCs est que la matrice semi-définie positive, qui est variable dans la relaxation SDP, n'augmente pas en taille lorsque le niveau augmente, contrairement aux hiérarchies de Lasserre.

Pour un sous-graphe induit donné I, un modèle d'optimisation (nommé distance-au-polytope) est présenté pour déterminer si la solution optimale de la relaxation SDP viole les kPPCs pour I et, dans l'affirmative, pour quantifier la violation. Le modèle distance-au-polytope a une fonction objectif quadratique, des contraintes linéaires et se résout rapidement. La solution optimale est la distance euclidienne entre le mineur principal de la solution optimale

de la relaxation (X_I^*) et le polytope de coupe ($\text{CUT}_{|I|}$). Si la distance est égale à zéro, alors l'inclusion de kPPCs pour I dans la relaxation SDP ne resserrera pas la borne. Si la distance est strictement supérieure à zéro, alors les kPPCs pour I ne sont pas satisfaites par la solution courante. Par conséquent, leur inclusion dans la relaxation SDP changera la solution courante X^* (bien qu'une amélioration de la borne ne soit pas garantie).

Ce mémoire présente un modèle d'optimisation binaire-mixte dans un cône de second ordre (SOC) qui, pour un k donné, trouve la kPPC la plus éloignée du polytope de coupe. Le problème interne est le modèle distance-au-polytope. Le problème externe comporte des variables binaires qui prennent en compte tous les sous-graphes induits de taille k. Les problèmes à deux niveaux sont intrinsèquement difficiles à résoudre. Une reformulation est donc présentée qui change le problème à deux niveaux en un problème SOC équivalent à un seul niveau. La reformulation utilise des techniques telles que les conditions KKT, les contraintes disjointes et le saut de dualité. De plus, nous montrons comment renforcer le modèle à un seul niveau en incluant des contraintes de bris de symétrie et en incluant des variables binaires additionnelles qui réduisent la taille de l'arbre d'énumération. MOSEK est utilisé pour résoudre le problème et les résultats sont présentés jusqu'à la taille 20.

À chaque itération d'une méthode de plan sécant, une relaxation est résolue et, si un critère d'arrêt n'est pas atteint, une procédure de séparation cherche les coupes violées ou valides à ajouter à la relaxation. Ce mémoire présente un algorithme de plan sécant utilisant les kPPCs pour le problème max-cut. Notre méthode de plan sécant comporte 3 étapes. La première résout la relaxation SDP simple pour fournir une solution optimale initiale. La seconde résout itérativement la relaxation SDP simple à laquelle s'ajoute des inégalités triangulaires. À chaque itération, l'ensemble des inégalités triangulaires est composé, d'une part, de certaines inégalités triangulaires qui sont violées par la solution précédente et, d'autre part, des inégalités triangulaires actives de l'itération précédente. Les inégalités non actives ne sont pas saturées et ne sont par conséquent pas conservées. La troisième étape débute quand l'étape 2 n'apporte plus d'amélioration significative : des kPPCs sont ajoutées au modèle (relaxation SDP simple avec inégalités triangulaires fournies par la dernière itération de l'étape 2). Pour trouver les kPPCs violées, la procédure de séparation résout le problème distance-au-polytope pour les indices générés à partir des inégalités triangulaires violées. Cette méthode donne de meilleurs résultats que la sélection aléatoire des sous-graphes induits pour en tester la violation. En particulier, nous montrons que davantage de kPPCs violées sont trouvées et que la violation est plus grande. Finalement, nous présentons des résultats numériques (pour $n = 500 - 1000$) montrant que, lorsque l'amélioration de la borne à partir d'inégalités

triangulaires est faible, les kPPCs sont encore capables de resserrer la relaxation.

ABSTRACT

This thesis introduces a new family of cuts called k-projection polytope constraints (kPPCs) that can be used to solve certain binary quadratic problems. Specifically those problems that satisfy a projection property in which feasible solutions on an induced subgraph have the same structure as feasible solutions on the full graph, such as the max-cut problem and the stable set problem.

Typically cuts (also called valid inequalities) are inequalities, however kPPCs differ as they are a set of equalities. Furthermore they can be defined for a single induced subgraph or a set of induced subgraphs and are used to tighten semidefinite programming (SDP) relaxations. Three aspects of kPPCs are examined in this thesis: a hierarchy that converges to an exact formulation, a formulation to find the most violated kPPC and a practical implementation of a cutting plane algorithm using kPPCs that improves the upper bound (of a maximization problem) for the max-cut problem.

The SDP relaxation with kPPCs forms a hierarchy. The k^{th} level of the hierarchy is the SDP relaxation with kPPCs for all induced subgraphs of size k. As k increases, the strength of the relaxation also increases since $\text{CUT}_k \subseteq \text{CUT}_{k+1}$ where CUT_k is the cut polytope of size k. At the n^{th} level the formulation is no longer a relaxation and defines the original problem, CUT_n, exactly. There are $\binom{n}{k}$ unique induced subgraphs for a graph with n vertices. Therefore explicitly producing the levels of the hierarchy is only possible for small examples. However the strength of the hierarchy of kPPCs is that the positive semidefinite matrix variable in the SDP relaxation does not grow in size as the level is increased. This is in contrast to other hierarchies including the Lasserre hierarchy.

For a given induced subgraph I, an optimization model (denoted distance-to-polytope) is presented to determine if the optimal solution to an SDP relaxation violates the kPPC for I and, if so, to quantify the violation. The distance-to-polytope model has a quadratic objective function, linear constraints and solves quickly. The optimal solution is the euclidean distance between the principal minor of the optimal solution to the relaxation (X^*_I) and the cut polytope ($\text{CUT}_{|I|}$). If the distance equals zero then including the kPPC for I in the SDP relaxation will not tighten the bound. If the distance is strictly greater than zero then the kPPC for I is not satisfied by the current solution. Therefore including it in the SDP

relaxation will change the current solution X^* (although a strict improvement in the bound is not guaranteed).

The maximally violated valid inequality problem (MVVIP) determines the valid inequality from a family of cuts that is most violated. This thesis examines this problem for kPPCs. Specifically we present a mixed-binary second order cone optimization model that, for a given k, finds the kPPC that is furthest from the cut polytope. The inner problem is the distance-to-polytope model. The outer problem includes binary variables that consider all induced subgraphs of size k. Bilevel problems are inherently hard to solve. A reformulation is presented that changes the bilevel model into an equivalent single level second order cone problem. The reformulation uses techniques such as KKT conditions, disjunctive constraints and the duality gap. Moreover we show how to strengthen the single level model by including symmetry breaking constraints and including additional binary variables that reduce the size of the enumeration tree. MOSEK is used to solve the problem and results are presented up to size 20.

At each iteration of a cutting plane method a relaxation is solved and if a stopping criteria is not met a separation procedure looks for violated and valid cuts to add to the relaxation. This thesis presents a cutting plane algorithm using kPPCs for the max-cut problem. There are 3 stages in our cutting plane method. The first solves the basic SDP relaxation to give an initial optimal solution. The second stage iteratively solves the basic SDP relaxation plus some triangle inequalities. At each iteration the set of triangle inequalities is composed of some triangle inequalities that are violated by the previous solution and the triangle inequalities from the previous iteration that are active. The non-active inequalities are not binding and therefore are not kept. When there are no more violated triangle inequalities (or the improvement has stalled) we begin the third stage in which kPPCs are added to the model (basic SDP relaxation plus triangle inequalities from the last iteration of stage 2). The separation procedure to find violated kPPCs solves the distance-to-polytope problem for indices generated from violated triangle inequalities. Compared to randomly selecting induced subgraphs to test for violation, generating them from the indices used in triangle inequalities gives better results. Specifically we show that more violated kPPCs are found and that the amount of violation is larger. Finally we examine dense graphs of size 500 to 1000 and present computational results showing that kPPCs are able to improve the bound even after triangle inequalities can no longer tighten the relaxation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

kPPC	k Projection Polytope Constraint
SDP	Semidefinite Programming
MVVIP	Maximally Violated Valid Inequality Problem
CPM	Cutting Plane Method
CUT_n	cut polytope of size n
C_r	cut matrix
$\mathcal{C}$	correlation matrix
$\mathcal{M}$	metric polytope

LIST OF APPENDIXES

CHAPTER 1 INTRODUCTION

Many important combinatorial problems are NP-hard to solve. Due to the inherent difficulty of these problems finding tighter convex relaxations that are tractable is of great interest. Semidefinite programming (SDP) relaxations often produce strong bounds for these combinatorial optimization problems. Surveys by Goemans (1997) and Lovász (2003) outline the connection between SDP and NP-hard problems. SDP relaxations exist for a variety of such problems and many ways to tighten them have been proposed, e.g. see the book of Anjos and Lasserre (2012).

Let $\mathcal{F} := \{X_1, \ldots\}$ be the set of feasible solutions to a combinatorial problem and the polyhedron $\mathcal{P} := \text{conv}\{\mathcal{F}\}$ be the convex hull of the feasible solutions. Then the combinatorial optimization problem (P) is $\max\{f(X) : X \in \mathcal{F}\}$ where $f(X)$ is a convex function defined on the variable X and the relaxed optimization problem (R) is $\max\{f(X) : X \in \mathcal{R}\}$ where $\mathcal{P} \subseteq \mathcal{R}$. Problem (R) is called a *relaxation* of problem (P) and is assumed to also be a polyhedron. Relaxations provide bounds for optimal solutions (upper bounds for maximization problems) and are critical components of many algorithms used to solve integer programs such as branch-and-bound, branch-and-cut and cutting plane methods.

The key features of the latter two algorithms are valid inequalities or 'cuts'. These are additional constraints (typically inequalities) that tighten the relaxation. In this thesis we will use the word 'cut(s)' since the new family of cuts we will define (called k-projection polytope constraints) is not an inequality. A family of cuts is a group or class of cuts that shares a special (and/or similar) structure. The separation problem determines 'good' cuts (from a family of cuts) that will tighten the relaxation.

This thesis introduces the new family of cuts called k-projection polytope constraints (kPPCs). Specifically three concepts related to kPPCs will be examined:

1. the construction of a hierarchy of relaxations using kPPCs,
2. the formulation of a model to solve the maximally violated valid inequality problem (MVVIP) for kPPCs and
3. the impact that kPPCs have on the bound in a cutting plane method for large dense max-cut problems.

This thesis is organized as follows: the remainder of this chapter is devoted to introducing kPPCs for the max-cut problem. Chapter 2 covers an in-depth literature review on the max-cut problem and the three themes in this thesis: hierarchies, MVVIP and cutting plane

methods. Then each of these themes is examined in turn (Chapters 3, 4 and 5 respectively). Finally Chapter 6 summarizes the theory and results related to kPPCs.

1.1 Notation

We use the following notation throughout this thesis. Let $\mathcal{S}_n$ be the set of symmetric matrices of size n ; e is the vector of all ones of the appropriate size and let X be a matrix of size $n \times n$ and let X_{ij} denote the element in matrix X at row i and column j. For matrices A,$B \in \mathcal{S}_n$, $\langle A, B\rangle := \text{tr}(AB)$ is the matrix inner product and $\text{diag}(A)$ is a vector of the elements on the diagonal of matrix A. Conversely, $\text{Diag}(a)$ is a matrix with the elements of the vector a on the diagonal of an $|a| \times |a|$ matrix and zeros on the off-diagonal. Finally we define the laplacian matrix as $L := \text{Diag}(We) - W$.

1.2 k-projection polytope constraints

This section introduces the family of cuts called kPPCs specifically for the max-cut problem. Details will not be presented here but are instead contained in Section 3.3.1.

Consider the following exact formulation of the max-cut problem:

$$z_{\text{MC}} = \max\{\langle L, X\rangle : \ X \in \{\text{diag}(X) = e, \ X \in \mathcal{S}_n, \ \text{rank}(X) = 1\}$$

and the following initial relaxation:

$$\max\{\langle L, X\rangle \colon \text{diag}(X) = e, \ X \in \mathcal{S}_n\}. \tag{1.1}$$

The definition of kPPCs for the subset I where $|I| = k$ is:

$$C\lambda = \text{triu}(X_I), \quad \sum_{i=1}^{2^{k-1}} \lambda_i = 1, \quad \lambda \geq 0 \tag{1.2}$$

where $\text{triu}(X_I) = \begin{bmatrix} X_{i_1,i_2} & X_{i_1,i_3} & X_{i_2,i_3} & \dots & X_{i_{k-1},i_k} \end{bmatrix}^T$ is the strictly upper triangular part of matrix X_I indexed by $I = (i_1, i_2, \dots, i_k)$, the variable λ is a vector of length 2^{k-1} and $C := c_r c_r^T$ with $c_r \in \{-1, 1\}^k$ denotes the 2^{k-1} feasible max-cut solutions on k vertices.

This is equivalent to:

$$X_I \in \text{CUT}_k$$

which requires X_I to be in the convex hull of the vertices of CUT_k where CUT_k is the cut polytope of size k. By definition, the cut polytope of size n is the convex hull of feasible max-cut solutions on a graph with n vertices.

Including kPPCs in a max-cut relaxation such as (1.1) will tighten the relaxation. This thesis addresses questions related to including kPPCs to tighten a relaxation such as which ones to include and how to find them. In addition we examine why kPPCs work for the max-cut problem and generalize kPPCs to more general combinatorial problems. Finally computational results are presented showing that kPPCs can tighten max-cut relaxations. We focus on large ($n = 500$ to 1000), dense examples.

CHAPTER 2 CRITICAL LITERATURE REVIEW

The literature review will cover the following topics in turn: the max-cut problem, hierarchies of relaxations, exact separation of violated cuts and cutting plane algorithms.

2.1 The max-cut problem

Let $G = (V, E)$ be an undirected graph with no loops where $V = \{1, \ldots, n\}$ is the vertex set and the set of edges is E. For every edge $ij \in E$ we define a weight w_{ij} and denote the weighted adjacency matrix A such that $(A)_{ij} = w_{ij} \forall ij \in E$. A *cut* is defined as a partitioning of the vertices into two sets denoted S and $V \setminus S$. The *cut weight* of partition S is

$$\sum_{i \in S, j \notin S} w_{ij}.$$

The max-cut problem is to find the partition that maximizes the cut weight. The applications of the max-cut problem include circuit layout design and the ising spin problem in statistical physics (Barahona, Grötschel, Jünger, and Reinelt (1988), Lengauer (1990), Liers, Jünger, Reinelt, and Rinaldi (2004)).

2.1.1 Exact max-cut formulations

There are two formulations for the max-cut problem: vertex based and edge based. The vertex based approach determines if vertex i is in the set S or not while the edge based approach partitions the vertices by determining if the endpoints of an edge are in the same set or not. The edge-based max-cut formulation is used in this thesis, however we present both approaches.

Vertex based formulation

Let $x \in \{-1, 1\}^n$ such that

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases} \quad \forall i \in V.$$

The *Laplacian* is defined as $L := \text{Diag}(Ae) - A$ and can be used to define the cut weight :

$$\frac{1}{4}x^T Lx = \sum_{i \in S, j \notin S} w_{ij}.$$

Then the vertex based max-cut formulation is:

$$\text{(MC)} \qquad z_{\text{MC}} = \max\{\tfrac{1}{4}x^T Lx : \ x \in \{-1,1\}^n\}. \tag{2.1}$$

This formulation is a quadratic binary problem.

Edge based formulation

Recall that the edge based max-cut formulation determines if the endpoints of an edge are in the same set or not. If vertices i and j are in different partitions then we say that the edge ij is cut. Specifically,

$$X_{ij} := \begin{cases} 1 & \text{if } i,j \in S \text{ or } i,j \notin S \\ -1 & \text{otherwise} \end{cases} \qquad \forall ij \in E.$$

Therefore

$$X = xx^T \quad \text{and} \quad \frac{1}{4}x^T Lx = \langle L, X \rangle.$$

Furthermore the 2^{n-1} feasible solutions to the max-cut problem can be written as the vector $c_i \in \{-1,1\}^n$ or as a *cut matrix* C_i where $C_i = c_i c_i^T$. The term 'cut' in cut matrix refers to feasible max-cut solutions. The *cut polytope*, $\text{CUT}_n := \{C_i \ : \ \forall i = 1, \ldots, 2^{n-1}\}$, is the convex hull of all 2^{n-1} feasible solutions C_i.

With this notation the edge-based max-cut formulation is:

$$\text{(MC)} \qquad z_{\text{MC}} = \max\{\langle L, X \rangle : \ X \in \text{CUT}_n\}. \tag{2.2}$$

This is equivalent to the formulation given in (2.1).

The max-cut problem is known to be NP-complete (Karp, 1972). However certain classes of graphs can be solved in polynomial time (Poljak and Tuza, 1995). Results of the cut polytope are extensively studied in the book by Deza and Laurent (1997a). This includes a complete list of the facets defining inequalities for the cut polytope of size up to 8. Other work on

the cut polytope includes Barahona (1993), Poljak and Tuza (1994) and Deza, Laurent, and Poljak (1992).

2.1.2 Max-cut relaxations

A survey of max-cut relaxations can be found in Hansen (1979) (prior to 1980) and more recently in Wiegele (2006). Linear programming relaxations (for example Barahona and Ladanyi (2006) ; Barahona, Jünger, and Reinelt (1989) ; and Liers, Jünger, Reinelt, and Rinaldi (2004)) and semidefinite programming relaxations have both been extensively studied. We focus on SDP relaxations as that is the basis of the relaxation presented in this thesis. Poljak and Rendl (1995) introduced the primal version of the basic SDP relaxation and the strengthened SDP relaxation over the metric polytope. Delorme and Poljak (1993) presented the dual version. The following sets are used in the two most common SDP relaxations.

The set $\mathcal{C}$ of *correlation matrices* is:

$$\mathcal{C} := \{X \in \mathcal{S}_n : \ \operatorname{diag}(X) = e,\ X \succeq 0\}$$

and the *metric polytope* $\mathcal{M}$ is the set of all symmetric matrices with diagonal equal one and satisfying the triangle inequalities,

$$\mathcal{M} := \{X \in \mathcal{S}_n : \ \operatorname{diag}(X) = e,\ X_{ij} + X_{ik} + X_{jk} \geq -1, X_{ij} - X_{ik} - X_{jk} \geq -1 \ \forall i, j, k \in V\}.$$

For $n = 3$ or 4 the set of $4\binom{n}{3}$ triangle inequalities defines the cut polytope of size n. Additional inequalities (or an inner description using extreme points) is required to define the cut polytope for $n > 4$. The idea behind the triangle inequalities is that on each clique of size 3 either two edges or no edges are cut.

These two sets yield two popular semidefinite optimization relaxations of the max-cut problem:

$$z_{\mathcal{C}} := \max\{\langle L, X\rangle : \ X \in \mathcal{C}\} \tag{2.3}$$

$$z_{\mathcal{C}\cap\mathcal{M}} := \max\{\langle L, X\rangle : \ X \in \mathcal{C} \cap \mathcal{M}\} \tag{2.4}$$

Furthermore,

$$z_{\mathrm{MC}} \leq z_{\mathcal{C}\cap\mathcal{M}} \leq z_{\mathcal{C}}$$

since $\mathrm{CUT}_n \subseteq \mathcal{C} \cap \mathcal{M} \subseteq \mathcal{C}$.

The major theoretical result from Goemans and Williamson (1995) showed that the gap between the exact max-cut formulation and the SDP relaxation is bounded, specifically,

$$\frac{z_{\mathcal{C}}}{z_{MC}} \leq 1.1382.$$

There are $4\binom{n}{3}$ triangle inequalities. An interior-point method for solving this relaxation is provided in Helmberg, Rendl, Vanderbei, and Wolkowicz (1996). This semidefinite optimization problem can be solved in polynomial time up to a fixed prescribed precision. However it contains $O(n^3)$ inequality constraints, and hence it is a challenge for standard SDP solvers. A computationally efficient way to deal with this relaxation was introduced by Fischer, Gruber, Rendl, and Sotirov (2006). It combines interior-point methods with the bundle method to deal with the triangle inequalities. Exact methods where this relaxation is used in a branch-and-bound setting have been proposed by Rendl, Rinaldi, and Wiegele (2010) and Krislock, Malick, and Roupin (2014).

2.1.3 Binary quadratic program

The most basic binary quadratic programming problem that we will study has the following form:

$$\min\{y^T Q y + c^T y \ : \ y \in \{0,1\}^n\}$$

where Q is a symmetric matrix of order n and c is a vector of length n.

It can be shown that the binary quadratic problem is equivalent to the max-cut problem (for instance see Barahona, Jünger, and Reinelt (1989)). The following outlines the equivalence between the binary quadratic problem and the max-cut problem.

Let

$$x = 2y - e, \bar{x} = \begin{bmatrix} 1 \\ x \end{bmatrix} \text{ and } L = -\frac{1}{4}\begin{bmatrix} (Qe+2c)^T e & (Qe+c)^T \\ (Qe+c) & Q \end{bmatrix}$$

Using these definitions the following shows that the objective function of the binary quadratic

problem and the max-cut problem are equal:

$$\begin{aligned}
\bar{x}^T L \bar{x} &= \begin{bmatrix} 1 \\ 2y - e \end{bmatrix}^T \frac{1}{4} \begin{bmatrix} (Qe + 2c)^T e & (Qe + c)^T \\ (Qe + c) & Q \end{bmatrix} \begin{bmatrix} 1 \\ 2y - e \end{bmatrix} \\
&= \frac{1}{4} \Big((Qe + 2c)^T e + 2(Qe + c)^T (2y - e) + (2y - e)^T Q (2y - e) \Big) \\
&= \frac{1}{4} \Big(4c^T y + 4y^T Q y \Big) \\
&= y^T Q y + c^T y
\end{aligned}$$

2.1.4 Leading solvers

There are two main solvers for binary quadratic problems: BiqCrunch and BiqMac. We consider them in turn.

