Outer approximation for semidefinite programs and a vector clock problem

Thesis by

Daniel de Roux

In Partial Fulfillment of the Requirements for the
Degree of
Algorithms, Combinatorics and Optimization

CARNEGIE MELLON UNIVERSITY
Pittsburgh, Pennsylvania

May 2024

Daniel de Roux
ORCID: 0000-0003-4854-6785

ACKNOWLEDGEMENTS

First and foremost, I wish to thank my advisor, R.Ravi, for these 5 years of dedication to my work and growth as a person and researcher. Thank you for always being available for me and sitting long hours to check details, proofs, and ideas. Your sense of dedication and love for mathematics will always be an inspiration for me. Thanks to Fatma Kılınç-Karzan, for her guidance and willingness to teach me about what ultimately interested me the most during my PhD studies: Conic optimization. Thanks as well for being part of my Thesis committee, along with Ignacio Grossmann, R.Ravi, and Nihar Shah, all of who I thank as well for agreeing to read this manuscript. Special thanks to my two collaborators Bob Carr and Da Qi Chen. I wish to thank my close friends from the bottom of my heart, without whom I could not have successfully accomplished this endeavor. Thank you, Mateo, Will, Alex, Nilsu, Sofia, David, Em, Laura, Georgia, Daniel, Santiago, Kevin, Zhara, Sae, Gokcen, Ilayda, David (Bermeo), Diego, Maripi, and Guille. I hope Pittsburgh was just the beginning of our shared adventures. Thanks to my innumerable friends that I hold dear and made me company from far away. Thanks to Bananin, Sebastian, Santiago, Juan Pablo, Maria, Daniela, Maria Camila, Juliana, and Cami. Last but not least, thanks to my family, Piedad, Carlos Vicente, Nicolas, and Matias for the love, patience through difficult times, and for always believing in me.

ABSTRACT

In this thesis, we study linear outer approximations of semidefinite programs (SDPs) and extend these ideas to build algorithms that solve binary SDPs arising from quadratically constrained quadratic binary problems. We conclude by introducing a multi-commodity vector clock problem and deriving a structural result for it.

Chapter 1 introduces our generic technique to obtain linear relaxations of semidefinite programs with provable guarantees based on the commutativity of the constraint and the objective matrices. We study conditions under which the optimal value of the SDP and the proposed linear relaxation match, which we then relax to provide a flexible methodology to derive strong linear relaxations.

Chapter 2 introduces a spectral second-order outer approximation algorithm to solve to optimality integer semidefinite programs that are themselves exact formulations of binary quadratically constrained quadratic problems. Our approach fundamentally builds on the results of the previous chapter.

Chapter 3 considers rumor spreading problems in undirected graphs, generalizing the minimum broadcast time problem to the multi-commodity case. We also consider its extension to an infinite horizon version to minimize information latencies captured in a vector clock model. We show that the multi-commodity version of these problems on general graphs have locally periodic schedules that are within a poly-logarithmic factor of optimal by studying the properties of a non convex relaxation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOTATION

We denote the set of square, real, $n \times n$ symmetric matrices by $\mathbb{S}^n$ and the set of positive semidefinite matrices by $\mathbb{S}^n_+$. By convention, we assume that semidefinite matrices are always symmetric. We denote the Löwner order of symmetric matrices by $\succeq$. Hence, the notation $X \succeq Y$ means $X - Y \in \mathbb{S}^n_+$. In particular, $X \succeq 0$ indicates that X is positive semidefinite (PSD). For an integer $k \in \mathbb{N}$, $[n]$ denotes the set of natural numbers $\{1, \ldots, k\}$. We denote the cardinality of a set I by $|I|$. We denote by $e_1, \ldots, e_n$ the standard basis of $\mathbb{R}^n$ and the $n \times n$ identity matrix by I_n. For a symmetric matrix W we let $\lambda_1(W) \geq \lambda_2(W) \geq \cdots \geq \lambda_n(\mathrm{W})$ be its eigenvalues. When the matrix is clear from the context, we drop the terms in parentheses and simply write $\lambda_1 \geq \cdots \geq \lambda_n$. For $A \in \mathbb{S}^n$ we write $tr(A)$ for the trace of A: $tr(A) = \sum_{i=1}^n A_{ii}$ and write $\|A\|_F$ to denote the Frobenius norm of A: $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n A_{ij}^2}$. The ℓ_1 norm of A is given by $\|A\|_1 = \sum_{i,j} |A_{ij}|$. We denote by $\langle \cdot, \cdot \rangle$ the usual Frobenius inner product of two matrices in $\mathbb{S}^n$, recalling that for two matrices $A, B \in \mathbb{S}^n$, $\langle A, B \rangle = tr(A^T B) = tr(AB)$. We denote by $\vec{1}$ the vector of all ones in $\mathbb{R}^n$ and by J the matrix of all ones. If A is a matrix, we denote by $diag(A)$ the vector given be the diagonal of A. If u is a vector, $diag(u)$ denotes the matrix with u on its diagonal. We denote by $\mathcal{E}(A)$ an arbitrary orthonormal basis consisting of eigenvectors of A. In particular, if $A \in \mathbb{S}^n$ and $\mathcal{E}(A) = \{v_1, \ldots, v_n\}$ then we have $A = \sum_{i=1}^n \lambda_i v_i v_i^\top$ [78]. Finally, given a weighted graph G we denote respectively the value of the max cut of G, the adjacency matrix, the number of edges and the laplacian matrix by $mc(G), W(G)$, m and $\mathcal{L}(G)$. If G is clear from the context, we drop the dependency on G and simply write mc, W, m and $\mathcal{L}$.