BiqCrunch

BiqCrunch was developed by Krislock, Malick, and Roupin (2014). It is a branch-and-bound algorithm for binary quadratic problems including max-cut that introduces an improved bounding procedure. The algorithm uses the following regularized problem:

$$\begin{aligned}
(SDP)^{\alpha}_{\triangle} \qquad \max \quad & \langle Q, X \rangle + \frac{\alpha}{2} \Big(n^2 - ||X||_F^2 \Big) \\
\text{s.t.} \quad & \operatorname{diag}(X) = e \\
& X \succeq 0 \\
& \mathcal{A}_{\triangle}(X) \leq -e
\end{aligned}$$

and the following bound that is defined using the dual function of $(\text{SDP})^{\alpha}_{\triangle}$:

$$F^{\alpha}_{\triangle}(y, z) := \max_{X \succeq 0} \Big(\langle Q - \operatorname{Diag}(y) + \mathcal{A}^*_{\triangle}(z), X \rangle - \frac{\alpha}{2} ||X||_F^2 + e^T T y + e^T z + \frac{\alpha}{2} n^2 \Big)$$

where $\mathcal{A}_{\triangle}(X) \leq -e$ defines the triangle inequalities for the set $\triangle$.

Furthermore, the following relationship holds between the optimal objective value of the

aforementioned problems:

$$z_{(MC)} \leq z_{(SDP)_{\triangle}} \leq z_{(SDP)^{\alpha}_{\triangle}} \leq F^{\alpha}_{\triangle}(y, z)$$

where $z(*)$ is the optimal objective value of model $(*)$ and $(SDP)_{\triangle}$ is the SDP relaxation with triangle inequality set $\triangle$. Note that the first inequality is from the relaxation of the original (MC) problem and the third inequality is due to weak duality. The second inequality is less obvious and is derived from the following fact (see Krislock et al. (2014) for the proof):

$$\text{if } X \succeq 0 \text{ and } \text{diag}(X) = e \text{ then } n \leq ||X||_F^2 \leq n^2.$$

Therefore in $(\text{SDP})^{\alpha}_{\triangle}$ the regularized term is always non-negative for $\alpha \geq 0$.

The parameter α penalizes the loss of tightness in the bound and the parameter ϵ controls the tolerance of the quasi-Newton method. Both are reduced when the number of violated triangle inequalities is small. Adjusting these parameters dynamically changes the set of triangle inequalities and is used to efficiently give improved bounds. The bounding procedure is incorporated into a branch-and-bound.

BiqMac

BiqMac was developed by Rendl, Rinaldi, and Wiegele (2010). It is a branch-and-bound algorithm in which the bounding procedure is a dynamic version of the bundle method and is used to solve semidefinite max-cut relaxations and can find exact solutions for instances of size up to $n = 100$.

The disjunction for the branching decision is if two vertices i and j are 'joined' (i and j are on the same side of the partition) or 'split' (i and j are on different sides of the partition). Additional procedures include a bounding procedure to determine upper bounds ; a heuristic to produce lower bounds by finding feasible solutions and a branching strategy to determine the next i-j pair to be branched on.

Adding all $4\binom{n}{3}$ triangle inequalities is only tractable for very small problems therefore the number of triangle inequalities added at each iteration is limited. The number of inequalities to add is dynamically chosen using a bundle method. Inequalities are removed when it is unlikely that they are active at the optimal solution. Non-active constraints have a dual multiplier equal to 0. Therefore triangle inequalities with dual multiplier close to 0 are

removed.

2.2 Hierarchies

Semidefinite programming relaxations that provide strong bounds for combinatorial problems are of great interest. Several hierarchies of relaxations have been proposed that provide increasingly tighter bounds. These include the Sherali and Adams (1990) reformulation-linearization technique (RLT); the Lovász and Schrijver (1991) lift-and-project ; the Lasserre (2002) hierarchy ; and a lifting by Anjos and Wolkowicz (2002). For combinatorial problems these hierarchies have the property that they converge to the global optimal solution in a finite number of steps ; however they have the computational drawback that the numbers of variables at each level grows exponentially. Even the first nontrivial lifting step leads to matrices of order $\binom{n}{2}$ which is prohibitive even for very modest values of n such as $n \approx 50$. We examine the lifting of Anjos and Wolkowicz and the Lasserre hierarchy in turn.

2.2.1 Lifting of Anjos and Wolkowicz

Anjos and Wolkowicz (2002) present two strengthened SDP relaxations for the max-cut problem. The first of the two is:

$$
\begin{array}{lll}
\text{(SDP2)} \qquad v_2^* = \max\limits_{Y} & \operatorname{tr} H_Q Y & \\
\text{s.t.} & \operatorname{diag}(Y) = e & \\
& Y_{0,t(i)} = 1 & \forall i = 1, \ldots, n \\
& Y_{0,T(i,j)} = \frac{1}{n} \sum\limits_{k=1}^{n} Y_{T(i,k),T(k,j)} & \forall 1 \leq i < j \leq n \\
& Y \succeq 0, \;\; Y \in \mathcal{S}_{t(n)+1} &
\end{array}
$$

where

$$
T(i,j) := \begin{cases} t(j-1)+i & \text{if } i \leq j \\ t(i-1)+j & \text{otherwise} \end{cases}, \qquad H_Q := \begin{bmatrix} 0 & \frac{1}{2}\operatorname{dsvec}(Q)^T \\ \frac{1}{2}\operatorname{dsvec}(Q) & 0 \end{bmatrix},
$$

$t(i) = T(i,i) \;\; \forall i = 1, \ldots, n$ and $\operatorname{dsvec}(*)$ forms the $t(n)$-vector by taking the elements of an $n \times n$ matrix columnwise while ignoring the strictly lower triangular part of the matrix and multiplying the off-diagonal elements by 2.

(SDP2) is constructed by adding redundant quadratic constraints ; taking the lagrangian dual of the lagrangian dual ; and then removing the redundant constraints and projecting the

feasible set. The redundant constraints that are added to the basic SDP relaxation (2.3) are: $X \circ X = E$ and $X^2 - nX = 0$ where $\circ$ is the Hadamard (elementwise) product of matrices and E is the matrix of all ones.

The second SDP relaxation presented by Anjos and Wolkowicz is:

$$\begin{array}{lll} \text{(SDP3)} \qquad v_2^* = \max & \text{trace} H_Q Y & \\ \text{s.t.} & \text{diag}(Y) = e & \\ & Y_{0,t(i)} = 1 & \forall i = 1, \ldots, n \\ & Y_{0,T(i,j)} = Y_{T(i,k),T(k,j)} & \forall k,\ \forall 1 \leq i < j \leq n \\ & Y \succeq 0,\ \ Y \in \mathcal{S}_{t(n)+1} & \end{array}$$

and is a strengthening of (SDP2). It comes from adding the additional redundant constraints $X_{ij} = X_{ik}X_{kj}\ \forall 1 \leq i, j, k \leq n$.

2.2.2 Lasserre hierarchy

Consider the optimization problem:

$$\text{(Lass-P)} \qquad p^* = \inf_x \{f(x) : x \in \mathbb{K}\}$$

for some function $f : \mathbb{R}^n \to \mathbb{R}$ and for the basic semi-algebraic set:

$$\mathbb{K} := \{x \in \mathbb{R}^n : g_j(x) \geq 0,\ j = 1, \ldots, m\}$$

where $(g_j) \subset \mathbb{R}[x]$, $j = 1, \ldots, m$ are polynomials. Let $v := \max_{k=1,\ldots,m} \nu_k$ where the degree of the polynomial $g_j(x)$ is $2\nu_k$ or $2\nu_k - 1$ depending on the parity.

Note that problem (Lass-P) is not necessarily convex. For example if $g_j(x) = x_j^2 - x_j \geq 0, \forall j = 1, \ldots, n$ and $g_{j+n}(x) = x_j - x_j^2 \geq 0, \forall j = 1, \ldots, n$ then (Lass-P) defines a general 0-1 program, which is clearly not convex. The following semidefinite program (Lass-Q) (which is a function of d) is a relaxation of (Lass-P):

$$\begin{array}{lll} \text{(Lass-Q)} \qquad q_d = \min_y & L_y(f) & \\ \text{s.t.} & M_d(y) \succeq 0 & \\ & M_{d-v_j}(g_j y) \succeq 0, & j = 1, \ldots, m \\ & y_0 = 1 & \end{array}$$

The model is constructed by defining variables $y := (x^\alpha), \alpha \in \mathbb{N}^n_{2d}$; an objective function $L_y(f)$ such that $L_y(f) = f(x) \; \forall$ feasible y, and by constructing two matrices that act as the constraints. The matrices have the form:

$$M_d(y) = \sum_{\alpha \in \mathbb{N}^n_{2d}} y_\alpha B_\alpha \qquad \text{and} \qquad M_{d-v_j}(g_j y) = \sum_{\alpha \in \mathbb{N}^n_{2d}} y_\alpha C^j_\alpha, \; j = 1, \ldots, m$$

where B_α and C^j_α are real symmetric matrices such that:

$$v_d(x) v_d(x)^T = \sum_{\alpha \in \mathbb{N}^n_{2d}} x^\alpha B_\alpha \qquad \text{and} \qquad g_j(x) v_{d-v_j}(x) v_{d-v_j}(x)^T = \sum_{\alpha \in \mathbb{N}^n_{2d}} x^\alpha C^j_\alpha.$$

The matrix variable $M_d(y)$ is induced from the support vector $v = (x^\alpha), \alpha \in \mathbb{N}^n_{2d}$. For $d = n$ the support vector has the following form:

$$v := [1, x_1, \ldots, x_n, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n, x_1x_2x_3, \ldots, \prod_{i=1}^{n} x_i] = [x^\alpha], \forall \alpha \in \mathbb{N}^n_{2d}$$

For all $d \in \mathbb{N}, (Lass - Q)_d$ is a relaxation of (Lass-P) since

1. $q_d \leq p^*$
2. the constraints of (Lass-Q) are necessary conditions for y to be the support vector of $\mathbb{K}$.

In addition the associated feasible sets are nested within one another with $(Lass - Q)_{d+1} \subseteq (Lass - Q)_d$ with each subsequent program as least as strong as the previous one. As a result the sequence of semidefinite programs $(Lass - Q)_d$, $d \in \mathbb{N}$ form a hierarchy. Lasserre (2001) then showed that the following property holds under certain assumptions on $\mathbb{K}$:

$$\inf (Lass - Q)_d \uparrow p^* \;\text{ as } d \to \infty$$

Now we will consider a special case of the general optimization problem in which all variables are binary. The general quadratic binary optimization problem as studied by Lasserre is defined as follows:

$$\begin{aligned} \text{(QBO)} \qquad p^* = \inf_x \quad & f(x) \\ \text{s.t.} \quad & g_j(x) \geq 0 \quad j = 1, \ldots, m \\ & x_i^2 = x_i \qquad i = 1, \ldots, n \end{aligned}$$

The binary condition $x_i \in \{0, 1\}$ can be written as the quadratic constraint $x_i^2 - x_i = 0, \forall i = 1, \ldots, n$. By expressing these equations as two inequalities (QBO) can be written in exactly the same format as the general optimization problem (Lass-P). The sos-moment approach of creating the Lasserre hierarchy naturally follows. When the Lasserre hierarchy is applied to the quadratic binary optimization problem three special properties emerge:

1. All monomials can be simplified to monomials in which each term has degree 1 or 0. For example $x_1^2 x_2^3 = x_1 x_2$. This is a direct result from the fact that $x_i^2 = x_i \, \forall i = 1, \ldots, n$.
2. Identical monomials can be eliminated from the support vector without weakening the relaxation.
3. There is a finite number of distinct monomials to include in the support vector, since each distinct monomial has terms of degree 1 or 0. Therefore there exists at most $2^n - 1$ monomials.

Furthermore the matrix variable constructed from the support vector $X = vv^T$ can be simplified to only include monomials of degree 1 or 0 and as a result X contains at most $2^n - 1$ variables. We will now state the main result of Lasserre, the proof has been omitted:

Theorem 1 (Lasserre Theorem (Lasserre, 2002)) *Let (QBO) be defined as above and consider the hierarchy of semidefinite programs (Lass-Q). Then for every $i \geq n + \nu$*

1. *$(Lass - Q)_i$ is solvable with $p^* = \min(Lass - Q)_i$*
2. *Each optimal solution x^* of (QBO) corresponds to the following optimal solution of $(Lass - Q)_i$:*

$$y^* := \left(x_1^*, \ldots, x_n^*, \ldots, (x_1^*)^{2i}, \ldots, (x_n^*)^{2i}\right)$$

3. *Each optimal solution y^* of $(Lass - Q)_i$ is the (finite) vector supported on s optimal solutions of (QBO) with*

$$s = \text{rank}(M_i(y)) = \text{rank} M_n(y)$$

2.3 Exact separation

Separation procedures (or constraint identification problems) are defined as follows: given a point x and a family of valid constraints $\mathcal{L}$, identify one or more constraints in $\mathcal{L}$ violated by x, or prove that no such constraint exists (Padberg and Rinaldi, 1991). Note that although separation procedures are typically used to identify inequalities, the framework is identical for any set of valid constraints.

Separation procedures have been studied from both the practical and theoretical perspectives and are often discussed in the context of cuts which are used to tighten relaxations. Cuts that share a special structure can be categorized into a specific family or class. Applegate, Bixby, Chvàtal, and Cook (2006) called the paradigm of generating cuts from a given family the template paradigm. Different types of cuts include Chvátal cuts (Chvátal, 1973), Chvátal-Gomory (Nemhauser and Wolsey (1988), Bonami, Cornuéjols, Dash, Fischetti, and Lodi (2008)), $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts (Caprara and Fischetti, 1996), split cuts (Cook, Kannan, and Schrijver, 1990), MIR-inequalities (Nemhauser and Wolsey, 1988) and lift-and-project cuts (Balas, Ceria, and Cornuéjols, 1993).

Practically, the separation of valid inequalities (i.e. cuts) is a key component of cutting plane algorithms. Cutting plane algorithms have been well studied and are fundamental in solving integer (and mixed integer) optimization problems. For early research see Dantzig, Fulkerson, and Johnson (1954), Gomory (1963) and Grötschel, Lovász, and Schrijver (1981). For more recent advances see Mitchell (2002) and Marchand, Martin, Weismantel, and Wolsey (2002).

Different separation procedures are used for different families of cuts. Caprara and Letchford (2003) examined the complexity of the separation procedure for various inequalities. They proved strong NP-completeness for the separation of split cuts and strengthened NP-completeness results for $\{0, \frac{1}{2}\}$-cuts (initially in Caprara and Fischetti (1996)) and Chvàtal-Gomory cuts (initially in Eisenbrand (1999)). Matrix cuts and lift-and-project cuts have been shown to be separable in polynomial time (Balas, Ceria, and Cornuéjols (1993), Lovász and Schrijver (1991)).

2.3.1 Maximally violated inequalities

Optimization models have been proposed that look for maximally violated cuts. Caprara, Fischetti, and Letchford (2000) proposed a model that finds the mod-k cut that is maximally violated for a given point x^*. They also show that for any given k for which a prime factorization is known maximally violated mod-k cuts can be found efficiently in $O(mn \min\{m, n\})$ time.

Lodi, Ralphs, and Woeginger (2012) proposed a mixed-integer bilevel model for a general separation problem which finds the maximally violated valid inequality. They emphasize the conceptual nature of this formulation as challenges exist to explicitly write a compact description of the inner problem in addition to the practical issues surrounding solving bilevel problems. However for certain examples (split cuts and generalized subtour elimination constraints (GSECs) for the capacitated vehicle routing problem) the bilevel model can be converted to a single level linear optimization problem. Two key components in MVVIPs are

validity and membership. We address them in turn.

The validity verification problem determines if all points in a polyhedron satisfy the constraint. Lodi et al. (2012) formalize this concept for linear inequalities. For a given polyhedron $\mathcal{P} = \{x \in \mathcal{R}^n_+ \mid Ax \geq b\}$, (α, β) defines a valid inequality if and only if there exists $u \in \mathcal{R}^m_+$ such that $\alpha \geq u^T A$ and $\beta \leq u^T b$. The inequality $\alpha^T u \geq \beta$ is valid for the polyhedron $\mathcal{P}$ since all $x \in \mathcal{P}$ satisfy $\alpha^T x \geq \beta$.

The membership problem is a decision problem that asks whether a given point $\hat{x}$ is contained in a polyhedron $\mathcal{P}$ or the intersection of the polyhedron $\mathcal{P}$ and a given cut. This problem has been looked at in the context of many different families of cuts, for example Gomory-Chvátal cuts (Eisenbrand, 1999) and $\{0, \frac{1}{2}\}$ cuts (Caprara and Fischetti, 1997).

2.4 Cutting plane algorithms

Dantzig, Fulkerson, and Johnson (1954) presented the first cutting plane algorithm. They applied the procedure to a traveling salesman problem with 54 cities. Gomory (1965) later generalized cutting plane methods for general integer programming problems. Survey papers include Helmberg and Rendl (1998), Mitchell (2003), Krishnan and Terlaky (2005) and Anjos and Vannelli (2008).

Cutting plane methods are algorithms that repetitively solve relaxations, each time reducing the feasible region. By reducing the feasible region in a special way so that the previous optimal solution is not included and the new relaxation has a different optimal solution. Relaxations act as upper bounds to the objective value of a maximization problem. As relaxations are tightened the upper bound is strengthened. Although numerous cutting plane methods exist they generally have the following components:

Original problem This is an exact formulation of the problem that is too difficult to solve directly.

Relaxations The first relaxation is often the most general. This would include, but is not limited to, the LP relaxation of an integer program or the relaxation of the constraint $\text{rank}(X) = 1$ in an SDP model.

Tightening method This process iterates between creating a tighter relaxation and solving this tighter relaxation. The hope is that the optimal objective value of the relaxation will improve thus improving the upper bound. The goal is to create a tighter relaxation in such a way that the optimal solution of the previous model is no longer feasible. This will ensure that the optimal solution will change, although it does not guarantee that the optimal objective value will improve.

Separation procedure For some family of cuts a separation procedure finds valid and violated cuts that can be used to tighten the relaxation.

Stopping Criteria The stopping criteria indicates when the iterative process mentioned above stops. Ideally this is when the optimal solution to the current relaxation can be proven to be the optimal solution to the original problem.

Closely related to cutting plane methods is branch-and-cut algorithms. These algorithms combine branch-and-bound procedure while applying a cutting plane method at each node of the enumeration tree. A survey of branch-and-cut algorithms specifically for combinatorial problems is given in the chapter by Mitchell (2002).

2.4.1 Valid inequalities

Consider a combinatorial problem and let $\mathcal{F} = \{X_1, \ldots\} \subseteq S_n$ be the set of feasible solutions where S_n is the set of symmetric matrices of size n. We can define the polytope $\mathcal{P} := \text{conv}\{X_1, \ldots\}$ as the convex hull of the set of feasible solutions. Let $\mathcal{R}^i$ be some polyhedral relaxation for the polytope $\mathcal{P}$ and let X^* denote the optimal solution and z^* the optimal objective value obtained by solving $\mathcal{R}^i$. To tighten the relaxation $\mathcal{R}^i$ we can find a polyhedron $\mathcal{T}$ such that $X^* \notin \mathcal{T}$ and $\mathcal{P} \subseteq \mathcal{R} \subseteq \mathcal{T}$.

Often a set of hyperplanes (i.e. valid inequalities) are used to tighten the relaxation. Hyperplanes can be used to tighten linear programs and SDPs. In both cases a new, tighter relaxation $\mathcal{R}^{i+1} := \mathcal{R}^i \cap \mathcal{T}$ can be defined. In either case this process can be iteratively repeated such that tighter and tighter relaxations are found $\mathcal{P} \subseteq \cdots \subseteq \mathcal{R}^{i+1} \subseteq \mathcal{R}^i$. The process stops when the polytope $\mathcal{P}$ is defined or a provably optimal solution is found.

2.4.2 Non traditional cuts

In this section we examine one example of a non traditional cut (i.e. a cut that does not define a halfspace). Belotti, Góez, Pólik, Ralphs, and Terlaky (2013) define cuts called disjunctive conic cuts for mixed integer second order cone optimization (MISOCO) problems. Although these cuts are inequalities they do not define halfspaces (as traditional cuts do) but cones. Consider the following MISOCO problem:

$$\min \left\{ c^T x \ : \ Ax = b, \ x \in \mathcal{K}, x \in \mathbb{Z}^d \times \mathbb{R}^{n-d} \right\}$$

where $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = m; c \in \mathbb{R}^n; b \in \mathbb{R}^m$ and $\mathcal{K}$ is the intersection of a set of second order cones.

Let $\mathcal{E} \subset \mathbb{R}^n$ with $n > 1$ be a full dimensional closed convex set and consider two halfspaces $\mathcal{A} = \{x \in \mathbb{R}^n \, : \, a^T x \geq \alpha\}$ and $\mathcal{B} = \{x \in \mathbb{R}^n \, : \, b^T x \leq \beta\}$. Then a closed convex cone $\mathcal{K} \subseteq \mathbb{R}^n$ with $\dim(\mathcal{K}) > 1$ is a *disjunctive conic cut* for $\mathcal{E}$ and the disjunctive set $\mathcal{A} \cup \mathcal{B}$ if

$$\text{conv}(\mathcal{E} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{E} \cap \mathcal{K}$$

Essentially for a feasible region, $\mathcal{E}$ and a disjunctive condition ($x \in \mathcal{A}$ or $x \in \mathcal{B}$) the disjunctive conic cut is a cone that reduces the feasible region of the relaxation so that both sides of the disjunction ($\mathcal{E} \cap \mathcal{A}$ and $\mathcal{E} \cap \mathcal{B}$) are still feasible. This tightens the relaxation without having to branch on the disjunctive condition.

Depending on the type of disjunction a cylinder may be used as opposed to a cone. These are called disjunctive cylindrical cuts and are defined in a similar way.

CHAPTER 3 HIERARCHY OF SEMIDEFINITE RELAXATIONS USING k-PROJECTION POLYTOPE CONSTRAINTS

3.1 Introduction

Semidefinite programming relaxations exist for many combinatorial problems and several hierarchies of relaxations have been proposed that provide increasingly tight bounds.

This chapter examines a new hierarchy of SDP relaxations composed of kPPCs. It is applied to the classes of NP-hard graph problems that satisfies the projection property (specifically the max-cut problem and the stable set problem).

The k^{th} level of the proposed hierarchy consists of the basic SDP relaxation for the particular problem plus kPPCs for every induced subgraph of size k. This hierarchy has the distinguishing feature that all the SDP relaxations are formulated in the space of the original SDP relaxation.

The size of the relaxations increases rapidly with k because of the large number of induced subgraphs that exist. For this reason we also briefly explore the possibility of adding the projection polytope constraints for only selected subgraphs (a more detailed exploration of this topic is the focus of Chapter 5).

We provide computational results showing that the proposed hierarchy yields improved bounds when compared to the basic SDP relaxations for benchmark instances of the max-cut and stable-set problems. We also report results showing that the improved bounds results in a significantly smaller enumeration tree when the SDP relaxation is used in a branch-and-bound scheme to solve the problems to optimality or near-optimality. We begin by generalizing the definition of kPPCs presented in the Chapter 1.

3.2 The projection property

For a general description of our hierarchy of SDP relaxations, assume that $V := \{1, \ldots, n\}$ denotes the vertex set of the graph of the combinatorial problem at hand and that the combinatorial optimization problem under consideration is given through its set of feasible solutions $\mathcal{F} = \{X_1, \ldots\} \subseteq \mathcal{S}_n$, where $\mathcal{S}_n$ denotes the set of symmetric matrices of order n. Furthermore let $\mathcal{P}$ denote the convex hull of all feasible points, i.e. $\mathcal{P} = \text{conv}\{X_1, \ldots\}$.

Given a cost matrix C, the optimization problem is to find the solution $X \in \mathcal{P}$ that maxi-

mizes $\langle C, X\rangle$:

$$z_{\mathcal{P}} := \max\{\langle C, X\rangle \colon X \in \mathcal{P}\}.$$

For a set $I \subseteq V$, let

$$\pi_I(X) = X_I$$

denote the (orthogonal) projection mapping X to X_I (i.e. the principal submatrix of X indexed by I). Similarly

$$\pi_I(\mathcal{P}) = \text{conv}\{\pi_I(X_1), \ldots\}$$

denotes the projection of $\mathcal{P}$ onto I. We are particularly interested in problems for which $\pi_I(\mathcal{P})$ has a "simple" description. The simple description to be used is when solutions in $\pi_I(\mathcal{P})$ have the same structure as solutions in $\mathcal{P}$. In this chapter we consider optimization problems where the convex hull of feasible solutions, defined on some graph G and denoted by $\mathcal{P}(G)$, satisfies the following projection property:

$$\pi_I(\mathcal{P}(G)) = \mathcal{P}(G_I). \tag{3.1}$$

To simplify notation we write $\mathcal{P}$ instead of $\mathcal{P}(G)$ and $\mathcal{P}_I$ instead of $\mathcal{P}(G_I)$. We call this the *projection property* since the solutions in $\mathcal{P}$ project down to solutions in $\pi_I(\mathcal{P})$. Two combinatorial problems that satisfy this property are the max-cut problem and the stable set problem.

3.2.1 Projection property for the max-cut problem

Consider first the max-cut problem on a graph with n nodes. Recall that the feasible solutions are cut matrices of the form cc^T with $c \in \{-1, 1\}^n$. Let the convex hull of cut matrices be denoted by CUT_n. This is generally known as the cut polytope. It follows from the definition that if $|I| = k$ then

$$\pi_I(\text{CUT}_n) = \text{conv}\{cc^T \colon c \in \{-1, 1\}^k\} = \text{CUT}_k.$$

Note that the projection property holds for the max-cut problem because cuts on the graph G are induced cuts in the induced subgraph G_I. In other words, a partitioning of the vertices of G naturally leads to a partitioning of the subset of vertices in G_I.

3.2.2 Projection property for the stable set problem

Recall that $G = (V, E)$ is a given graph where $|V| = n$. A stable set

$$S := \{I \subseteq V \ : \ \forall i, j \in I, (i, j) \notin E\}$$

is a subset of vertices where no two are connected by an edge of the graph. The stable set problem is to find the stable set of maximum size for a given graph G. We consider

$$\text{STAB}(G) := \text{conv}\{s_i \ : \ s_i \text{ is the incidence vector of a stable set in } G\}. \tag{3.2}$$

Note that $\text{STAB}(G) \subseteq \mathbb{R}^n$ and that

$$\pi_I(\text{STAB}(G)) = \text{STAB}(G_I),$$

where G_I denotes the subgraph induced by I. In a slight abuse of notation we also denote by $\pi_I(x)$ the projection of the vector x onto the coordinates in I.

Similar to the max-cut problem we can observe that a stable set on the graph G is a stable set on the induced subgraph G_I.

3.2.3 Projection property failure

This property does not hold for combinatorial problems in general. For example the travelling saleman problem does not satisfy the projection property since the restriction of a hamiltonian cycle to a proper subgraph will not be a cycle. Similarly it does not hold for the quadratic assignment problem since the restriction of an assignment to a subset of vertices may not be an assignment.

3.3 A hierarchy of relaxations based on kPPCs

Recall that for a given cost matrix C, the general combinatorial optimization problem is:

$$z_{\mathcal{P}} := \max\{\langle C, X\rangle \colon X \in \mathcal{P}\}.$$

where $\mathcal{P} = \text{conv}\{X_1, \ldots\}$ (i.e. the convex hull of feasible combinatorial solutions). However the difficulty lies in enumerating all the feasible solutions, X_i, and defining the polytope $\mathcal{P}$.

A formal description of the new hierarchy of relaxations starts with an initial tractable relaxation over the set $\mathcal{R} \supseteq \mathcal{P}$ that can be solved efficiently. Let the initial relaxation be:

$$z_{\mathcal{R}} := \max\{\langle C, X\rangle \colon X \in \mathcal{R}\} \tag{3.3}$$

We are particularly interested in cases where $\mathcal{R}$ is a spectrahedron (i.e. the intersection of the cone of semidefinite matrices $\mathcal{S}_n^+$ with an affine linear space) making the above problem a semidefinite optimization problem.

For fixed $k \in \mathbb{N}$, the relaxation (3.3) is tightened by adding the *k-projection polytope constraints*:

$$\pi_I(X) \in \pi_I(\mathcal{P}) \; \forall I \subseteq N, \; |I| = k.$$

Under the projection property (3.1), this simplifies to

$$X_I \in \mathcal{P}_I \forall I \subseteq N, \; |I| = k.$$

For small values of k, we can express this condition in a more convenient way by exploiting the fact that the vertices v_i^I of $\mathcal{P}_I$ can be enumerated explicitly and requiring that X_I lay in the convex hull of the vertices of $\mathcal{P}_I$:

$$X_I = \sum_i \lambda_i^I v_i^I \text{ with } \lambda_i^I \geq 0, \; \sum_i \lambda_i^I = 1. \tag{3.4}$$

Thus level k of our hierarchy is:

$$\begin{aligned} z_{\mathcal{R},k} := \max \Big\{ \langle C, X\rangle \quad : \quad & X \in \mathcal{R}, \\ & X_I = \sum_i \lambda_i^I v_i^I \text{ with } \lambda_i^I \geq 0, \; \sum_i \lambda_i^I = 1 \; \forall I \subseteq V, \; |I| = k \Big\}. \end{aligned}$$

It is clear from the definitions that

$$z_{\mathcal{R}} \geq z_{\mathcal{R},1} \geq \ldots \geq z_{\mathcal{R},n} = z_{\mathcal{P}}.$$

– *Remark 1* In our applications we focus mostly on relaxations where $\mathcal{R}$ is some spectra-

hedron. It is a nontrivial task to actually identify subsets I so that the current iterate x violates $x \in \mathcal{P}_I$ by a substantial amount.

– *Remark 2* We select the cardinality of I in such a way that $\mathcal{P}_I$ has a relatively small number of vertices. In this case we prefer maintaining the vertex-based description (3.4) of $\pi_I(\mathcal{P})$, which imposes much more structure than adding a single cutting plane.

– *Remark 3* An important distinguishing feature of our hierarchy is that all of the relaxations are formulated in the original space $\mathcal{S}_n$ meaning the size of the matrix variables does not change. This is not the case in other generic hierarchies and SDP relaxations such as the Anjos and Wolkowicz (2002) relaxation, Lasserre (2002), Lovász and Schrijver (1991) and the Sherali and Adams (1990) hierarchies. In these constructions the dimension of the matrix space grows exponentially making even the smallest levels challenging to compute.

3.3.1 kPPC hierarchy of relaxations for the max-cut problem

Recall that the max-cut problem is:

$$z_{\mathrm{MC}} = \max\{\langle L, X\rangle : \ X \in \mathrm{CUT}_n\}$$

To apply our new hierarchy to the max-cut problem, we take the SDP relaxation over the intersection $\mathcal{R} := \mathcal{C} \cap \mathcal{M}$ as our initial relaxation:

$$z_{\mathcal{R}} = \max\{\langle L, X\rangle \colon X \in \mathcal{R}\}.$$

where $\mathcal{C}$ is the set of correlation matrices and $\mathcal{M}$ is the metric polytope.

The motivation for this choice is that this relaxation provides one of the most competitive bounds if both practical efficiency and strength of the relaxation are taken into account.

The k^{th} level of the new hierarchy applied to the max-cut relaxation with $\mathcal{R} = \mathcal{C} \cap \mathcal{M}$ is:

$$z_{\mathcal{R},k} = \max\{\langle L, X\rangle \colon \ X \in \mathcal{R},\ X_I \in \mathrm{CUT}_k \ \forall I \subseteq N,\ |I| = k\}. \tag{3.5}$$

Recall the definition of kPPCs for the k-subset I:

$$C\lambda = \mathrm{triu}(X_I), \quad \sum_{i=1}^{2^{k-1}} \lambda_i = 1, \quad \lambda \geq 0$$

where $C_r = c_r c_r^T$ (with $c_r \in \{-1, 1\}^n$) is a feasible matrix solutions to the max-cut problem.

Therefore the new hierarchy (3.5) can be written as:

$$z_{\mathcal{R},k} = \max\{\langle L, X\rangle \colon \ X \in \mathcal{R},\ C\lambda = \mathrm{triu}(X_I), \quad \sum_{i=1}^{2^{k-1}} \lambda_i = 1, \quad \lambda \geq 0 \forall I \subseteq N,\ |I| = k\}.$$

Trivially, this hierarchy yields z_{MC} for $k = n$. We end this section with a few additional remarks.

– *Remark 1:* kPPCs force $X_I \in \mathrm{CUT}_k$. For $k = 3$ or 4, CUT_k can be completely defined with triangle inequalities (Barahona and Mahjoub, 1986). Therefore the smallest interesting case is $k = 5$.
– *Remark 2:* The definition of kPPCs requires the complete enumeration of all 2^{k-1} feasible solutions on a graph of size k. This is possible only for small values of k. As a result we consider $5 \leq k << n$.

3.3.2 kPPC hierarchy of relaxations for the stable set problem

We now apply the kPPC hierarchy of relaxations to the stable set problem. The stable set polytope $\mathrm{STAB}(G)$ (see (3.2)) of a graph G with $|V| = n$ is contained in $\mathbb{R}^n$. The stability number $\alpha(G)$ of a graph G which gives the cardinality of the largest stable set, is given by

$$\alpha(G) = \max\left\{\sum_{i=1}^{n} x_i \colon \ x \in \mathrm{STAB}(G)\right\}.$$

One of the most well-studied relaxations of $\mathrm{STAB}(G)$ is based on the theta body $\text{Þ}(G)$ introduced by Lovász (1979):

$$\text{Þ}(G) := \{x \in \mathbb{R}^n \colon \ \exists X \in \mathcal{S}_n,\ x = \mathrm{diag}(X),\ X - xx^T \succeq 0,\ x_{ij} = 0\ \forall [i,j] \in E(G)\}.$$

Note that any characteristic vector $s \in \{0,1\}^n$ of a stable set in G yields a *stable set matrix* $S := ss^T$ such that

$$s = \mathrm{diag}(S),\ S - ss^T \succeq 0,\ (S)_{ij} = s_i s_j = 0\ \forall\ [i,j] \in E(G).$$

Hence $\mathrm{STAB}(G) \subseteq \text{Þ}(G)$.

A direct application of the projection approach would impose, for given $x \in \mathbb{R}^n$, the constraint $x_I \in \mathrm{STAB}(G_I)$ for subsets $I \subseteq V$. On the other hand, the set $\text{Þ}(G)$ can also be viewed as a matrix relaxation of the stable set problem projected to the main diagonal. We define

$\text{STAB}^2(G)$ to be the convex hull of stable set matrices:

$$\text{STAB}^2(G) := \text{conv}\{ss^T : s \text{ characteristic vector of stable set}\}.$$

Thus the projection of $\text{STAB}^2(G)$ to the main diagonal gives $\text{STAB}(G)$:

$$\text{diag}(\text{STAB}^2(G)) = \text{STAB}(G).$$

We propose to apply the subgraph projection idea to $\text{STAB}^2(G)$. Our starting point is the relaxation over $\text{Þ}(G)$. It is also called the Lovász theta function and can be formulated as

$$\theta(G) := \max\{\text{tr}(X) : X \in \mathcal{S}_n, x_{ij} = 0 \ \forall [i,j] \in E,\ x = \text{diag}(X),\ X - xx^T \succeq 0\}.$$

This relaxation is now strengthened by the k-projection polytope constraints:

$$X_I \in \text{STAB}^2(G_I) \ \forall I \subseteq N,\ |I| = k.$$

We emphasize the fact that it is possible to have $X_I \notin \text{STAB}^2(G_I)$, but $\text{diag}(X_I) \in \text{STAB}(G_I)$. This could even happen for subsets $I = \{r, s\}$ if $x_{rs} < 0$.

3.4 Max-cut examples

This section presents computational results for max-cut problems. The goal is to illustrate two key features of the kPPC hierarchy. First we compare our hierarchy with other hierarchies (i.e. Lasserre and Anjos & Wolkowicz). Since all three hierarchies (kPPCs, Lasserre and Anjos-Wolkowicz) can only be completely enumerated for small examples we consider examples of size 7, 9 and 11. Secondly, we note that not all $\binom{n}{k}$ kPPCs are required to improve the bound. Section 3.4.2 examines larger examples (with $n = 80$ and 100) and shows that even when only a limited number of violated kPPCs are iteratively added to the relaxation there is still an improvement on the bound.

3.4.1 Small max-cut examples

In this subsection we illustrate the behaviour of the hierarchy (3.5) on selected small max-cut instances. Because these instances are small, all the relaxations in the hierarchy can be solved to optimality.

We first consider the 7×7 matrix

$$Q = -\frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 1 & -2 & -1 & 0 \\ 1 & 0 & 1 & 1 & -2 & 0 & -1 \\ 1 & 1 & 0 & 1 & -2 & -1 & 0 \\ 1 & 1 & 1 & 0 & -2 & 0 & -1 \\ -2 & -2 & -2 & -2 & 0 & 1 & 1 \\ -1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 1 & -1 & 0 \end{pmatrix}. \tag{3.6}$$

Grishukhin (1990) showed that $\langle Q, X\rangle \geq 5$ is a facet of the cut polytope CUT_7. Hence maximizing $\langle Q, X\rangle$ over various supersets of CUT_7 shows how close we come to this facet using the respective relaxations. The results are reported in Table 3.1.

Table 3.1 Bounds and relative gaps to optimality (%) obtained from various relaxations for the Grishukhin inequality of CUT_7

Relaxation (bound)	Bound	Gap (%)
$\mathcal{C}_7$ ($z_{\mathcal{C}}$)	6.9518	39.04
$\mathcal{C}_7 \cap \mathcal{M}_7$ ($z_{\mathcal{R}}$)	6.0584	21.17
$\mathcal{C}_7 \cap \mathcal{M}_7$ and all CUT_5s ($z_{\mathcal{R},5}$)	5.8000	16.00
$\mathcal{C}_7 \cap \mathcal{M}_7$ and all CUT_6s ($z_{\mathcal{R},6}$)	5.6667	13.33
Anjos & Wolkowicz	5.7075	14.15
Lasserre level 2	5.6152	12.30
CUT_7 ($z_{\mathcal{P}}$)	5.0000	0.00

A similar distinction between the relaxations occurs in case of the clique web inequalities (Deza and Laurent, 1997b). Recall that these inequalities are defined as follows : Let n, p, q, r be integers such that $n = p+q, p-q = 2r+1, q \geq 2$ and let $b := (1, \ldots, 1, -1, \ldots, -1)^T$ be a vector of length n where the first p coefficients are equal to $+1$ and the last q coefficients are equal to -1. AW_p^r defines the antiweb as the graph with vertex set $V_p = \{1, 2, \ldots, p\}$ and edge set defined by the pairs $(i, i+1), (i, i+2), \ldots, (i, i+r)$, $\forall i \in V_p$. Then the clique web inequalities are

$$\sum_{1 \leq i < j \leq n} b_i b_j x_{ij} - \sum_{ij \in \mathrm{AW}_p^r} x_{ij} \leq 0$$

We consider the cases $n = 9$ and $n = 11$ and compare again the various levels of our new hierarchy. These inequalities are parametrized by the integer r with $0 \leq r \leq \frac{n-5}{2}$. The results are reported in Tables 3.2 and 3.3.

Table 3.2 Bounds for the clique web inequality with $n = 9$.

Relaxation (bound)	$r = 1$	$r = 2$
$\mathcal{C}_9$ ($z_{\mathcal{C}}$)	8.40	8.99
$\mathcal{C}_9 \cap \mathcal{M}_9$ ($z_{\mathcal{R}}$)	7.12	7.12
$\mathcal{C}_9 \cap \mathcal{M}_9$ and all CUT_5s ($z_{\mathcal{R},5}$)	6.86	7.07
$\mathcal{C}_9 \cap \mathcal{M}_9$ and all CUT_6s ($z_{\mathcal{R},6}$)	6.86	7.07
$\mathcal{C}_9 \cap \mathcal{M}_9$ and all CUT_7s ($z_{\mathcal{R},7}$)	6.75	6.57
$\mathcal{C}_9 \cap \mathcal{M}_9$ and all CUT_8s ($z_{\mathcal{R},8}$)	6.64	6.56
Anjos & Wolkowicz	6.72	6.79
Lasserre level 2	6.59	6.55
CUT_9 ($z_{\mathcal{P}}$)	6.00	6.00

Table 3.3 Bounds for the clique web inequality with $n = 11$.

Relaxation (bound)	$r = 1$	$r = 2$	$r = 3$
$\mathcal{C}_{11}$ ($z_{\mathcal{C}}$)	10.83	13.31	12.10
$\mathcal{C}_{11} \cap \mathcal{M}_{11}$ ($z_{\mathcal{R}}$)	9.28	10.62	9.28
$\mathcal{C}_{11} \cap \mathcal{M}_{11}$ and all CUT_5s ($z_{\mathcal{R},5}$)	9.21	10.62	9.25
$\mathcal{C}_{11} \cap \mathcal{M}_{11}$ and all CUT_6s ($z_{\mathcal{R},6}$)	9.21	10.62	9.25
$\mathcal{C}_{11} \cap \mathcal{M}_{11}$ and all CUT_7s ($z_{\mathcal{R},7}$)	9.00	9.96	8.87
$\mathcal{C}_{11} \cap \mathcal{M}_{11}$ and all CUT_8s ($z_{\mathcal{R},8}$)	8.86	9.96	8.87
$\mathcal{C}_{11} \cap \mathcal{M}_{11}$ and all CUT_9s ($z_{\mathcal{R},9}$)	8.79	9.59	8.52
$\mathcal{C}_{11} \cap \mathcal{M}_{11}$ and all CUT_{10}s ($z_{\mathcal{R},10}$)	8.56	9.50	8.44
Anjos & Wolkowicz	8.87	10.02	8.93
Lasserre level 2	8.72	9.62	8.59
CUT_{11} ($z_{\mathcal{P}}$)	8.00	9.00	8.00

These first examples (Tables 3.1-3.3) show that our new hierarchy is competitive, even compared to the 2nd level of the Lasserre hierarchy. Even though we have included the kPPCs for all subsets of cardinality k in these computations, a closer look at the computational results shows that in fact only a small fraction of these constraints are necessary to obtain the given bounds. It is also quite striking that going down to level $k = n - 1$ still leaves a rather large gap in these instances. Since the objective function corresponds to a facet of the cut polytope, this is an illustration of the worst case behaviour of our hierarchy.

3.4.2 Larger max-cut examples

This section reports the results of our computational experiments with the new hierarchy on larger instances of max-cut. The inclusion of all kPPCs for some $k \geq 5$ is computationally

prohibitive. Instead, we run through all 5-projection polytope constraints and include only the 100 most violated ones, iterating this process. To check whether or not $X_I \in \mathrm{CUT}_{|I|}$ we could compute the projection of X_I to $\mathrm{CUT}_{|I|}$. This requires in general the solution of a convex quadratic problem in 2^{k-1} variables if $|I| = k$.