INTRODUCTION

Is linear programming fundamentally weaker than semidefinite programming? Linear programming consists of minimizing a linear function over the intersection of an affine plane and the non-negative orthant. In contrast, semidefinite programming minimizes a linear function over the cone of positive semidefinite matrices. Semidefinite programs (SDPs) are able to model any linear program (LP), while also being able to capture many convex non-linear problems, e.g., the minimization of the largest eigenvalue of a matrix. Consequently, the family of LPs is strictly contained in the family of SDPs. However, delving deeper into this comparison reveals additional nuances worthy of discussion. Since any convex set can be expressed as the intersection of a set of hyperplanes, it follows, nominally, that any convex optimization problem can be approximated with a linear program if a description of the hyperplanes is known. This is precisely the case of semidefinite-representable convex regions, due to the fact that a matrix is positive semidefinite if all of the quadratic forms it defines are non-negative.

After decades of research in this field, the broad conclusion is that in fact linear programming is strictly weaker than semidefinite programming. For example, see [26, 54]. A manifestation of this phenomenon appears in the field of approximation algorithms in the setting of the maximum cut problem. Here one is given a graph and must split the set of vertices into two disjoint sets in a way that maximizes the number of edges crossing. This problem is NP-hard, and linear programs that approximate it with a factor better than 2 require an exponential number of constraints [32, 33, 89, 152]. In sharp contrast, there is a polynomial time approximation algorithm based on semidefinite programming achieving a ratio of ~ 1.13 [64]. A second manifestation appears in terms of the somewhat poor performance of LP-based outer approximation algorithms to solve semidefinite programs as they are generally slow and progress usually stalls [106]. All in all, it seems that the question of approximating semidefinite representable convex regions with polyhedra has fallen out of favour.

In this thesis, we make the case that there is a fruitful setting in which it is worthwhile to revisit the question of approximating a semidefinite program with linear relaxations. Our motivation stems from quadratically constrained binary quadratic problems, a fundamental class of optimization problems. In part, their importance comes from the fact that any continuous function can be approximated arbitrarily well (in a compact set) by a polynomial of arbitrary degree which, in turn, can be expressed with a quadratic expression by introducing additional variables and quadratic constraints [60].

Thus, binary quadratically constrained quadratic problems (BQCQPs) are roughly as expressive as binary nonlinear problems [60]. In fact, they capture problems from many different fields, such as combinatorial optimization and computer science [23, 48, 52, 100, 131], machine learning [57, 107, 127], chemical engineering, as seen in [21] and the references therein, portfolio optimization [22, 43, 139] (although most of these problems require reformulating an integer problem to a binary one), and all binary linear and polynomial optimization problems. Examples of problems captured by binary QCQPs are the maximum cut problem, k-cluster problems, the k-partition problem, the binary regression problem, the quadratic assignment problem, the stable set number, the quadratic knapsack problem, the chromatic number, and the quadratic set cover problem. Solving BQCQPs to optimality is hard both in theory [130] and in practice, even for moderately sized instances [60].

Very recently, exploiting results from positive semidefinite matrices with entries in $0, -1, 1$, de Meijer and Sotirov showed in [110] that BQCQPs can be reformulated as binary semidefinite programs (BSDPs), opening a new avenue to solve arbitrary problems of the former class. In [103], Lubin et al. propose an outer approximation algorithm based on the ideas of Duran, Grossmann and Leyffer, to solve mixed integer, conic optimization programs, and their ideas can thus be specialized to mixed integer semidefinite optimization. The strength of the outer approximation algorithm depends critically on the quality of a linear relaxation of the semidefinite feasible region of the integer SDP, and hence advances in the study of linear approximations of SDPs directly translates to improvements in the outer approximation algorithm. The key insight of our work is that the power of approximation of LPs is weak especially when the feasible region of the linear program does not depend on the objective function or the constraint matrices of the semidefinite program. To the best of our knowledge, the quality of relaxations explicitly depending on the objective function of the SDP has been seldom studied. An explicit form of this question is whether there exists a linear program that approximates the value of the maximum cut of any graph within a factor strictly better than 2 if we allow the LP to depend on the graph at hand. Naturally, this question is ill-posed as stated, and some considerations must be taken to limit the class of linear programs considered. Chapter 1 is dedicated to these considerations and explores conditions under which hardness of approximations results can be avoided. Chapter 2 explores the behaviour of outer approximation algorithms for integer semidefinite programs when the ideas of Chapter 1 are taken into account.