For the case $k = 5$ we exploit the fact that the facets of CUT_5 are given by the triangle inequalities, which are always satisfied as we assume $X \in \mathcal{M}$, and the pentagonal inequalities $f^T X_I f \geq 1$ for all $f \in \{-1, 1\}^5$. We scan through all pentagonal inequalities and select the 100 subsets I corresponding to the largest violations. We add the corresponding projection constraints to the SDP relaxation, solve the resulting relaxation using SDPT3, and iterate this process. In the tables below, the number of these iterations is limited to 10. The final bound approximates $z_{\mathcal{C}\cap\mathcal{M},5}$ from above. The following tables contain representative results from our experiments.

We first look at max-cut examples of random unweighted graphs from the Erdős-Renyi model where each edge appears with probability p independent of the other edges. We consider graphs on $n = 80$ nodes with $p = \frac{1}{2}$. These instances can be found in the BiqMac Library (see `http://biqmac.uni-klu.ac.at/biqmaclib.html`). For a comparison, we also provide the number of nodes used to prove optimality by the software package BiqMac (Rendl, Rinaldi, and Wiegele, 2010). The results are reported in Table 3.4.

We observe that our new bound is strong enough to solve most of the instances either at the root node or at the first two levels of the branching tree. In sharp contrast with the results for BiqMac, only two out of the ten instances could not be solved within the first two levels of a branch-and-bound procedure when using the new bound. (Note that for the instance g05_80.1, the relaxation $\mathcal{C} \cap \mathcal{M}$ already closes the gap.)

We also look at larger instances of size $n = 100$. We consider graphs with both positive and negative edge weights and collect a sample of results in Table 3.5. Again these instances can be found in the BiqMac Library. Here we report the percentage gap between the optimal cut value and each of the bounds (with respect to the optimal). We again see that our rather simple-minded improvement strategy limited to $k = 5$ yields a significant improvement of the bound.

3.5 Small stable set examples

To further emphasize the potential of our new bounding procedure, this section includes computational results of the new hierarchy applied to the stable set problem. Here we iteratively include only the most violated kPPCs for $k \leq 6$. We consider random graphs with edge den-

Table 3.4 Bounds and number of nodes in a branch-and-bound tree for unweighted graphs on $n = 80$ nodes.

Graph name	Optimal cut value	Optimizing over $\mathcal{C}$	Optimizing over $\mathcal{C} \cap \mathcal{M}$	New bound	Nodes with BiqMac	Nodes with new bound
g05_80.0	929	950.92	934.24	931.01	59	5
g05_80.1	941	957.25	941.76	–	3	–
g05_80.2	934	955.55	937.24	934.52	17	1
g05_80.3	923	947.59	932.32	929.15	523	>7
g05_80.4	932	955.31	936.53	933.83	39	3
g05_80.5	926	947.51	931.42	928.41	65	7
g05_80.6	929	948.68	933.24	930.40	31	3
g05_80.7	929	949.86	932.63	929.58	23	1
g05_80.8	925	946.67	930.53	927.42	73	7
g05_80.9	923	943.66	929.95	926.67	157	>7

sity 25% (g60-25, g80-25) and a graph with density 10% (g100-10). We also consider a cubic graph with $n = 74$ (CubicVT74-9) available through the internet at `http://www.matapp.unimib.it/~spiga/census.html`, and finally a 3-dimensional grid graph (spin5). For these graphs there is a significant difference between θ and α.

In all cases the new bound (with 100 kPPCs) provides a clear improvement over the theta number $\theta(G)$. This fact is particularly impressive for the cubic graph and the grid graph.

3.6 Conclusion

The focus of this chapter has been the definition of a hierarchy of relaxations for combinatorial problems that satisfy the projection property. The key features of this hierarchy is that all resulting relaxations are formulated in the original matrix space. The max-cut and the stable set problem both satisfy the projection property. For both problems we:
– defined kPPCs,
– constructed the hierarchy of relaxations based on an initial relaxation specific to the problem, and
– presented computational results showing how iteratively adding 100 violated kPPCs can significantly improve the bound.

The following observations can be made regarding our computational results for max-cut and stable set instances:

1. The hierarchy may not reach optimality until the final level and in the worst case situation the gap can still be quite large at the $k = n - 1$ level.

Table 3.5 Bounds and relative gaps to optimality (%) for dense graphs with positive and negative weights on $n = 100$ nodes.

Graph name	Optimal cut value	Optimizing over $\mathcal{C}$	Optimizing over $\mathcal{C} \cap \mathcal{M}$	New bound	Gap for $\mathcal{C} \cap \mathcal{M}$	Gap for new bound
w09_100.0	2121	2500.30	2234.39	2189.54	5.35	3.23
w09_100.1	2096	2522.03	2263.82	2218.30	8.01	5.83
w09_100.2	2738	3129.99	2880.60	2833.92	5.21	3.50
w09_100.3	1990	2333.05	2131.55	2084.76	7.11	4.76
w09_100.4	2033	2424.98	2154.71	2109.86	5.99	3.78
w09_100.5	2433	2733.64	2454.66	2433.08	0.89	0.00
w09_100.6	2220	2552.11	2281.17	2241.92	2.76	0.99
w09_100.7	2252	2639.73	2355.48	2312.90	4.60	2.70
w09_100.8	1843	2213.12	1924.37	1882.62	4.42	2.15
w09_100.9	2043	2409.78	2161.63	2116.84	5.81	3.61

Table 3.6 Results for instances of stable-set problems of various sizes and densities

Graph	n	$\theta(G)$	New bound	$\alpha(G)$
g60-25	60	15.0058	14.71	14
cubic	74	34.8561	33.34	≥ 32
g80-25	80	17.1670	17.01	17
g100-10	100	32.1166	31.52	≥ 29
spin5	125	55.9017	51.61	≥ 50

2. Significant improvements in the bound can be reached at the first level ($k = 5$) of the hierarchy. This strong bound can greatly reduce the number of nodes BiqMac requires.
3. Although there are $\binom{n}{k}$ kPPCs at each level of the hierarchy, we can attain a value close to the bound for a level with only a small fraction of the kPPCs.
4. On the other hand, there is no guarantee that including all kPPCs from a level will improve the bound from the previous level. However this outcome seems to be atypical.

The major limitations of the hierarchy arises in the practical implementation. The entire k^{th} level of the hierarchy cannot be explicitly enumerated for large n or large k. However, results presented in this chapter show that the entire level of the hierarchy is not necessary to improve the bound. Therefore a good separation algorithm is essential to a practical implementation. The following chapters examine how to find the (limited number of) kPPCs that will improve the bound for a given small k.

CHAPTER 4 EXACT SEPARATION OF K-PROJECTION POLYTOPE CONSTRAINTS

4.1 Introduction

Cuts are often used as an efficient means to tighten continuous relaxations of (mixed) integer optimization problems and are a vital component of branch-and-cut and cutting plane algorithms. A critical step of these algorithms is solving the separation problem to find valid cuts (typically called valid inequalities), that are violated by the current solution but are satisfied by every feasible integer solution. The problem of finding a cut that achieves maximal violation over all possible cuts for a given solution to the relaxation is called the maximally violated valid inequality problem (MVVIP) (Lodi, Ralphs, and Woeginger, 2012).

This chapter presents a model for finding the most violated k-projection polytope constraint. Because kPPCs are not inequalities we refer to the problem as the maximally violated k-projection polytope constraint problem (MVkPPC).

Specifically we present a bilevel optimization model that fits into the MVVIP framework and finds the maximally violated k-projection polytope constraint. We also show how to reformulate the model as a single level mixed integer second order cone optimization problem, and how the single level model can be strengthened by reformulating it using a different set of binary variables and by breaking symmetry.

4.2 Finding maximally violated kPPCs

Two definitions critical to the MVVIP are validity and membership. We discuss them in the context of kPPCs in turn.

4.2.1 Validity

Recall that the validity verification problem asks if all feasible solutions ($\mathcal{F}$) are satisfied by the cut(s) (i.e. constraint(s)) under consideration. The cut is valid if all feasible solutions are satisfied. The kPPCs are always valid because they satisfy the projection property (i.e. for all feasible solutions $X \in \mathcal{F}$ we know that $X_I \in \text{CUT}_k \ \forall I \subseteq V$ with $|I| = k$), therefore the validity verification problem will not be explicitly addressed for kPPCs .

4.2.2 Membership

Recall that the membership problem determines whether a given point is contained in the intersection of a polyhedron and a given cut. Therefore the membership problem for a kPPC is: for a given $k \times k$ submatrix X_I^* of X^*, where X^* is the optimal solution to a relaxation and $|I| = k$, is $X_I^* \in \text{CUT}_{|I|}$? If $X_I^* \notin \text{CUT}_{|I|}$ then adding the kPPC for set I will tighten the relaxation.

The following problem, denoted *distance-to-polytope* (D2P), not only determines if the point X^* is a member of the polyhedron $\text{CUT}_{|I|}$ but also quantifies the separation if X^* is not a member of $\text{CUT}_{|I|}$:

$$(D2P) \qquad d^* = \min \left\{ ||\text{triu}\,(X_I^*) - Q\lambda|| \;:\; e^T\lambda = 1,\; \lambda \geq 0 \right\}$$

where X_I^* is the principal submatrix indexed by I of X^*, e is the vector of all ones of the appropriate size, triu(X) is the vector formed from the elements in the strictly upper triangular part of matrix X taken column-wise and Q is a $\binom{|I|}{2} \times 2^{|I|-1}$ matrix with the column, Q_i, being the feasible cut solution in vector form (i.e. $Q_i = \text{triu}(C_i)$).

The optimal objective value d^* equals the euclidean distance between X_I^* and $\text{CUT}_{|I|}$. Therefore

$$\begin{aligned} &\text{If } d^* = 0 \text{ then } X_I^* \in \text{CUT}_{|I|} \\ &\text{If } d^* > 0 \text{ then } X_I^* \notin \text{CUT}_{|I|} \end{aligned} \tag{4.1}$$

To illustrate the definition we revisit the Grishukhin (1990) example of size 7 where $\langle Q, X \rangle$ is maximized (see (3.6) for definition of Q). Let X^* be the optimal solution when optimized over $\mathcal{C} \cap \mathcal{M}$. For $k = 5$ we can find the distance-to-polytope (d^*) for each $I \in V$ such that $|I| = 5$. Table 4.1 shows the results of adding projection polytope constraints to the initial relaxation. The 3rd column shows the bound when the projection polytope constraint associated with the single induced subgraph I is added to $\mathcal{C} \cap \mathcal{M}$. The 4^{th} and 5^{th} column show the bound when multiple kPPCs are added.

The bound of the relaxation $\mathcal{C} \cap \mathcal{M}$ is 6.0584 and the final optimal objective value is 5.0. Further bounds for this problem can be seen in Table 3.1.

Note that for any induced subgraph I where the distance-to-polytope is equal to (approximately) 0 adding the corresponding kPPC does not change the optimal objective value or optimal solution. We observe that even if we add all the kPPCs corresponding to the 14 induced subgraphs with $d^* \leq .0008$ the optimal objective value does not change since for

Table 4.1 Bounds for adding different $k = 5$ PPCs to the SDP relaxation $\mathcal{C} \cap \mathcal{M}$ which has bound 6.0584.

d^*	I	bound	bound	bound
0.1274	[1 3 5 6 7]	5.9800	5.9000	
	[2 4 5 6 7]	5.9800		
0.0800	[1 2 3 5 6]	6.0371	5.9412	5.8000
	[1 2 4 5 7]	6.0371		
	[1 3 4 5 6]	6.0371		
	[2 3 4 5 7]	6.0371		
0.0563	[1 2 3 4 5]	6.0485		
≤ 0.0008	the remaining 14 subsets		6.0584	

each of these sets of indices $X_I^* \in \text{CUT}_{|I|}$ and the optimal solution X^* is still feasible. Adding all $k = 5$ projection polytope constraints improves the bound to 5.8000.

4.3 Formulation of the MVkPPCP as a bilevel problem

The MVVIP can now be easily extended to find the maximally violated kPPC . The following is the MVkPPCP formulation:

$$
\begin{aligned}
(\text{DP}_{\text{Bilevel}}) \qquad & \max_{B,d} \quad d \\
& \text{s.t.} \quad Be_n = e_k && (4.2)\\
& \qquad\; Be_k \leq e_n && (4.3)\\
& \qquad\; B \in \{0,1\}^{n\times k} && (4.4)\\
& \qquad\; d = \left\{ \min_{e^T\lambda=1,\, \lambda\geq 0} ||\text{triu}\left(B^T X B\right) - Q\lambda|| \right\} && (4.5)
\end{aligned}
$$

where e_n is a $n \times 1$ vector of all ones ; triu(X) is the vector formed from the elements in the strictly upper triangular part of matrix X taken column-wise. The inner problem solves the (D2P) problem for a certain k-projection polytope constraint. The variables of the outer problem define the parameters of the kPPC , namely the k indices that define the submatrix X_I.

In ($\text{DP}_{\text{Bilevel}}$) the outer problem variable B is a $n \times k$ row selection matrix. Recall that B_{ij} is the element of the matrix B at row i and column j. Constraints (4.2)-(4.4) require that

exactly k rows are selected, these rows correspond to the vertices of the induced subgraph that defines the kPPC. The inner problem (4.5) is (D2P) where the principal submatrix X_I is written more generally as $B^T XB$. Since the outer variable B is assumed to be given in the inner problem (hence assumed to be a row selection matrix) this product selects the principal submatrix indexed by the rows selected in B. Namely,

$$X_I = B^T XB \text{ where (4.2)-(4.4) are satisfied and } \sum_{j=1}^{k} B_{ij} = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{if } i \notin I \end{cases}.$$

4.4 Reformulation of the MVkPPCP as a single-level problem

In this section we reformulate ($\text{DP}_{\text{Bilevel}}$) to a single level mixed binary second order cone optimization problem. We begin by presenting the full single level problem and in the remainder of this section we show the equivalence between the two problems.

$$
\begin{aligned}
(\mathrm{DP}_{\mathrm{single}}) \qquad \max_{B,d,\lambda,\mu,y,z,\alpha,\beta,\gamma} \quad & d \\
\text{s.t.} \quad & (4.2)-(4.4) \\
& e^T\lambda = 1 && (4.6)\\
& \mu_{jt} + Q\lambda - \sum_{i=1\ldots n}\ \sum_{s=1\ldots n: s\neq i} X_{is}\beta_{ijst} = 0 \quad \forall 1 \le j < t \le k && (4.7)\\
& \lambda \ge 0 && (4.8)\\
& \begin{bmatrix} d \\ \mu \end{bmatrix} \in \mathrm{SOC}^{1+\binom{k}{2}} && (4.9)\\
& ye + Q^T z \le 0 && (4.10)\\
& \begin{bmatrix} 1 \\ -z \end{bmatrix} \in \mathrm{SOC}^{1+\binom{k}{2}} && (4.11)\\
& d - y - \sum_{ijst\in\Gamma} X_{is}\gamma_{ijst} = 0 && (4.12)\\
& \alpha \in \{0,1\}^{2^{k-1}} && (4.13)\\
& \alpha_i - \lambda_i \ge 0 \qquad \forall i = 1\ldots 2^{k-1} && (4.14)\\
& \left(2^{\frac{k+1}{2}}\right)\alpha_i - y - Q_i^T z \le \left(2^{\frac{k+1}{2}}\right) \qquad \forall i = 1\ldots 2^{k-1} && (4.15)\\
& \beta_{ijst} - B_{ij} \le 0 \qquad \forall\, ijst \in \Gamma && (4.16)\\
& \beta_{ijst} - B_{st} \le 0 \qquad \forall\, ijst \in \Gamma && (4.17)\\
& \sum_{ijst\in\Gamma} \beta_{ijst} = \binom{k}{2} && (4.18)
\end{aligned}
$$

$$
\begin{aligned}
& 0 \le \beta_{ijst} \le 1 \qquad \forall\, ijst \in \Gamma && (4.19)\\
& \gamma_{ijst} + \beta_{ijst} \ge 0 \qquad \forall\, ijst \in \Gamma && (4.20)\\
& \gamma_{ijst} - \beta_{ijst} \le 0 \qquad \forall\, ijst \in \Gamma && (4.21)\\
& \gamma_{ijst} - z_{jt} - \beta_{ijst} \ge -1 \qquad \forall\, ijst \in \Gamma && (4.22)\\
& \gamma_{ijst} - z_{jt} + \beta_{ijst} \le 1 \qquad \forall\, ijst \in \Gamma && (4.23)
\end{aligned}
$$

where $\Gamma = \{ijst \mid i, s = 1, \ldots, n, i \neq s, 1 \le j < t \le k\}$

4.4.1 Reformulating steps

In this section we present the steps to transform ($\mathrm{DP}_{\mathrm{Bilevel}}$) into ($\mathrm{DP}_{\mathrm{single}}$). For clarity we split the reformulation into the following three steps:

1. replace the inner problem by its optimality conditions,
2. rewrite the complementary slackness conditions and
3. linearize the nonlinear terms.

We consider the steps in turn.

Step 1 : Rewrite the inner problem

The first step in the reformulation is to transform the bilevel problem to a single level problem. Recall the definition of second order cones (SOC):

$$\begin{bmatrix} x_o \\ \bar{x} \end{bmatrix} \in \mathrm{SOC}^{1+n} \Leftrightarrow x_o \geq ||\bar{x}||$$

where x_o is a scalar and $\bar{x}$ is a vector of length n.
Using this property we can reformulate the inner problem (4.5) to the following SOC problem:

$$\begin{aligned} (\mathrm{P}_{\mathrm{Inner}}) \qquad \min_{d,\lambda,\mu} \quad & d \\ \text{s.t.} \quad & (4.6), (4.8), (4.9) \\ & \mu_{jt} + Q\lambda - \sum_{i=1}^{n-1} \sum_{s=i+1}^{n} X_{is} B_{ij} B_{st} = 0 \qquad \forall 1 \leq j < t \leq k \end{aligned} \tag{4.24}$$

where constraint (4.24) ensures that $\mu = \mathrm{triu}\left(B^T X B\right) - Q\lambda$, and the minimization of d implies $d = ||\mu||$ at optimality. Recall that B is given (and not a variable) in this formulation. The dual of $\mathrm{P}_{\mathrm{Inner}}$ is:

$$\begin{aligned} (\mathrm{D}_{\mathrm{Inner}}) \qquad \max_{y,z} \quad & y + \sum_{ijst \in \Gamma} X_{is} B_{ij} B_{st} z_{jt} \\ \text{s.t.} \quad & (4.10), (4.11) \end{aligned}$$

where $y \in \mathbb{R}$ and $z \in \mathbb{R}^{\binom{k}{2}}$ are variables.

The inner problem ($\mathrm{P}_{\mathrm{Inner}}$) is convex (since it is a SOC problem) and both the objective

function and constraints are differentiable. In addition the problem satisfies Slater's condition. An example of a strictly feasible primal solution is given by μ, λ and d that satisfy the following:

$$\lambda_i = \frac{1}{2^{k-1}},\ \ \mu_{jt} = -Q\lambda + \sum_{i=1}^{n-1} \sum_{s=i+1}^{n} X_{is} B_{ij} B_{st}\, \forall 1 \leq j < t \leq k;\ \ d > ||\mu||$$

and a strictly feasible dual solution is given by the following y and z:

$$z_{jt} = 0\, \forall 1 \leq j < t \leq k;\ \ y < 0$$

Therefore the KKT conditions are both necessary and sufficient for the optimality of ($\mathrm{P_{Inner}}$) and we can replace the rewritten inner problem (4.5) with the primal feasibility constraints (4.6), (4.8), (4.9) and (4.24); the dual feasibility constraints (4.10) and (4.11) and the complementary slackness constraints:

$$\lambda_i(y + Q_i^T z) = 0 \quad \forall i = 1, \ldots, \binom{k}{2} \tag{4.25}$$

$$d - \mu^T z = 0 \tag{4.26}$$

The resulting problem is no longer bilevel but is exact (and not a relaxation).

Step 2 : Rewrite complementary slackness conditions

The next step in the reformulation process is to substitute linear/binary constraints for the quadratic complementary slackness constraints (4.25) and (4.26). Constraint (4.25) implies $\lambda_i = 0$ or $y + Q_i^T z = 0$ for each i. Lemma 1 shows how this disjunction is modeled with the binary variables α.

Lemma 1

There exists a feasible solution to constraints (4.8), (4.10), (4.13)-(4.15) *if and only if there exists a feasible solution to* (4.25).

Proof

($\Rightarrow$) Assume constraints (4.8), (4.10), (4.13)-(4.15) are satisfied. For each i (4.13) implies $\alpha_i = 0$ or 1. Consider these cases in turn.

If $\alpha_i = 0$ then (4.8) and (4.14) imply $\lambda_i = 0$.

If $\alpha_i = 1$ then (4.10) and (4.15) imply $y + Q_i^T z = 0$

($\Leftarrow$) Assume there exists a feasible solution (λ, y, z) such that (4.25) is satisfied. Then either $\lambda_i = 0$ or $y + Q_i^T z = 0$. For all i such that $\lambda_i = 0$ set $\alpha_i = 0$ and for all i such that $y + Q_i^T z = 0$ set $\alpha_i = 1$. If $\lambda_i = 0$ and $y + Q_i^T z = 0$ then $\alpha_i = 0$ or 1 will satisfy the constraints. ■

Moreover constraints (4.14) and (4.15) are constructed so that they do not introduce any additional restrictions on λ_i or $y + Q_i^T z$. If $\alpha_i = 0$ then (4.15) implies $y + Q_i^T z \geq -\left(2^{\frac{k+1}{2}}\right)$ which is unrestrictive and if $\alpha_i = 1$ then (4.14) implies $\lambda \leq 1$ which is also unrestrictive.

The following constraint is added to the model so that the quadratic constraint (4.26) is implicitly enforced.

$$d - y - \sum_{ijst \in \mathcal{S}} X_{is} B_{ij} B_{st} z_{jt} = 0 \tag{4.27}$$

Lemma 2 details how (4.27) plus constraints already in the model imply (4.26), however we begin by discussing the motivation for this constraint. Constraint (4.27) states strong duality of the inner problem holds, namely that the gap between the primal and dual objective values is 0. For strong duality to hold for a pair of conic optimization problems a constraint qualification must be satisfied (Slater's condition was verified in the previous step). It is important to note the difference between the observation that Slaters condition implies that strong duality holds and Lemma 2. Lemma 2 shows that the strong duality equation with primal and dual feasibility constraints together imply the complementary slackness constraint (4.26).

Lemma 2
If a feasible solution exists for (4.8), (4.10), (4.13)-(4.15) *and* (4.24) *then* (4.26) *is satisfied if and only if* (4.27) *is satisfied.*

Proof
Assume constraints (4.8), (4.10) (4.13)-(4.15) and (4.24) are satisfied. By Lemma 1 we know that $\lambda_i(y + Q_i^T z) = 0 \, \forall i$. Summing over all i we get that $\lambda^T(ye + Q^T z) = 0$. We use this fact in the following equivalences:

$$\begin{aligned}
(4.27) \text{ is satisfied} &\Leftrightarrow d - ye^T\lambda - (\mu + Q\lambda)^T z = 0 \text{ (since (4.6) and (4.24) are satisfied)} \\
&\Leftrightarrow d - \mu^T z - \lambda^T(ye + Q^T z) = 0 \\
&\Leftrightarrow (4.26) \text{ is satisfied} \quad (\text{since } \lambda^T(ye + Q^T z) = 0)
\end{aligned}$$

■

Step 3 : Linearization

The final step of the reformulation is to linearize $B_{ij}B_{st}$ in (4.24) with the variable β_{ijst} to get (4.7) and to linearize $B_{ij}B_{st}z_{jt}$ in (4.27) with the variable γ_{ijst} to get (4.12). We consider these in turn.

Recall that B_{ij} is defined $\forall i = 1, \ldots, n \; \forall j = 1, \ldots, k$. Constraints (4.2)-(4.4) imply that exactly k of the nk variables are equal to 1 and that the remaining are equal to 0. These constraints imply certain $B_{ij}B_{st}$ products will always be 0. Namely,

$$(4.2) \Rightarrow B_{ij}B_{it} = 0 \qquad \forall i = 1, \ldots, n \; \forall j, t = 1, \ldots, k$$
$$(4.3) \Rightarrow B_{ij}B_{sj} = 0 \qquad \forall i, s = 1, \ldots, n \; \forall j = 1, \ldots, k$$

Therefore there is no need to linearize the terms in which $i = s$ or $j = t$. Since $B_{ij}B_{st} = B_{st}B_{ij}$ we can further limit the number of products we linearize to only those with $j < t$. Note that of the $(nk)^2$ products only $n(n-1)\binom{k}{2}$ of them need to be linearized. The indices of the terms that are linearized are denoted by Γ where

$$\Gamma = \{ijst \mid i, s = 1, \ldots, n, \; i \neq s, \; 1 \leq j < t \leq t\}$$

Furthermore since exactly k elements of b are 1 then exactly $\binom{k}{2}$ of the products equal 1 with the remaining products equal to 0. Lemma 3 formalizes this idea and show the constraints necessary to enforce it.

Lemma 3

If constraints (4.2)-(4.4) *are satisfied then there exists a feasible solution to constraints* (4.16)-(4.19) *if and only if* $B_{ij}B_{st} = \beta_{ijst} \; \forall ijst \in \Gamma$.

Proof

Assume constraints (4.2)-(4.4) are satisfied and consider the cases in turn.

($\Rightarrow$) Let $(\hat{B}, \hat{\beta})$ be any feasible solution to constraints (4.16)-(4.19). Constraint (4.4) implies $\hat{B}_{ij}, \hat{B}_{st} \in \{0, 1\}$. Consider the cases in turn.
If $\hat{B}_{ij} = 0$ (resp. $\hat{B}_{st} = 0$) then (4.16) (resp. (4.17)) and (4.19) imply $\hat{\beta}_{ijst} = 0$.
If $\hat{B}_{ij} = \hat{b}_{st} = 1$ then constraints (4.2)-(4.4) imply that exactly k of the nk elements in $\hat{B}$ will be equal to 1 (with the rest equal to 0). Therefore all but $\binom{k}{2}$ of the terms in $\sum_{ijst \in \mathcal{S}} \hat{\beta}_{ijst}$ will be forced to 0 because $\hat{B}_{ij}$ or $\hat{B}_{st}$ equals 0. Since $\hat{\beta}_{ijst} \leq 1 \; \forall ijst \in \mathcal{S}$ and the sum of the nonzero elements of $\hat{\beta}$ equals $\binom{k}{2}$ then the remaining $\hat{\beta}$'s are forced to 1.
Therefore $\hat{\beta}_{ijst} = \hat{B}_{ij}\hat{B}_{st} \; \forall ijst \in \Gamma$ as required.

($\Leftarrow$) Let $B_{ij}B_{st} = \beta_{ijst}\ \forall ijst \in \Gamma$. Constraint (4.4) implies $B_{ij}, B_{st} \in \{0,1\}$. Therefore (4.19) is feasible since $\beta_{ijst} \in \{0,1\}$.
Constraint (4.16) implies $B_{ij}B_{st} - B_{ij} = B_{ij}(B_{st}-1) \leq 0 \forall B_{ij}, B_{st} \in \{0,1\}$. Constraint (4.17) follows similarly.
Finally constraint (4.2) implies that if $B_{ij} = 1$ then $B_{it} = 0\ \forall t \neq j$ and (4.3) implies that there exists exactly one i for each $1 \leq j \leq k$ such that $B_{ij} = 1$ and that if $B_{ij} = 1$ then $B_{sj} = 0\ \forall s \neq i$. Therefore there exists exactly $\binom{k}{2}$ β's equal to 1 and (4.18) is feasible.
■

Note that although β_{ijst} is binary this does not need to be included as an explicit constraint in ($\text{DP}_{\text{single}}$).

The final step is to linearize $B_{ij}B_{st}z_{jt}$ in constraint (4.27) using the variable γ_{ijst}. The linearization happens over the same set Γ. Lemma 4 and its proof provide the details.

Lemma 4
If constraints (4.2)-(4.4) *are satisfied then there exists a feasible solution to* (4.20)-(4.23) *if and only if* $B_{ij}B_{st}z_{jt} = \gamma_{ijst}\ \forall ijst \in \Gamma$.

Proof
Assume constraints (4.2)-(4.4) are satisfied. Consider the directions in turn.

($\Rightarrow$) Let there exist a feasible solution to (4.20)-(4.23). Constraint (4.4) implies $B_{ij}, B_{st} \in \{0,1\}$. Consider the cases in turn.
If $B_{ij} = 0$ or $B_{st} = 0$ then $\beta_{ijst} = 0$ (by Lemma 3). Constraints (4.20) and (4.21) then imply $\gamma_{ijst} = 0\ \forall ijst \in \Gamma$.
If $B_{ij} = B_{st} = 1$ then $\beta_{ijst} = 1$ (by Lemma 3). Constraints (4.22) and (4.23) then imply $\gamma_{ijst} - z_{jt} = 0\ \forall ijst \in \Gamma$.
Therefore in both cases $\gamma_{ijst} = B_{ij}B_{st}z_{jt}\ \forall ijst \in \Gamma$ as required.

($\Leftarrow$) Constraints (4.20)-(4.23) follow immediately from the fact that $\gamma_{ijst} = B_{ij}B_{st}z_{jt}$, $\beta_{ijst} = B_{ij}B_{st}$ and $B_{ij}, B_{st} \in \{0,1\}$.
■

4.4.2 Equivalence of the bilevel and single level models

Recall that the purpose of this section was to show that the bilevel model ($\text{DP}_{\text{Bilevel}}$) can be reformulated to a single level problem. The following theorem combines the steps and lemmas presented previously to formally prove the equivalence of the ($\text{DP}_{\text{Bilevel}}$) and ($\text{DP}_{\text{single}}$).

Theorem 2 *($DP_{Bilevel}$) is equivalent to (DP_{single}).*

Proof

Since the objective function in both models is the same it is sufficient to show that the feasible regions of ($\mathrm{DP_{Bilevel}}$) and ($\mathrm{DP_{single}}$) are equivalent. Namely that (d, B) is a feasible solution to ($\mathrm{DP_{Bilevel}}$) if and only if there exists a $(\mu, \beta, \gamma, \alpha, y, z, \lambda)$ so that (d, B) is a feasible solution to ($\mathrm{DP_{single}}$).

By the KKT theorem and convexity (4.5) can be replaced by the KKT conditions and the feasible region is the same. Therefore

$$
\begin{aligned}
(4.5) &\Leftrightarrow (4.6), (4.8) - (4.11), (4.24) - (4.26).\\
(4.25) &\Leftrightarrow (4.8), (4.10), (4.13) - (4.15)\\
(4.26) &\Leftrightarrow (4.27)\\
(4.24) \text{ and } \beta_{ijst} = B_{ij}B_{st} &\Leftrightarrow (4.7), (4.16) - (4.19)\\
(4.27) \text{ and } \gamma_{ijst} = B_{ij}B_{st}z_{jt} &\Leftrightarrow (4.12), (4.20) - (4.23)\\
\therefore (d, B)\text{is feasible for } (4.2) - (4.5) &\Leftrightarrow \text{there exists a } (\mu, \beta, \gamma, \alpha, y, z, \lambda) \text{ so that}\\
&\qquad (d, B) \text{ is a feasible solution to}\\
&\qquad (4.2) - (4.4), (4.6) - (4.23)
\end{aligned}
$$

■

4.5 Strengthening the single level model

4.5.1 Symmetry

Symmetry exists within the exact separation problem because a subset of k indices will induce the same projection polytope regardless on the order. To eliminate this symmetry we can enforce lexicographical order on B with the following set of constraints:

$$B_{s,j-1} + \sum_{i=1}^{s} B_{ij} \leq 1 \ \ \forall s = 2, \ldots, n, j = 2, \ldots, k. \tag{4.28}$$

Lemma 5

If constraints (4.2)-(4.4) *and* (4.28) *are satisfied then lexicographical order must hold.*

Proof

Assume constraints (4.2)-(4.4) and (4.28) are satisfied. Variable B is a row selection matrix in which each column contains exactly 1 element equal to 1, with the rest equal to 0

(constraints (4.2)-(4.4)). Going through the columns in order we will show that the index of the row selected can only strictly increase.
Considering $B_{i1} \forall i$ (i.e. column 1 of B) we know there exists an i' such that $B_{i'1} = 1$ and $B_{i1} = 0 \forall i \in \{1, \ldots, n\} \setminus \{i'\}$. Therefore (4.28) implies $B_{i2} = 0 \, \forall i \leq i'$ and since each column sums to 1 (and each row sums to at most 1) then there exists $i'' > i'$ such that $B_{i''2} = 1$. By a similar argument $B_{i3} = 0 \forall i \leq i''$ and there exists $i''' > i''$ such that $B_{i'''3} = 0$. Repeating the argument k times implies that if $B_{i'1}, B_{i''2}, \ldots, B_{\bar{i},k}$ are the k elements of B equal to 1 then $i' < i'' < \cdots < \bar{i}$. ■

4.5.2 Reformulation with fewer binary variables

We reduce the number of binary variables from $2^{k-1}+nk$ to $2^{k-1}+n$ by adding constraints (4.29)-(4.33). The proof later in this section shows that the feasible region of the model does not change and that we no longer need to require binarity of the variables B_{ij}. The benefit of these constraints is seen in the computational results which are examined in Section 4.6.

$$\sum_{i=1}^{n} a_i = k \tag{4.29}$$

$$a_i - \sum_{j=1}^{k} B_{ij} = 0 \tag{4.30}$$

$$a_i - \sum_{i'=1}^{i=1} a_{i'} - B_{i1} \leq 0 \qquad \forall i = 1 \ldots n \tag{4.31}$$

$$a_i + \sum_{i'=1}^{i-1} B_{i',j-1} - \sum_{i'=1}^{i-1} B_{i',j} - B_{ij} \leq 1 \qquad \forall i = 2 \ldots n, \forall j = 2 \ldots k \tag{4.32}$$

$$a_i \in \{0, 1\} \qquad \forall i = 1 \ldots n \tag{4.33}$$

These constraints along with the symmetry constraints (4.28) are included in the ($\mathrm{DP}_{\mathrm{single}}$) model. Constraint (4.4) (binarity of B) is removed as it is automatically enforced by constraints (4.29)-(4.33). The model ($\mathrm{DP}_{\mathrm{fewerBinary}}$) is defined as follows:

$$(\mathrm{DP}_{\mathrm{fewerBinary}}) \qquad \max \quad d$$
$$\text{s.t.} \quad (4.2), (4.3), (4.6) - (4.23), (4.28) - (4.33).$$

Lemma 6 certifies that the given set of constraints (including $a \in \{0, 1\}^n$) implies that $B \in \{0, 1\}^{n \times k}$.

Lemma 6

If constraints (4.2), (4.3), (4.29)-(4.33) *and* $0 \leq B_{ij} \leq 1 \, \forall i = 1 \ldots n, j = 1 \ldots k$ *are satisfied then* $B_{ij} \in \{0, 1\}$.

Proof

Let (4.2), (4.3), (4.29)-(4.33) and $0 \leq B_{ij} \leq 1 \, \forall i = 1 \ldots n, j = 1 \ldots k$ be satisfied.
Constraints (4.29) and (4.33) imply there exists exactly k a_i's equal to 1 with the remaining $n - k$ a_i's equal to 0. Let $a_i = 1$ for $i \in \mathcal{A} := \{i_1 < i_2 < \cdots < i_k\}$
For all $i = 1 \ldots n$, $\sum_{i'=1}^{i-1} a_{i'} \in \{0, 1, 2, \ldots, k\}$ and $a_i \in \{0, 1\}$ therefore constraint (4.31) is unrestrictive (since $B_{ij} \geq 0$ is already enforced) unless $a_i = 1$ and $\sum_{i'=1}^{i-1} a_{i'} = 0$. This is only the case for $i = i_1$. Therefore $B_{i_1,1} = 1$.

$$\text{For } j = 2, \sum_{i'=1}^{i-1} b_{i'1} = \begin{cases} 0 & \text{if } i \leq i_1 \\ 1 & \text{if } i > i_1 \end{cases} \text{ (since } B_{i_1,1} = 1 \text{ and } B_{i,1} = 0 \, \forall i \neq i_1)$$

$$a_i = \begin{cases} 1 & \text{if } i \in \Gamma \\ 0 & \text{if } i \notin \Gamma \end{cases} \text{ and } \sum_{i'=1}^{i-1} B_{i'2} = \begin{cases} 0 & \text{if } i \leq i_2 \\ 1 & \text{otherwise} \end{cases} \text{ (since } B_{i2} = 0 \, \forall i \leq i_1)$$

Combining the above in (4.32) we get :

$$B_{i2} \geq a_i + \sum_{i'=1}^{i-1} B_{i',j-1} - \sum_{i'=1}^{i-1} B_{i',j} - 1 = \begin{cases} 1 + 0 - 0 - 1 = 0 & \text{if } i \in \mathcal{A}, i \leq i_1 \\ 1 + 1 - H - 1 = 1 - H & \text{if } i \in \mathcal{A}, i > i_2 \\ 1 + 1 - 0 - 1 = 1 & \text{if } i \in \mathcal{A}, i > i_1, i \leq i_2 \\ 0 + 0 - 0 - 1 = -1 & \text{if } i \notin \mathcal{A}, i \leq i_1 \\ 0 + 1 - 1 - 1 = -1 & \text{if } i \notin \mathcal{A}, i > i_2 \\ 0 + 1 - 0 - 1 = 0 & \text{if } i \notin \mathcal{A}, i > i_1, i \leq i_2 \end{cases}$$

Therefore when $i \in \mathcal{A}, i > i_1$ and $i \leq i_2$ (case 3) implies $B_{i_2,2} = 1$ (since $i \in \Gamma, i_1 < i \leq i_2 \Rightarrow i = i_2$). If $B_{i_2,2} = 1$ then $H = 1 \, \forall i > i_2$ and all cases (except case 3) do not restrict B_{ij}.
Repeating this process for $j = 3 \ldots k$ implies $B_{i_1,1} = B_{i_2,2} = \cdots = B_{i_k,k} = 1$ and all other $B_{ij} = 0$.

■

4.6 Computational performance of the formulations

In Section 4.4 we presented the exact ($\text{DP}_{\text{single}}$) model and in Section 4.5 we showed how to strengthen the model with symmetry breaking constraints and by changing the binary variable. This section presents computational results comparing the three formulations:

1. ($\text{DP}_{\text{single}}$),
2. ($\text{DP}_{\text{single}}$) + (4.28) and
3. ($\text{DP}_{\text{fewerBinary}}$).

The purpose of these results is to see how the different approaches compare with respect to computational time.

The data comes from max-cut problems from the BiqMac Library (Wiegele, 2007). Specifically the g05_80_i and pm1d_100_i examples, each with 10 instances. All models were formulated in MATLAB (2011) and solved with MOSEK ApS (2013).

The original solution X^* is the solution of solving the full size n problem over $\mathcal{C} \cap \mathcal{M}$. Computationally finding the deepest cut over a problem of this size is not yet possible. Therefore we consider a principal minors of size $\bar{n}$ and find the deepest cut of size k for this smaller problem. Results are shown for varying $\bar{n}$ and varying k in order to see how the models perform.

4.6.1 Comparison of the single level models

We begin by looking at the specific instance pm1d_100_i0 and comparing the three different formulations for different problem sizes (i.e. different values of $\bar{n}$ and k). Figure 4.1 shows the computational time (in seconds) vs the problem size ($\bar{n}$) for $k = 6$. Note that we only consider $k < \bar{n}$ and only run formulations ($\text{DP}_{\text{single}}$), ($\text{DP}_{\text{single}}$) + (4.28) on smaller examples. The results for $k = 5, \ldots, 9$ are similar. We observe that ($\text{DP}_{\text{fewerBinary}}$) is faster and able to solve larger problems. This observation is seen in all instances tested. The remaining results in this section focus on the ($\text{DP}_{\text{fewerBinary}}$) formulation.

4.6.2 Performance of $\text{DP}_{\text{fewerBinary}}$ formulation

Figure 4.2 shows the computational time to solve $\text{DP}_{\text{fewerBinary}}$ vs the problem size $\bar{n}$ for $k = 7$. The results are similar for all $k = 5, \ldots, 9$ and for the g05_80 instances. We observe that computational time increases as the size of the problem increases. This is as expected since the size of the problem (both in terms of variables and constraints) increases as n increases.

Figure 4.3 shows the computational time for the 10 instances of g05_80. This plot shows the instances for $k = 5, \ldots, 9$ and $n = 12$. The key observation is that computational time does not increase as k increases. This is most clearly seen in instances 0, 2, 4 and 6 where a different value of k gives the case which took the most ($k = 7, 8, 9$ and 6, respectively) and least ($k = 5, 9, 6$ & 7 (tied) and 8 respectively) time. The lack of pattern between computational time and k is similar for all $\bar{n}$ tested and both data sets.

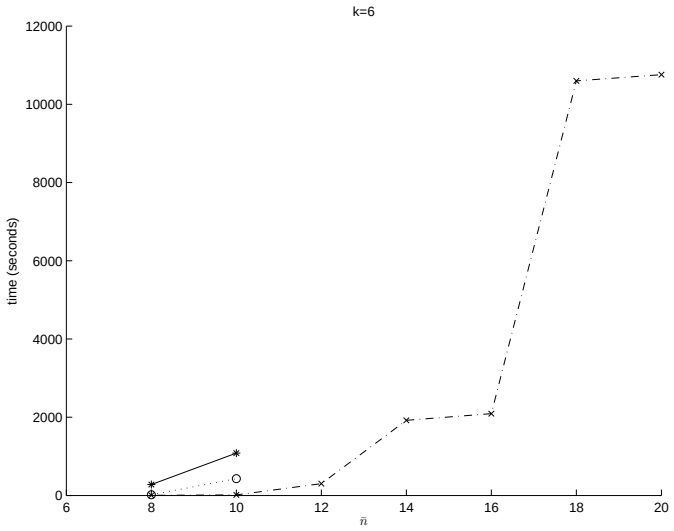

Figure 4.1 Comparison of computational time for formulations $\mathrm{DP}_{\mathrm{single}}$, $\mathrm{DP}_{\mathrm{single}}$ + (4.28) and $\mathrm{DP}_{\mathrm{fewerBinary}}$ for max-cut instance pm1d_100_i0 with $k = 6$

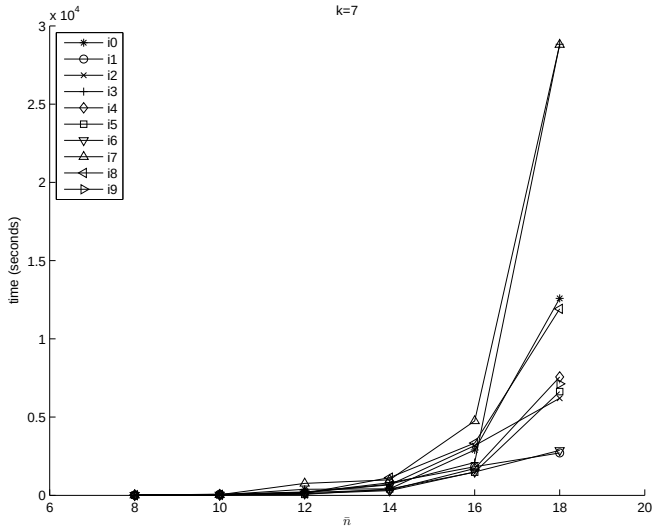

Figure 4.2 Computation time to solve $\mathrm{DP}_{\mathrm{fewerBinary}}$ for instances pm1d_100_i0,...,_i9 for $k = 7$.

4.7 Conclusion

Finding the most violated valid inequality from a family of cuts is called the maximally violated valid inequality problem (MVVIP). This chapter presented a bilevel model to solve the MVVIP for kPPCs (called the MVkPPCP). The bilevel model was reformulated to a single

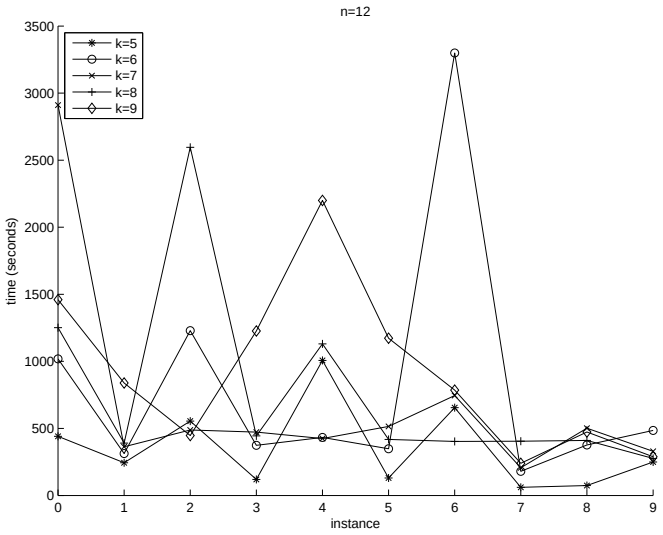

Figure 4.3 Computational time to solve $\mathrm{DP}_{\mathrm{single}}$ for max-cut instance g05_80 with $n = 12$.

level mixed integer second order cone problem so that is could be computationally solved in MOSEK. The model was further reformulated to include symmetry breaking constraints and additional binary variables. These changes both reduced the size of the enumeration tree and therefore sped up the solving time.

The focus of this chapter was the formulation of a model to solve the MVkPPCP. The major limitation of the model presented is that, even once reformulated the model can only be solved for small instances. In the next chapter we will abandon the idea of finding the most violated kPPC and focus on a practical implementation of a cutting plane method.

The distance-to-polytope model used to quantify the violated of a kPPC by calculating the shortest euclidean distance between a projected solution and the appropriately sized cut polytope will also be critical in the next chapter.

CHAPTER 5 K-PROJECTION POLYTOPE CONSTRAINTS IN A CUTTING PLANE ALGORITHM

5.1 Introduction

Cutting plane algorithms are used to tighten the relaxations of integer programs. In general cutting plane methods include a separation procedure to find violated cuts and a relaxation that is solved to (potentially) update the upper bound. The purpose of the cutting plane method presented in this chapter is to show how kPPCs can tighten upper bounds of large max-cut problems once all triangle inequalities are satisfied. Therefore the cutting plane method begins by focusing on triangle inequalities and then proceeds to kPPCs. Figure 5.1 outlines the cutting plane method that will be presented in this thesis.

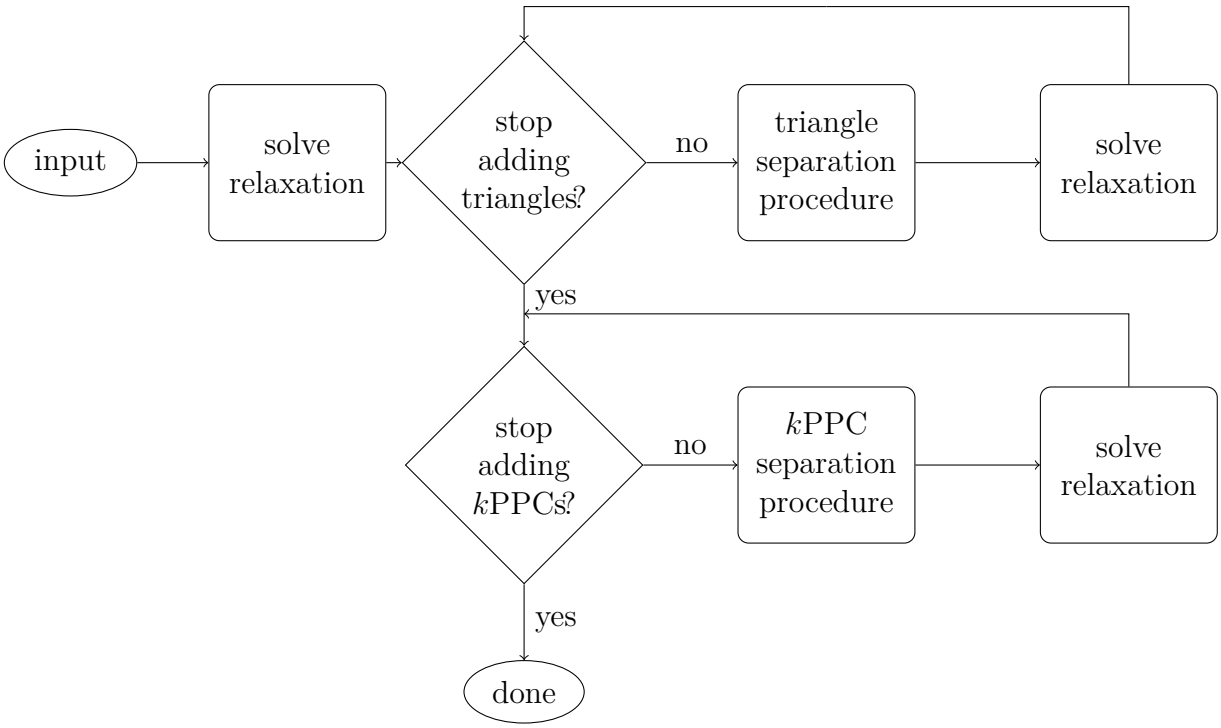

Figure 5.1 Overview of cutting plane method

The method is broken down into two stages based on the family of cuts found during the separation procedure. The first row of the diagram shows the triangle cutting plane stage and includes the initialization step. The second row of the diagram outlines the kPPC cutting plane stage. The difference lies mainly in the separation procedure since violated triangle

inequalities and violated kPPCs are identified differently. The details of the triangle and kPPC cutting plane stages are given in Sections 5.2 and 5.3, respectively. The relaxation is presented in this section, however we begin with some relevant notation.

5.1.1 Notation

For a given positive integer k, let

$$\square^k := \{I \,:\, \forall I \subseteq V, |I| = k\}$$

be the set of all induced subgraphs of size k. Therefore $|\square^k| = \binom{n}{k}$ where $|V| = n$.

Recall that a kPPC is defined for an induced subgraph, $I \in \square^k$ where $|I| = k$. Then for any set $\hat{\square}^k \subseteq \square^k$ let

$$\mathcal{PPC}(\hat{\square}^k) := \left\{ X : \; C\lambda^j = \text{triu}(X_I), \; \sum_{i=1}^{2^{k-1}} \lambda_i^j = 1, \; \lambda^j \geq 0, \; \forall I \in \hat{\square}^k \right\}$$

be the solution space where X satisfies all kPPCs defined by the induced subgraphs in $\hat{\square}^k$.

Let

$$\triangle := \left\{ \begin{array}{l} (I, c) : \forall \; I = (i_1, i_2, i_3) \in \square^3 \text{ and} \\ \qquad \forall \; (c_1, c_2, c_3) \in \{(-1,-1,-1), (-1,1,1), (1,-1,1), (1,1,-1)\} \end{array} \right\}$$

encode the set of all triangle inequalities. Namely each $(I, c) \in \triangle$ defines the triangle inequality $c_1 X_{i_1,i_2} + c_2 X_{i_1,i_3} + c_3 X_{i_2,i_3} \leq 1$.

Then for any $\hat{\triangle} \subseteq \triangle$ let

$$\mathcal{T}\text{ri}(\hat{\triangle}) := \left\{ X : \; c_1 X_{i_1,i_2} + c_2 X_{i_1,i_3} + c_3 X_{i_2,i_3} \leq 1, \; \forall \; ((i_1, i_2, i_3), (c_1, c_2, c_3)) \in \hat{\triangle} \right\}$$

be the solution space where X satisfies all triangles inequalities encoded in the set $\hat{\triangle}$.

The purpose of sets $\hat{\square}^k$ and $\hat{\triangle}$ is to encode the information needed to define the kPPCs and triangle inequalities within the relaxation. For simplicity we will refer to $\hat{\triangle}$ ($\triangle$) as a set of (all) triangle inequalities. Note that although it is not a set of inequalities it does provide all the necessary information to define triangle inequalities (specifically a set of 3

vertices $(i_1, i_2, i_3) \in \square^3$ and a set of coefficients (c_1, c_2, c_3)). Similarly for a given k, we will refer to $\hat{\square}^k$ ($\square^k$) as a set of (all) kPPCs even though it is technically a set of induced subgraphs.

We present the general relaxation in the next section.

5.1.2 SDP relaxation

For any $\hat{\triangle} \subseteq \triangle$ and $\hat{\square}^k \subseteq \square^k$, $\forall 5 \leq k << n$ the PPC-SDP optimization model is:

$$\begin{aligned} \text{(PPC-SDP)} \qquad \max \quad & \langle L, X \rangle \\ \text{s.t.} \quad & X \in \{X \ : \ \text{diag}(X) = e, X \succeq 0\} \\ & X \in \mathcal{T}\text{ri}(\hat{\triangle}) \\ & X \in \mathcal{PPC}(\hat{\square}^k) \qquad\qquad \forall 5 \leq k << n \end{aligned}$$

5.2 Triangle cutting plane stage

This section will present the triangle cutting plane stage and discuss the relevant details. However, we begin with a few words about the motivation behind our approach.

The goal of our method is to show how kPPCs can improve the bound over triangle inequalities. Therefore the idea is to get as much improvement as possible from triangle inequalities so that we can then see how much further kPPCs can improve the bound. This means, the primary focus of the triangle cutting plane stage is to get the best bound possible. The focus is not computational time.

The algorithm for the triangle cutting plane stage is:

initialize $t = 0$, $\hat{\triangle}_t = \emptyset$ and $\hat{\square}_t^k = \emptyset$, $\forall k$, tol=.001;
solve (PPC-SDP), denote X^0 as the optimal solution;
while *triangle stopping criteria is not met* **do**
  $t = t + 1$;
  set $(\hat{\triangle}_t)_{\text{viol}}$ (details discussed below);
  if $|(\hat{\triangle}_t)_{viol}| = \emptyset$ **then**
    stop
  else
    set
    $(\hat{\triangle}_t)_{\text{act}} := \left\{((i_1, i_2, i_3), (c_1, c_2, c_3)) \in \hat{\triangle}^{t-1} \ : \ |c_1 X^{t-1}_{i_1,i_2} + c_2 X^{t-1}_{i_1,i_3} + c_3 X^{t-1}_{i_2,i_3} - 1| \leq tol\right\}$;
    set $\hat{\triangle}_t = (\hat{\triangle}_t)_{\text{viol}} \cup (\hat{\triangle}_t)_{\text{act}}$;
    solve (PPC-SDP), denote X^t as the optimal solution;
    update upper bounds (details discussed below);
  end
end

Algorithm 1: Triangle cutting plane stage

5.2.1 Triangle cutting plane stage details

Initialization

To begin the triangle cutting plane stage the basic SDP relaxation is solved (i.e. the model (PPC-SDP) with $\hat{\triangle}_0 = \emptyset$ and $\hat{\square}_0^k = \emptyset \ \forall k$). We denote X^0 as the optimal solution of the relaxation and z^0 as the optimal objective value of the relaxation.

Stopping criteria

The triangle cutting plane stage stops when at least one of the following conditions is satisfied:
– there are no violated triangle inequalities ($(\hat{\triangle}_t)_{\text{viol}} = \emptyset$)
– the solver is unable to appropriately solve the model. For example this happens when SDPT3 (the solver used) has a termination code not equal to 0. These limitations of the solver are minor but exist and are described in Toh, Todd, and Tütüncü (1999a).

Violated constraints

For any solution X^{t-1} (with $t \geq 1$) and any triangle inequality represented by $((i_1, i_2, i_3), (c_1, c_2, c_3))$ let

$$\text{violation} := c_1 X^{t-1}_{i_1,i_2} + c_2 X^{t-1}_{i_1,i_3} + c_3 X^{t-1}_{i_2,i_3} - 1.$$

All $4\binom{n}{3}$ triangle inequalities are tested using the optimal solution X^{t-1} of the previous relaxation. Constraints that are satisfied by X^{t-1} have violation less than or equal to 0. Those constraints with positive violation are not satisfied (i.e. are violated constraints) and are potential cuts to include into the next iteration to tighten the model. To account for computational issues only those cuts with violation $\geq .001$ are considered in the selection stage. Then $(\hat{\triangle}_t)_{\text{viol}}$ is defined as the 1000 triangle inequalities with the largest violation.

Active constraints

Active constraints, also called binding constraints, are those constraints that are satisfied with equality at the optimal solution. The set $(\hat{\triangle}_t)_{\text{act}}$ is defined in Algorithm 1 and denotes the active triangle inequalities at iteration t. A tolerance of .001 is used to handle computational issues (i.e. if $|c_1 X^{t-1}_{i_1,i_2} + c_2 X^{t-1}_{i_1,i_3} + c_3 X^{t-1}_{i_2,i_3} - 1| \leq .001$ then $((i_1, i_2, i_3), (c_1, c_2, c_3)) \in (\hat{\triangle}_t)_{\text{act}}$). Active constraints at iteration t are also used in the relaxation at iteration $t+1$. Non-active constraints are not. This approach is common for triangle inequalities and has been shown to be successful. It reduces the number of triangle inequalities in the relaxation by only focusing on those that are at their limit at the solution.

Upper bound

The upper bound is updated if the current objective value is less than the current upper bound (PPC-SDP is a maximization problem). Since triangle inequalities are removed (i.e. those that are not active) it is possible for the current optimal objective value to be larger (i.e. worse) than the current upper bound. However although it is possible it is rarely the case.

5.3 kPPC cutting plane stage

This section will present the details of the kPPC cutting plane method, namely the part of the cutting plane stage where kPPCs are added. It is assumed that in the final iteration of

the triangle cutting plane stage there are no violated triangle inequalities. The algorithm for the kPPC cutting plane stage is:

From triangle cutting plane stage : t, X^t, $\hat{\triangle}_t$ and $\hat{\square}_t$;
while *kPPC stopping criteria is not met* **do**
  $t = t + 1$;
  choose $\bar{k}$;
  set $(\hat{\square}_t^{\bar{k}})_{\text{viol}}$ using algorithm 3;
  if $|(\hat{\square}_t^{\bar{k}})_{viol}| = \emptyset$ **then**
    stop
  else
    set $\Psi^k := \left\{ I \ : \ \forall I \subseteq \hat{I} \in \hat{\square}_t^{\bar{k}})_{\text{viol}} \text{ with } |I| = k, \forall \bar{k} > k \right\}$ for $k = 3$ and $k \geq 5$;
    set $\hat{\triangle}_t = \hat{\triangle}_{t-1} \setminus \Psi^3$;
    set $\forall k \geq 5$, $\hat{\square}_t^k = \begin{cases} \hat{\square}_{t-1}^k \cup (\hat{\square}_t^{\bar{k}})_{\text{viol}} & \text{if } k = \bar{k} \\ \hat{\square}_{t-1}^k \setminus \Psi^k & \text{otherwise} \end{cases}$;
    solve (PPC-SDP) denote X^t as the optimal solution;
    update upper bounds;
  end
end

Algorithm 2: kPPC cutting plane stage

5.3.1 kPPC cutting plane stage details

Redundant cuts

Redundant constraints are constraints in the optimization model that can be removed without changing the feasible region. Essentially they are constraints that are guaranteed to be satisfied because of other constraints in the model. For example in the following simple linear program:

$$\max \{x \ : \ x \geq 0, x \geq 2\}$$

the constraint $x \geq 0$ is redundant.

Recall that

$$\text{CUT}_3 \subseteq \text{CUT}_4 \subseteq \cdots \subseteq \text{CUT}_k \subseteq \text{CUT}_{k+1} \subseteq \cdots \subseteq \text{CUT}_n$$

therefore

$$\text{if } I_1 \subseteq I_2 \subseteq V \text{ and } X \in \mathcal{PPC}(I_2) \text{ then } X \in \mathcal{PPC}(I_1).$$

Meaning $X \in \mathcal{PPC}(I_1)$ is a redundant constraint. Therefore a triangle inequality or kPPC is a redundant constraint if it is defined on an induced subgraph that is a smaller induced subgraph of another kPPC. In the algorithm Ψ^k is the set of all smaller kPPCs. The set ensures that redundant triangle inequalities and kPPCs are not kept in the model. The following example illustrates this concept.

Example:

$$\begin{aligned}
&\text{Let } (\hat{\square}_t^5)_{\text{viol}} = \{(1,2,3,4,5),(2,3,4,5,6)\} \\
&\text{and } \hat{\triangle}_{t-1} = \{(1,2,3),(1,2,6),(1,3,6),(2,3,4),(3,4,5)\} \\
&\text{then } \Psi^3 := \left\{ \begin{array}{l} (1,2,3),(1,2,4),(1,2,5),(1,3,4),(1,3,5), \\ (1,4,5),(2,3,4),(2,3,5),(2,4,5),(3,4,5), \\ (2,3,4),(2,3,5),(2,3,6),(2,4,5),(2,4,6), \\ (2,5,6),(3,4,5),(3,4,6),(3,5,6),(4,5,6) \end{array} \right\} \\
&\text{and } \hat{\triangle}_t := \{(1,2,6),(1,3,6)\} \text{ since } (1,2,3),(2,3,4),(3,4,5) \in \hat{\triangle}_{t-1} \cap \Psi^3.
\end{aligned}$$

In this example three redundant triangle inequalities are removed between the $(t-1)^{\text{th}}$ and t^{th} iteration.

Choosing $\bar{k}$

For the results presented in the chapter the choice of $\bar{k}$ is fixed throughout the entire algorithm. Since triangle inequalities are completely satisfied $\bar{k}$=3 or 4 will not yield any violated kPPCs. Therefore we start with $\bar{k}$=5.

Stopping criteria

The PPC-cutting plane stage stops when at least one of the following conditions is satisfied:
- there are no violated kPPCs ($(\hat{\square}_t)_{\text{viol}} = \emptyset$)
- the solver is unable to appropriately solve the model. For example this happens when SDPT3 (the solver used) has a termination code not equal to 0. These limitations of the solver are minor but exist and are described in Toh, Todd, and Tütüncü (1999a).
- a maximum number of iterations is reached.

Upper bounds

The upper bound is updated if the current objective value is less than the current upper bound (recall we are solving a maximization problem). Since kPPCs are never removed it is not possible for the current objective value to be larger (i.e. worse) than the current upper bound. However it is possible for the upper bound to remain the same at two (or more) consecutive iterations if the objective value stays the same. Note that even in the case where the objective value stays the same the solution at consecutive iterations will always change. This is because the solution is infeasible once kPPCs that were violated by that solution are included in the model.

Violation

Complete enumeration is not used to identify violated kPPCs. The details of how violated kPPCs are found and selected is covered in Section 5.3.2. For now one must simply know that the set $(\hat{\square}^{\bar{k}})^t_{\text{viol}}$ contains at most $nWant$ PPCs of size $\bar{k}$.

5.3.2 Generating violated kPPCs algorithm

For any k there are $\binom{n}{k}$ PPCs to consider. For large problems it is impractical (if not impossible) to test all kPPCs (as we did for triangle inequalities). This section presents the algorithm used to find a set, $(\hat{\square}^k_t)_{\text{viol}}$, of violated kPPCs. Let $nWant$ be the parameter for the maximum number of kPPCs to add to the relaxation. This section presents the details of defining a set of at most $nWant$ violated kPPCs.

For a given $\hat{k}$ Algorithm 3 presents how the set $(\hat{\square}^{\hat{k}}_t)_{\text{viol}}$ is constructed. The algorithm contains three main parts :

1. construction of an induced subgraph, I, of size $\hat{k}$,
2. solving the distance-to-polytope problem to identify if the $\hat{k}$PPC for subgraph I is violated and
3. once a set of induced graphs corresponding to violated $\hat{k}$PPCs is found, at most $nWant$ of them are selected.

Prior to examining the final algorithm we define the following function for a given induced subgraph I of size k and the set of triangle inequalities from the previous iteration, $\hat{\triangle}^{t-1}$:

$$\text{triangleList} := \{\hat{I} \ : \ \hat{I} \subseteq I, \ |\hat{I}| = 3\} \cap \{\hat{I} \ : \ (\hat{I}, f) \in \hat{\triangle}^{t-1}\}$$

Namely, this function identifies the induces subgraphs of size 3 within a given induced subgraph of size k for which triangle inequalities were used in the previous iteration. If no such induced subgraphs exist the set is empty.

The algorithm is presented and then the details are discussed.

Input : X^{t-1}, k, $\hat{\triangle}_{t-1}$ and parameters *maxTime*, *tol*, *nWant*;
Output $(\hat{\square}_t^k)_{\text{viol}}$=a set of at most *nWant* violated kPPCs
initialize $\Phi = \emptyset$;
set $r = \begin{cases} 2 & \text{if } k \in \{5,6\} \\ 3 & \text{if } k \in \{7,8,9\} \end{cases}$;
while *runtime* $<$ *maxTime* **do**
  choose unique $\alpha_1, \alpha_2, \ldots, \alpha_r \subseteq \hat{\triangle}_{t-1}$
  set $I = \cup_{i=1}^{r} I_{\alpha_i}$ where $\triangle_{\alpha_i} = (I_{\alpha_i}, f_{\alpha_i})$;
  if $|I| = k$ **then**
    solve (D2P) for X_I^{t-1} to get d^*;
    if $d^* >$ *tol* **then**
      $\Phi = \Phi \cup (I, d^*)$;
    end
  end
end
Sort Φ so that $\hat{\Phi} := \left\{(I, d^*) \subseteq \Phi \ : \ d_i^* \geq d_{i+1}^*\right\}$;
if $|\hat{\Phi}| <$ *nWant* **then**
  $\hat{\square}^t = \hat{\Phi}$
else
  set $\omega = \min\{nWant, |\hat{\Phi}|\}$ and $i = 1$;
  set $(\hat{\square}_t^k)_{\text{viol}} = \{I_1 \ : (I_1, d_1) \in \hat{\Phi}\}$ and $\Gamma :=$ triangleList(I_1);
  while $(|(\hat{\square}_t^k)_{viol}| < \omega)$ *and* $(i < |\hat{\Phi}|)$ **do**
    $i = i + 1$;
    if $\Gamma \cap$ triangleList$(I_i) = \emptyset$ **then**
      $(\hat{\square}_t^k)_{\text{viol}} = (\hat{\square}_t^k)_{\text{viol}} \cup I_i$;
    end
  end
end

Algorithm 3: Generation and selection of violated kPPCs

We begin with a brief outline of the algorithm:
– The input X^{t-1} and $\hat{\triangle}_{t-1}$ come from the final stage of the triangle cutting plane stage.

The parameter *maxTime* limits the amount of time that the algorithm looks for violated kPPCs at each iteration, *tol*=.001 and is used to approximate 0 and *nWant* is the maximum number of violated kPPCs one wants to select.

- The first while loop is the generation stage where induced subgraphs are constructed and tested to determine if they form a violated kPPC.
- The set Φ is sorted by distance (decreasing order) to give the sorted list $\hat{\Phi}$.
- The second while loop is the selection stage. The goal is to select at most *nWant* violated kPPCs. If fewer than *nWant* violated kPPCs were found in the generation stage then all the kPPCs in $\hat{\Phi}$ are selected. Otherwise kPPCs are selected according to the 'triangle coverage' method. This is discussed in the Selection section below.

Construction

The induced subgraph I is constructed from r randomly selected triangle inequalities in $\hat{\triangle}_{t-1}$. If $|I| = k$ then the (D2P) model (presented in Chapter 4) is used to determine the distance (d^*) between X_I^{t-1} and CUT_k. If the distance is nonzero (i.e. $X_I^{t-1} \notin \text{CUT}_k$) the result (induced subgraph I and distance d^*) is stored in the set Φ.

In section 5.3.3 this method is compared to a random generation method. The results show that generating kPPCs from $\hat{\triangle}_{t-1}$ produces more violated kPPCs than constructing PPCs randomly.

Selection

This stage of the separation procedure selects which of the violated kPPCs in the sorted list $\hat{\Phi}$ will actually be added to the relaxation. The set $(\hat{\square}_t^k)_{\text{viol}}$ denotes the set of violated kPPCs that have been selected. Recall that *nWant* is the maximum number of violated kPPCs one will select. If fewer than *nWant* violated kPPCs are found, one simply selects all of them. Otherwise we base the selection off of the triangles contained in them. We call this process 'triangle coverage'. It is explained in the following section.

Triangle coverage

Prior to describing this process we say a few words about the motivation. Typically cuts with the greatest violation are selected at each iteration. This is what we did for triangle inequalities. However, preliminary results (presented in Section 5.3.3) suggest that taking the most violated kPPCs was not as good as taking kPPCs that were generated from different triangle inequalities (recall that r randomly selected triangle inequalities were combined to

make each induced subgraph that was tested). Currently we do not know exactly why this happens.

The violated kPPC with the largest distance is selected. The induced subgraphs of size three contained in both the selected kPPC and $\hat{\triangle}^{t-1}$ are stored in Γ. Then we iteratively go through the list of violated kPPCs ($\hat{\Phi}$) and the most violated kPPC is selected if it does not contain an induced subgraph from $\hat{\triangle}_{t-1}$ already in Γ. The iterations stop once $nWant$ kPPCs are selected. An example is included to clarify this process.

Example:

Let $\hat{k} = 5, nWant = 3$

If $\hat{\triangle}_{t-1} := \{(1,2,5),(1,3,7),(2,3,4),(5,7,9),(5,6,9)\}$

and $\hat{\Phi} := \begin{pmatrix} ((2,3,4,5,6),0.06);((1,2,3,4,5),.05);((1,3,5,7,9),.04); \\ ((1,2,5,6,9),.035);((1,2,5,7,9),.02) \end{pmatrix}$

then for $i = 1 : (\hat{\square}_t^5)_{\text{viol}} = \{(2,3,4,5,6)\}$ and $\Gamma = \{(2,3,4),(3,5,6)\}$

$i = 2 : (\hat{\square}_t^5)_{\text{viol}} = \{(2,3,4,5,6)\}$ since $\text{triangleList}((1,2,3,4,5)) \cap \Gamma \neq \emptyset$

$i = 3 : (\hat{\square}_t^5)_{\text{viol}} = \{(2,3,4,5,6),(1,3,5,7,9)\}$ and $\Gamma = \{(1,3,7),(2,3,4),(3,5,6),(5,7,9)\}$

$i = 4 : (\hat{\square}_t^5)_{\text{viol}} = \{(2,3,4,5,6),(1,3,5,7,9),(1,2,5,6,9)\}$

and $\Gamma = \{(1,2,5),(1,3,7),(2,3,4),(3,5,6),(5,6,9),(5,7,9)\}$

then stop because $|(\hat{\square}_t^5)_{\text{viol}}| = nWant$.

5.3.3 Comparing generation and selection methods

In this section we consider two methods of generating violated kPPCs and two methods for selecting which violated kPPCs to include in the model. They are :

Generating violated kPPCS

This step results in a sorted list ($\hat{\Phi}$) of kPPCs of violation at least .001. The two methods of generating violated kPPCs are :

1. random Select $I \subseteq V$ so that $|I| = k$, then test if $X_I \in \text{CUT}_k$ by solving the distance-to-polytope model.
2. overlap triangles this method involves selecting enough triangle inequalities from $\hat{\square}_{t-1}^k$, combining them and if the combined size equals k then testing for violation by solving the distance-to-polytope model. The details were given in Algorithm 3.

Selecting violated kPPCs

Let $nWant$ be the parameter denoting the maximum number of kPPCs that will be

selected from the list of violated kPPCs ($\hat{\Phi}$). This step results in a set $(\hat{\square}_t^k)^{\text{viol}}$ of at most $nWant$ violated kPPCs that will be new cuts in the t^{th} iteration. The two methods of selecting violated kPPCs are :

1. best distance The $nWant$ kPPCs with the most violation are selected. If $|\hat{\Phi}| < nWant$ then all kPPCs are selected ($(\hat{\square}_t^k)^{\text{viol}} = \hat{\Phi}$).
2. triangle coverage No two selected kPPCs are generated from the same triangle inequality. The details are given in Algorithm 3.

The four methods are compared on the gkaf5 instance with $n = 500$, density=1 (see Section 5.4 for details about data sets) and with limited triangle inequalities (specifically after 10 iterations of the triangle cutting plane stage). Figure 5.1 shows the optimal objective value for the four methods (columns 4 to 7). The best (i.e. lowest) objective value is shown in bold. Columns 1 to 3 denote the parameters for the instance ($maxTime$, k and $nWant$).

Table 5.1 Comparing generation and selection methods for gkaf5 instance

time			random		overlap triangle	
(mins)	k	nAdd	best distance	triangle coverage	best distance	triangle coverage
5	5	50	201637.099	201618.446	**201601.262**	201602.352
5	5	100	201634.760	201617.002	201576.268	**201572.980**
5	6	50	201637.787	201620.268	**201593.429**	201593.523
5	6	100	201636.249	201612.953	201578.514	**201578.188**
5	7	50	201626.919	201606.989	201582.454	**201579.909**
5	7	100	201637.301	201607.113	201553.627	**201547.666**
5	8	50	201631.247	201608.991	**201560.366**	201561.397
5	8	100	201620.899	201587.218	201535.156	**201525.378**
10	5	50	201635.682	201619.938	201597.699	**201594.448**
10	5	100	201637.730	201607.166	201569.454	**201565.771**
10	6	50	201633.926	201615.462	201596.770	**201592.774**
10	6	100	201633.756	201595.055	201562.823	**201560.161**
10	7	50	201618.406	201606.445	201575.942	**201575.841**
10	7	100	201627.539	201595.319	201539.310	**201530.459**
10	8	50	201614.221	201601.036	201566.134	**201561.326**
10	8	100	201614.580	201579.087	201526.296	**201519.796**

Note that for each generation method and each case the selection process was run on the same list of violated kPPCs. Therefore, for example a list of violated PPCs with $k = 5(\hat{\Phi})$ were found using the random generation method with time limit of 5 minutes and then from that single list the two selection methods were compared. When the 50 PPCs with $k = 5$ with the largest violation ('best distance') were selected and included in the relaxation PPC-SDP

(with triangle inequalities from level 5) the optimal objective value was 201637.099 while when the 50 PPCs with $k = 5$ containing different triangles ('triangle coverage') the optimal objective value was 201618.446.

Generating kPPCs by overlapping triangles clearly results in better bounds compared to randomly generating kPPCs. We suspect this is because previously violated triangles suggest an area of the feasible region that could benefit from a tighter relaxation. However this is only our intuition.

When *maxTime*=5 mins (i.e. 5 mins is spent generating and testing kPPCs for violation) the results of the 'best' method for selecting violated kPPCs are mixed. However after 10 mins using the triangle coverage method results in a better bound in all instances tested. Recall that kPPCs act on the principal minors of the matrix X corresponding the induced subgraphs in $\hat{\square}_t^k$. Therefore the effect of the triangle coverage method is that many different parts of the matrix X are constrained. In essence 'spreading out' the kPPCs over the matrix X yields tighter relaxations.

An additional benefit of the triangle coverage method is that more triangle inequalities are redundant and removed from the model. While this has no effect on the bound (since the triangles are removed because the constraints are contained within the kPPCs) it does effect the size of the model. This quickens the computational time required to solve the model.

After considering these results we use the overlap triangle method for generating violated kPPCs and the triangle coverage method for selecting which kPPCs to include in the model in the kPPC cutting plane stage of our algorithm.

5.4 Computational Results

Triangle inequalities are strong and are typically able to make significant progress in tightening the relaxation. These computational results examine how kPPCs can be used to further improve the bound for max-cut problems with density equal to 1 of size 500 to 1000.

Test problems We use two types of test instances:

1. 'gkaf' test set: these are binary quadratic optimization problems with $n = 500$ from the BiqMac Library (Wiegele, 2007). We only look at the gkaf5 instance with density equal to 1. Note that this is the only large ($n \geq 500$) instance with density equal to 1 in the BiqMac Library.
2. 'cmc' test set: these are new randomly generated 'c'omplete 'm'ax-'c'ut instances of size 600 to 1000 generated with rudy. They are random complete graphs with

integer edge weights uniformly distributed from $[-75, 75]$ with density=1 and $n = 600, 700, 800, 900$ and 1000. The specific rudy input is:

```
rudy -rnd_graph 600 100 601 -random -75 75 601 >cmc_n600
rudy -rnd_graph 700 100 701 -random -75 75 701 >cmc_n700
rudy -rnd_graph 800 100 801 -random -75 75 801 >cmc_n800
rudy -rnd_graph 900 100 901 -random -75 75 901 >cmc_n900
rudy -rnd_graph 1000 100 1001 -random -75 75 1001 >cmc_n1000
```

where the seed for the random instance is given by 601, 701, ..., 1001.

Machine In our tests we use a Acer Aspire 4752 with 6 GB of memory running Windows 7. We implemented our algorithm in MATLAB R2011b (MATLAB, 2011) and use version 4.0 of SDPT3 (Toh et al., 1999b) to solve the SDP relaxations.

The following two sections examine results from gkaf5 and cmc_600 in turn. We hold off our conclusions until the final section.

5.4.1 Results for gkaf test instance

This section presents the results for the gkaf5 instance of size 500. Table 5.2 shows the optimal objective value for the kPPC cutting plane stage. The final triangle cutting plane stage (i.e. once there were no more violated triangle inequalities) is denoted as iteration 0. Columns 2 and 3 show the objective value and number of PPCs with $k = 5$ (respectively) at each iteration of the kPPC cutting plane algorithm. Columns 4-11 show similar results when $k = 6, 7$ and 8 in the cutting plane algorithm. Note that for these results the entire kPPC cutting plane stage uses the same value of k. Therefore four implementations of the PPC cutting plane stage are shown (namely when $k = 5, 6, 7$ and 8). For the same four implementations Table 5.3 shows the total iteration time and cputime (in minutes) for each iteration. Cputime refers to the time SDPT3 takes to solve the model (PPC-SDP) at that iteration. Iteration time includes generating and selecting violated kPPCs, formulation and solving (PPC-SDP), updating the bound (if necessary) and checking the stopping criteria.

Note that the triangle cutting plane method terminated after 17 iterations since SDPT3 was unable to successfully solve the problem (specifically the termination code was -5 i.e. "the Schur complement matrix becomes too ill-conditioned for further progress"). Even if only the single most violated triangle was added SDPT3 was still unable to solve the problem. It was at this point that we moved to the kPPC cutting plane stage to further improve the bound.

Table 5.2 Results of kPPC cutting plane stage for instance gkaf5 with $k = 5$ to 8

iter	$k = 5$			$k = 6$		
	objective value	# of kPPC	# of Δ	objective value	# of kPPC	# of Δ
0	200271.941	0	6295	200271.941	0	6295
1	200248.934	50	6191	200248.611	50	6192
2	200225.794	100	6090	200227.835	100	6088
3	200206.197	150	5989	200210.781	150	5985
4	200186.827	200	5887	200190.635	200	5881
5	200168.202	250	5786	200173.727	250	5777
6	200153.306	300	5686	200155.137	300	5669
7	200134.212	350	5586	200139.970	350	5565
8	200117.241	400	5484	200123.713	400	5460
9	200104.633	450	5384	200110.155	450	5358
10	200091.290	500	5284	200096.633	500	5254

iter	$k = 7$			$k = 8$		
	objective value	# of kPPC	# of Δ	objective value	# of kPPC	# of Δ
0	200271.941	0	6295	200271.941	0	6295
1	200240.795	50	6131	200233.905	50	6134
2	200213.803	100	5976	200200.391	100	5966
3	200189.602	150	5819	200167.878	150	5810
4	200164.375	200	5666	200140.823	200	5650

The parameters were: $maxTime$=10 mins, tol=.001, $nWant$=50.

Table 5.3 Time (mins) per iteration for kPPC cutting plane stage for instance gkaf5 with $k = 5$ to 8

	$k = 5$		$k = 6$	
iteration	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)
1	22.4	7.0	25.0	9.5
2	23.8	8.2	26.2	11.5
3	25.1	9.2	29.5	14.2
4	26.2	10.5	31.1	15.9
5	28.4	12.1	35.2	19.4
6	30.7	14.6	39.6	23.1
7	32.7	16.0	43.9	27.3
8	33.5	17.1	53.5	35.8
9	35.7	19.0	55.4	37.8
10	37.5	20.7	168.3	147.5

	$k = 7$		$k = 8$	
iteration	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)
1	23.8	9.5	25.4	12.3
2	27.7	13.0	31.1	17.7
3	31.8	16.6	40.2	26.1
4	37.1	21.6	49.4	34.9

5.4.2 Results for cmc test instances

This section includes results for one of the 5 cmc instances, specifically when $n = 600$. The remaining results are included in Appendix A. There are two key differences between these examples and the gkaf5 example previously examined. First the triangle cutting plane stage terminated when there were no more violated triangle inequalities. Second it was much more difficult to find violated kPPCs. As a result the parameter $maxTime$ was set to 20 minutes. $nWant$ remained equal to 50, however this limit was never reached. The algorithm added all the violated kPPCs that it found (violation $\geq .001$).

Table 5.4 shows the optimal objective value for the kPPC cutting plane stage and Table 5.5 shows the computational time. These tables follow the same format as in the previous section. The 7PPC cutting plane stage was terminated after 13 iterations due to the large cputime.

As previously mentioned the results for instances cmc 700 to 1000 are included in Appendix A. For $n = 1000$ we had to make a few changes to the algorithm. Firstly, it was not possible to check all triangle inequalities because of the shear number of induced subgraphs of size 3 (over 166 million). Therefore we randomly test 50 million induced subgraphs at each iteration. The four possible coefficient combinations were tested for each induced subgraph. Then, as we did previously, we added the 1000 inequalities that were most violated. This means when the algorithm terminated because no violated triangle inequalities were found it does not mean that there are no violated inequalities.

Secondly, the number of iterations was limited to 8 for $k = 5$ and 4 for $k = 6$. The computational time for each iteration was getting large (over 2 hours).

We did not test problems of size greater than 1000 as these issues with our implementation of the cutting plane algorithm would have only gotten worse.

Table 5.4 Optimal objective value of kPPC cutting plane stage for instance cmc_n600 with $k = 5, 6$ and 7

iter	$k=5$			$k=6$			$k=7$		
	objective value	# of kPPC	# of Δ	objective value	# of kPPC	# of Δ	objective value	# of kPPC	# of Δ
0	293606.893	0	6438	293606.893	0	6438	293606.893	0	6438
1	293605.040	7	6425	293600.064	21	6395	293593.166	38	6319
2	293603.670	21	6400	293593.358	45	6344	293585.613	70	6217
3	293601.489	47	6359	293585.960	58	6317	293577.836	112	6086
4	293596.295	71	6311	293580.992	84	6264	293571.615	140	6000
5	293595.141	90	6272	293571.474	108	6214	293564.547	184	5862
6	293592.596	120	6228	293563.000	134	6161	293554.475	231	5713
7	293590.911	140	6198	293558.247	165	6097	293547.530	262	5614
8	293588.463	179	6146	293549.375	194	6039	293541.614	298	5502
9	293585.864	214	6099	293543.830	212	6003	293535.937	337	5379
10	293584.240	246	6055	293537.030	240	5946	293531.326	376	5259
11	293582.690	268	6011	293529.377	260	5905	293525.916	419	5123
12	293581.881	306	5963	293522.263	282	5858	293520.244	453	5017
13	293580.021	328	5930	293515.400	311	5798	293515.297	492	4898
14	293578.703	364	5885	293511.573	338	5744	-	-	-
15	293578.437	371	5874	293504.553	363	5690	-	-	-

Table 5.5 Time (mins) per iteration for kPPC cutting plane stage for instance cmc_n600 with $k = 5, 6$ and 7

iter.	$k = 5$		$k = 6$		$k = 7$	
	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)
1	27.4	6.4	48.2	7.4	49.3	8.2
2	28.1	6.7	49.1	8.1	51.6	10.2
3	29.7	7.4	49.6	8.5	55.8	13.9
4	31.5	7.4	50.7	9.3	56.8	14.6
5	32.6	8.0	51.9	10.3	61.4	18.9
6	32.3	8.8	53.6	11.6	66.7	23.6
7	31.9	9.1	55.7	13.4	70.3	26.9
8	34.8	10.6	58.3	15.7	75.7	32.2
9	35.7	11.4	59.1	16.4	82.2	38.3
10	36.7	12.2	61.2	18.2	156.2	111.1
11	39.1	13.5	62.7	19.5	308.0	262.1
12	40.1	14.8	64.8	21.4	366.1	319.6
13	40.4	15.5	67.6	23.9	526.0	480.5
14	44.7	18.8	70.0	26.1	-	-
15	41.6	17.5	72.5	28.4	-	-

5.5 Conclusion

Table 5.6 summarizes the results. The first two columns identify the instance and the third gives the value of k. Let

$$\text{Improvement} := 100 \times \frac{z_{\text{last triangle}} - z_{\text{last } k\text{PPC}}}{z_{\text{last triangle}}}$$

where $z_{\text{last triangle}}$ is the optimal objective value at the last iterations where triangle inequalities were added and $z_{\text{last } k\text{PPC}}$ is the optimal objective value at the last iterations of the kPPC cutting plane stage. Column 5 gives the number of iterations in the kPPC cutting plane stage (this does not include iterations from the triangle cutting plane stage) and the total time (in minutes) of the entire kPPC cutting plane stage is given in the final column.

Table 5.6 Comparison of results for all instances

Data Set	n	k	Improvement (%)	# iterations	time (mins)
gkaf	5	5	0.090	12	378.4
	5	6	0.088	12	507.8
	5	7	0.054	4	120.4
	5	8	0.065	4	146.1
cmc	600	5	0.010	15	526.3
	600	6	0.035	15	874.8
	600	7	0.031	13	1926.1
	700	5	0.009	15	536.2
	700	6	0.024	15	853.1
	700	7	0.025	15	1032.7
	800	5	0.005	15	586.5
	800	6	0.016	15	926.9
	900	5	0.004	15	662.3
	900	6	0.013	15	956.7
	1000	5	0.002	8	615.9
	1000	6	0.001	3	323.8

The following observations can be made:

– kPPCs continue to improve the bound once all triangle inequalities are satisfied. Although the improvement is small the instances being tested are dense graphs. These represent the most difficult instances because efficient techniques for solving sparse graphs exist (Helmberg and Rendl, 1998).

– For the cmc instances the bound improves more the larger the value of k. This is as expected since $\text{CUT}_k \subseteq \text{CUT}_{k+1}$. For the gkaf instance this is not always the case. It is unclear as

to why this happens.
– Furthermore the time to solve the SDP increases as k increases. Since the number of equations in each kPPC is dependent on k this is also what one would expect.

This chapters provides an algorithm for a cutting plane method that includes kPPCs. The primary objective was to show that kPPCs can improve the bound once the benefit from triangle inequalities is exhausted. We presented a separation procedure for kPPCs based on combining triangle inequalities and testing them for violation using the distance-to-polytope model.

Five new test instances with density equal to 1 were created with rudy. The sizes range from 600 to 1000. These instances were created to test kPPCs on larger max-cut problems (both BiqMac and BiqCrunch present results for instances with $n \leq 500$).

The following are limitations of the current cutting plane algorithm:
– We use the same k for all iterations in the kPPC cutting plane stage. Alternative methods for deciding how and when to change the value of k should be considered.
– Since kPPCs are equalities, identifying active constraints is not relevant. The concept of when kPPCs as a whole are active or nearly active (i.e. when X_I is on or near the boundary of CUT_k) should be examined so that kPPCs that are no longer relevant can be removed from future iterations.

CHAPTER 6 CONCLUSION

This thesis has presented a new family of cuts called k-projection polytope constraints. kPPCs are defined for problems that satisfy the projection property (i.e. feasible solutions on the full graph and projections of feasible solutions onto induced subgraphs have the same structure). We considered two specific combinatorial problems: the max-cut problem and the stable set problem. Three concepts related to kPPCs were explored: a hierarchy of relaxations, an exact separation formulation for the MVVIP and a cutting plane implementation. Essentially all of these concepts address the issue of how an initial relaxation is tightened. The difference is just which kPPCs are added. Specifically, for any given k,

1. If all kPPCs are added to a relaxation the k^{th} level of the hierarchy is defined. As k increases the relaxations of the hierarchy get stronger. The n^{th} level of the hierarchy defines the exact problem.
2. For a given induced subgraph, the distance-to-polytope problem identifies if the kPPC is violated. Adding any violated kPPC to the relaxation will change the solution and may improve the bound. The most violated kPPC can be found by solving the MVkPPCP.
3. In practice, finding violated kPPCs can be done by combining triangle inequalities. When included as the separation procedure of a cutting plane algorithm kPPCs can continue to improve the bound even once the benefit of triangle inequalities is exhausted.

6.1 Advancement of knowledge

The following are the main advances made in this thesis:

– A new family of cuts called kPPCs was defined. This family differs from most cuts as it is a set of equalities, not inequalities.
– kPPCs define a hierarchy that is in the same matrix space as the original problem. This differs from other hierarchies for combinatorial problems which grow exponentially.
– A tractable model for the MVkPPCP was presented. Only a few families of cuts have similar models for the MVVIP.
– Upper bounds have been presented for large, dense max-cut instances. The bounds are an improvement over bounds obtained by just using triangle inequalities.

6.2 Limits and constraints

The following are the main limitations of kPPCs and the main drawbacks of the concepts presented in this thesis:

- kPPCs can only be defined for small k since they require the enumeration of all feasible solutions on an induced subgraph of size k. Specifically for the max-cut problem there are 2^{k-1} feasible solutions on an induced subgraph of size k.
- The hierarchy of kPPCs can only be explicitly enumerated for very small problems since the number of kPPCs at each level grows exponentially. Specifically there are $\binom{n}{k}$ kPPCs included in the k^{th} level of the hierarchy.
- The MVkPPCP can only be solved for small instances due to the size of the formulation. Specifically there are $\mathcal{O}(k^2n^2)$ constraints, 2 second order cones of size $1+\binom{k}{2}$ and $2^{k-1}+n$ binary variables in the final formulation.

6.3 Recommendations

The following are recommendations for future research concerning kPPCs:

- A cutting plane method that combines triangle inequalities and kPPCs of different sizes in a more dynamic way should improve the quality of the upper bounds and the computational time.
- A method, similar to the bundle method used for triangle inequalities, for identifying kPPCs that can be removed from the model without worsening the bound would greatly improve the computational time required to solve each SDP relaxation.
- It may be possible to extend kPPCs to problems that do not satisfy the projection property. As opposed to looking at all feasible solutions on an induced subgraph one may need to consider how to define the partial solutions that could exist.

REFERENCES

M. F. Anjos et J. B. Lasserre, éds., *Handbook on Semidefinite, Conic and Polynomial Optimization*, série International Series in Operations Research & Management Science. Springer-Verlag, 2012.

M. F. Anjos et A. Vannelli, "Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes", *INFORMS Journal on Computing*, vol. 20, no. 4, pp. 611–617, 2008.

M. F. Anjos et H. Wolkowicz, "Strengthened semidefinite relaxations via a second lifting for the max-cut problem", *Discrete Applied Mathematics*, vol. 119, no. 1, pp. 79–106, 2002.

D. Applegate, R. Bixby, V. Chvàtal, et W. Cook, "The traveling salesman problem : a computational study", 2006.

M. ApS, *TheMOSEKoptimization toolbox for MATLAB manual. Version 7.0 (Revision 141).*, 2013. En ligne : `http://docs.mosek.com/7.0/toolbox/index.html`

E. Balas, S. Ceria, et G. é. r. Cornuéjols, "A lift-and-project cutting plane algorithm for mixed 0-1 programs", *Mathematical programming*, vol. 58, no. 1-3, pp. 295–324, 1993.

F. Barahona, "On cuts and matchings in planar graphs", *Mathematical Programming*, vol. 60, no. 1-3, pp. 53–68, 1993.

F. Barahona et L. Ladanyi, "Branch and cut based on the volume algorithm : Steiner trees in graphs and max-cut", *RAIRO-Operations Research*, vol. 40, no. 01, pp. 53–73, 2006.

F. Barahona et A. R. Mahjoub, "On the cut polytope", *Math. Programming*, vol. 36, no. 2, pp. 157–173, 1986.

F. Barahona, M. Grötschel, M. Jünger, et G. Reinelt, "An application of combinatorial optimization to statistical physics and circuit layout design", *Operations Research*, vol. 36, no. 3, pp. 493–513, 1988.

F. Barahona, M. Jünger, et G. Reinelt, "Experiments in quadratic 0–1 programming", *Mathematical Programming*, vol. 44, no. 1-3, pp. 127–137, 1989.

P. Belotti, J. C. Góez, I. Pólik, T. K. Ralphs, et T. Terlaky, "Disjunctive conic cuts for mixed integer second order cone optimization", *preparation for publication*, 2013.

P. Bonami, G. Cornuéjols, S. Dash, M. Fischetti, et A. Lodi, “Projected chvátal–gomory cuts for mixed integer linear programs”, *Mathematical Programming*, vol. 113, no. 2, pp. 241–257, 2008.

A. Caprara et M. Fischetti, “{0, 1/2}-chvátal-gomory cuts”, *Mathematical Programming*, vol. 74, no. 3, pp. 221–235, 1996.

——, “Branch-and-cut algorithms”, *Annotated bibliographies in combinatorial optimization*, pp. 45–64, 1997.

A. Caprara et A. N. Letchford, “On the separation of split cuts and related inequalities”, *Mathematical Programming*, vol. 94, no. 2-3, pp. 279–294, 2003.

A. Caprara, M. Fischetti, et A. N. Letchford, “On the separation of maximally violated mod-k cuts”, *Mathematical Programming*, vol. 87, no. 1, pp. 37–56, 2000.

V. Chvátal, “Edmonds polytopes and a hierarchy of combinatorial problems”, *Discrete mathematics*, vol. 4, no. 4, pp. 305–337, 1973.

W. Cook, R. Kannan, et A. Schrijver, “Chvátal closures for mixed integer programming problems”, *Mathematical Programming*, vol. 47, no. 1-3, pp. 155–174, 1990.

G. Dantzig, R. Fulkerson, et S. Johnson, “Solution of a large-scale traveling-salesman problem”, *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

C. Delorme et S. Poljak, “Laplacian eigenvalues and the maximum cut problem”, *Mathematical Programming*, vol. 62, no. 1-3, pp. 557–574, 1993.

M. Deza et M. Laurent, *Geometry of cuts and metrics*. Springer Science & Business Media, 1997, vol. 15.

——, *Geometry of Cuts and Metrics*, série Algorithms and Combinatorics. Berlin : Springer-Verlag, 1997, vol. 15.

M. Deza, M. Laurent, et S. Poljak, “The cut cone iii : on the role of triangle facets”, *Graphs and Combinatorics*, vol. 8, no. 2, pp. 125–142, 1992.

F. Eisenbrand, “Note–on the membership problem for the elementary closure of a polyhedron”, *Combinatorica*, vol. 19, no. 2, pp. 297–300, 1999.

I. Fischer, G. Gruber, F. Rendl, et R. Sotirov, “Computational experience with a bundle approach for semidefinite cutting plane relaxations of Max-Cut and equipartition”, *Math. Programming*, vol. 105, no. 2-3, Ser. B, pp. 451–469, 2006.

M. X. Goemans, “Semidefinite programming in combinatorial optimization”, *Math. Programming*, vol. 79, pp. 143–161, 1997.

M. X. Goemans et D. P. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”, *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

R. E. Gomory, “An algorithm for integer solutions to linear programs”, *Recent advances in mathematical programming*, vol. 64, pp. 260–302, 1963.

——, “On the relation between integer and noninteger solutions to linear programs”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 53, no. 2, p. 260, 1965.

V. P. Grishukhin, “All facets of the cut cone $\mathbf{C}_n$ for $n = 7$ are know.” *European Journal of Combinatorics*, vol. 11(2), pp. 115–117, 1990.

M. Grötschel, L. Lovász, et A. Schrijver, “The ellipsoid method and its consequences in combinatorial optimization”, *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.

P. Hansen, “Methods of nonlinear 0-1 programming”, *Annals of Discrete Mathematics*, vol. 5, pp. 53–70, 1979.

C. Helmberg et F. Rendl, “Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes”, *Mathematical Programming*, vol. 82, no. 3, pp. 291–315, 1998.

C. Helmberg, F. Rendl, R. J. Vanderbei, et H. Wolkowicz, “An interior-point method for semidefinite programming”, *SIAM J. Optim.*, vol. 6, no. 2, pp. 342–361, 1996.

R. M. Karp, *Reducibility among combinatorial problems.* Springer, 1972.

K. Krishnan et T. Terlaky, “Interior point and semidefinite approaches in combinatorial optimization”, dans *Graph theory & combinatorial optimization.* Springer, 2005, pp. 101–157

.

N. Krislock, J. Malick, et F. é. é. Roupin, “Improved semidefinite bounding procedure for solving max-cut problems to optimality”, *Mathematical Programming*, vol. 143, no. 1-2, pp. 61–86, 2014.

J. B. Lasserre, “An explicit exact sdp relaxation for nonlinear 0-1 programs”, dans *Integer Programming and Combinatorial Optimization*. Springer, 2001, pp. 293–303.

——, “An explicit equivalent positive semidefinite program for nonlinear 0-1 programs”, *SIAM J. Optim.*, vol. 12, no. 3, pp. 756–769 (electronic), 2002.

T. Lengauer, *Combinatorial algorithms for integrated circuit layout*. John Wiley & Sons, Inc., 1990.

F. Liers, M. Jünger, G. Reinelt, et G. Rinaldi, “Computing exact ground states of hard ising spin glass problems by branch-and-cut”, *New Optimization Algorithms in Physics*, pp. 47–68, 2004.

A. Lodi, T. K. Ralphs, et G. J. Woeginger, “Bilevel programming and the separation problem”, *Mathematical Programming*, pp. 1–22, 2012.

L. Lovász, “On the Shannon capacity of a graph”, *IEEE Trans. Inform. Theory*, vol. 25, no. 1, pp. 1–7, 1979.

——, “Semidefinite programs and combinatorial optimization”, dans *Recent advances in algorithms and combinatorics*, série CMS Books Math./Ouvrages Math. SMC. New York : Springer, 2003, vol. 11, pp. 137–194.

L. Lovász et A. Schrijver, “Cones of matrices and set-functions and 0-1 optimization”, *SIAM Journal on Optimization*, vol. 1, no. 2, pp. 166–190, 1991.

H. Marchand, A. Martin, R. Weismantel, et L. Wolsey, “Cutting planes in integer and mixed integer programming”, *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 397–446, 2002.

MATLAB, *version 7.10.0 (R2011b)*. Natick, Massachusetts : The MathWorks Inc., 2011.

J. E. Mitchell, “Branch-and-cut algorithms for combinatorial optimization problems”, *Handbook of Applied Optimization*, pp. 65–77, 2002.

——, “Polynomial interior point cutting plane methods”, *Optimization Methods and Software*, vol. 18, no. 5, pp. 507–534, 2003.

G. L. Nemhauser et L. A. Wolsey, *Integer and combinatorial optimization*. Wiley New York, 1988, vol. 18.

M. Padberg et G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems", *SIAM review*, vol. 33, no. 1, pp. 60–100, 1991.

S. Poljak et F. Rendl, "Solving the max-cut problem using eigenvalues", *Discrete Applied Mathematics*, vol. 62, no. 1, pp. 249–278, 1995.

S. Poljak et Z. Tuza, "The expected relative error of the polyhedral approximation of the max-cut problem", *Operations Research Letters*, vol. 16, no. 4, pp. 191–198, 1994.

——, "Maximum cuts and large bipartite subgraphs", *DIMACS Series*, vol. 20, pp. 181–244, 1995.

F. Rendl, G. Rinaldi, et A. Wiegele, "Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations", *Math. Programming*, vol. 121, no. 2, p. 307, 2010.

H. D. Sherali et W. P. Adams, "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems", *SIAM Journal on Discrete Mathematics*, vol. 3, no. 3, pp. 411–430, 1990.

K. C. Toh, M. J. Todd, et R. H. Tütüncü, "Sdpt3 — a matlab software package for semidefinite programming, version 1.3", *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 545–581, 1999. DOI : 10.1080/10556789908805762. En ligne : http://dx.doi.org/10.1080/10556789908805762

K.-C. Toh, M. J. Todd, et R. H. Tütüncü, "Sdpt3—a matlab software package for semidefinite programming, version 1.3", *Optimization methods and software*, vol. 11, no. 1-4, pp. 545–581, 1999.

A. Wiegele, *Nonlinear optimization techniques applied to combinatorial optimization problems*. na, 2006.

——, "Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size". Tech. rep., Alpen-Adria-Universit at Klagenfurt, Klagenfurt, Austria, 2007.

ANNEXE A CUTTING PLANE METHOD RESULTS FOR CMC INSTANCES

Table A.1 Time for kPPC cutting plane stage for instance cmc_n700 with $k = 5, 6$ and 7

iteration	$k = 5$		$k = 6$		$k = 7$	
	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)
1	29.6	8.3	49.2	8.4	49.7	8.6
2	30.3	8.4	49.4	8.3	50.7	9.3
3	30.9	8.9	51.3	10.1	53.3	11.7
4	33.1	9.5	52.1	10.7	55.9	13.9
5	35.8	10.8	53.0	11.3	57.1	14.8
6	33.2	10.5	54.1	12.2	59.7	17.1
7	34.5	11.1	55.1	13.0	62.1	19.4
8	35.6	11.8	56.7	14.4	64.2	21.4
9	36.1	12.4	57.5	15.0	67.4	24.3
10	36.5	13.0	59.2	16.6	67.9	24.7
11	39.4	15.1	60.3	17.4	72.2	28.5
12	40.4	16.1	61.7	18.6	75.1	31.3
13	41.3	17.1	63.0	19.8	78.8	34.7
14	39.9	15.9	64.8	21.3	85.5	41.2
15	39.7	16.1	65.6	22.0	132.9	87.4

Table A.2 Results for kPPC cutting plane stage for instance cmc_n700 with $k = 5, 6$ and 7

iter	$k=5$			$k=6$			$k=7$		
	objective value	# of kPPC	# of Δ	objective value	# of kPPC	# of Δ	objective value	# of kPPC	# of Δ
0	350675.242	0	6811	350675.242	0	6811	350675.242	0	6811
1	350672.138	8	6796	350670.126	10	6790	350667.132	15	6765
2	350668.687	22	6772	350664.181	24	6761	350659.291	34	6705
3	350666.244	37	6745	350657.411	46	6715	350653.935	68	6598
4	350661.532	75	6694	350650.418	59	6688	350650.873	88	6536
5	350657.239	98	6648	350645.574	75	6656	350640.893	115	6451
6	350655.835	113	6627	350638.080	97	6611	350632.128	134	6394
7	350654.627	140	6588	350631.150	114	6576	350626.354	156	6325
8	350652.685	167	6545	350624.511	134	6534	350622.081	173	6273
9	350652.550	187	6518	350618.513	149	6503	350618.383	195	6205
10	350650.534	204	6490	350614.086	168	6464	350615.998	214	6144
11	350649.003	228	6453	350610.463	188	6424	350608.919	241	6062
12	350647.526	250	6423	350605.076	206	6388	350601.838	260	6002
13	350645.925	268	6394	350599.716	227	6345	350596.736	282	5933
14	350645.102	280	6373	350594.648	248	6301	350593.186	305	5862
15	350644.254	285	6362	350591.914	261	6274	350589.266	332	5778

Table A.3 Results for kPPC cutting plane stage for instance cmc_n800 with $k = 5$ and 6

iter.	$k = 5$			$k = 6$		
	objective value	num of kPPCs	num of triangles	objective value	num of kPPCs	num of triangles
0	456272.381	0	7777	456272.381	0	7777
1	456270.843	11	7759	456266.182	20	7736
2	456269.971	20	7742	456262.071	34	7704
3	456269.754	24	7734	456257.293	47	7678
4	456268.097	43	7705	456252.283	59	7654
5	456264.926	63	7677	456247.962	72	7627
6	456263.028	89	7640	456242.642	91	7588
7	456262.273	107	7613	456236.939	105	7559
8	456261.466	119	7596	456231.814	127	7513
9	456259.175	160	7551	456225.700	143	7480
10	456256.361	186	7518	456217.931	155	7453
11	456256.358	187	7516	456212.545	169	7425
12	456255.996	204	7491	456208.045	178	7407
13	456254.251	217	7470	456204.000	196	7370
14	456253.840	223	7460	456201.513	207	7348
15	456251.561	247	7429	456198.383	224	7313

	$k = 5$		$k = 6$	
iteration	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)
1	33.6	11.7	54.3	12.8
2	33.7	11.7	55.1	13.4
3	33.4	11.8	56.2	14.3
4	35.2	12.3	56.9	14.9
5	37.4	14.3	57.8	15.6
6	37.8	14.3	58.9	16.5
7	38.2	14.8	60.0	17.4
8	38.5	15.3	61.5	18.6
9	42.8	17.2	62.8	19.7
10	42.5	17.7	63.5	20.3
11	41.2	17.9	66.0	22.3
12	42.9	18.3	66.6	23.1
13	42.4	18.1	67.8	24.0
14	42.3	18.3	69.0	25.1
15	44.6	19.3	70.5	26.5

Table A.4 Results for kPPC cutting plane stage for instance cmc_n900 with $k = 5$ and 6

iter.	$k = 5$			$k = 6$		
	objective value	num of kPPCs	num of triangles	objective value	num of kPPCs	num of triangles
0	519387.145	0	8251	519387.145	0	8251
1	519386.077	7	8237	519378.422	10	8231
2	519382.853	18	8216	519375.247	18	8215
3	519381.590	31	8195	519371.716	30	8190
4	519381.237	43	8177	519362.916	40	8168
5	519379.969	55	8160	519358.567	55	8137
6	519379.906	58	8154	519353.542	70	8107
7	519377.085	92	8111	519351.269	88	8071
8	519375.442	115	8090	519344.874	102	8042
9	519372.655	151	8044	519342.112	115	8016
10	519370.395	171	8020	519337.927	132	7982
11	519368.355	184	7999	519332.441	149	7947
12	519367.311	229	7953	519328.853	160	7925
13	519366.248	257	7914	519325.205	174	7897
14	519364.388	276	7886	519322.231	187	7871
15	519363.966	282	7874	519318.689	206	7833

iteration	$k = 5$		$k = 6$	
	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)
1	36.6	14.4	56.7	14.7
2	36.7	14.1	57.2	15.1
3	37.5	14.7	57.8	15.5
4	38.1	15.1	58.6	16.1
5	38.8	15.7	59.2	16.6
6	38.3	15.6	61.8	19.0
7	42.5	17.6	63.2	20.1
8	43.1	18.5	64.4	21.1
9	45.2	19.7	66.8	23.3
10	45.8	20.8	67.1	23.4
11	46.7	22.0	66.9	23.0
12	51.5	24.5	67.6	23.6
13	53.4	27.2	68.6	24.4
14	53.7	27.6	69.6	25.2
15	54.3	28.9	71.0	26.3

Table A.5 Results for kPPC cutting plane stage for instance cmc_n1000 with $k = 5$ and 6

iter.	$k = 5$			$k = 6$		
	objective value	num of kPPCs	num of triangles	objective value	num of kPPCs	num of triangles
0	629940.170	0	12025	629940.170	0	12025
1	629938.675	10	11989	629938.355	11	12005
2	629938.434	16	11973	629934.157	26	11978
3	629937.711	20	11962	629931.983	64	11917
4	629936.748	27	11947	-	-	-
5	629935.080	35	11927	-	-	-
6	629934.038	41	11910	-	-	-
7	629932.424	47	11897	-	-	-
8	629929.503	71	11844	-	-	-

	$k = 5$		$k = 6$	
iteration	iteration time (mins)	solver cpu time (mins)	iteration time (mins)	solver cpu time (mins)
1	65.2	41.2	83.3	39.9
2	66.9	42.7	88.1	44.4
3	67.0	43.0	152.4	107.2
4	65.6	41.3	-	-
5	69.1	44.5	-	-
6	71.5	47.0	-	-
7	73.2	48.3	-	-
8	137.5	110.0	-	-