The final chapter of this thesis is dedicated to a combinatorial optimization problem which we call the multi-commodity vector clock problem and concerns the minimization of latencies in a graph where information is perpetually being created and shared. Such problems find applications in network communications and databases. Our study mainly involves finding structural properties of optimal solutions, in the hopes that such structure can be exploited to derive polynomial time approximation algorithms for the multi-commodity vector clock problem.

Chapter 1

INSTANCE-SPECIFIC LINEAR RELAXATIONS OF SEMIDEFINITE OPTIMIZATION PROBLEMS

1.1 Introduction

The generic formulation for a semidefinite optimization problem (SDP) is

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \ & \langle C, X \rangle \\ \text{s.t: } & \langle A_i, X \rangle = b_i, \ \forall i \in [r], \\ & X \succeq 0 \end{aligned} \tag{SDP}$$

where $C \in \mathbb{S}^n$ is a symmetric (without loss of generality) cost matrix and $r \in \mathbb{N}$.

Semidefinite optimization, i.e., the optimization of a linear function over the set of positive semidefinite matrices intersected with an affine subspace [151]. It arises naturally in combinatorial optimization [6, 64, 101, 121], control theory [3, 74, 129], polynomial optimization [94, 128, 129], machine learning [41, 93] and others. These optimization problems are solvable in polynomial time up to an arbitrary accuracy via the theory of interior-point methods [120], which in addition are one of the most successfully approaches used in practice to solve SDPs. It is well known that SDPs are challenging to solve in practice. Typical off-the-shelf solvers use interior-point methods, which require computation of large Hessian matrices (and their inverses) and are often intractable due to memory limitations. For an illustration, see [16, chapter 6.7], [106], and [18] where it is mentioned that state-of-the-art solver such as MOSEK [8] cannot solve semidefinite problems with a symmetric matrix X on more than 250 rows. Inspired by these practical limitations, researchers have proposed several ideas to solve large-scale semidefinite programs. Such techniques are, amongst others, *i*) exploiting structure of the problem (such as sparsity and symmetry), *ii*) producing low rank solutions, *iii*) algorithms based on augmented Lagrangians and the alternating direction method of multipliers. See [105] for a survey of all of these methods.

A relevant family of algorithms to solve semidefinite programs consists in constructing inner and outer polyhedral approximations of the semidefinite cone in order to find a sequence of improving feasible solutions together with tighter bounds on the objective of the SDP, allowing one to trade off between scalability and conservatism.

Research on this class of algorithms is relevant due to its intimate connection with a fundamental question in convex geometry: can the positive semidefinite cone be approximated by polyhedra? Taking the perspective of the field of optimization, this question can be framed by asking if linear programs are strong enough to approximate semidefinite ones. These twin questions, relevant in the fields of optimization and convex geometry respectively, have given rise to a thriving body of research [4, 18, 33, 54].

Outer approximations have been the focus of substantial effort since the hardness of SDP comes from the semidefinite constraint and so one may drop it and and add linear constraints on X implied by $X \succeq 0$. In this case, (SDP) is relaxed to a linear program. A typical example is to add the constraints $X_{i,i} \geq 0,\ \forall i \in [n]$ and $X_{ii} + X_{jj} \pm 2X_{i,j} \geq 0,\ \forall i \in [n],\ \forall j \in [n]$ which are valid for any $X \succeq 0$. These relaxations tend to be weak and seldom used in practice [26, 32]. A well studied example of this phenomenon is the maximum cut problem and the theoretical hardness of approximating it with linear programs which we will discuss in depth in Section 1.3.

The previous approach can be improved using ideas of Kelley [85]. The strategy is to sequentially refine the linear relaxations by aggregation of cutting planes. More concretely, consider the linear relaxation of SDP given by

$$
\begin{aligned}
&\min_{X \in \mathbb{S}^n} \langle C, X \rangle \\
\text{s.t: } &\langle A_i, X \rangle = b_i,\ \forall i \in [r], \\
&v^\top X v \geq 0\ \forall\ v \in \mathcal{S}
\end{aligned}
\tag{$L_{\mathcal{S}}$}
$$

where $\mathcal{S}$ is a finite subset of $\mathbb{R}^n$. Here, we simply insist that $v^\top X v \geq 0$ only for the elements v of the set $\mathcal{S}$. If a solution to this program is not positive semidefinite, we may update $\mathcal{S}$ iteratively. This results in the following algorithm:

The specific implementations of this algorithm mainly differ in how one updates the set $\mathcal{S}$. In [4, 5],the authors use the extreme rays of the set of diagonally dominant matrices, which are then rotated by matrices obtained from a Cholesky decomposition of an optimal solution to the dual of ($L_{\mathcal{S}}$). They also propose an inner approximation of the positive semidefinite cone based on the so-called $DSOS_n$ and $SDSOS_{n,d}$ cones. In a different line of work [11, 44, 135, 144, 157] chose the elements v of $\mathcal{S}$ favoring

Algorithm 1

Fix a finite set $\mathcal{S} \subseteq \mathbb{R}^n$. Drop the semidefinite constraint $X \succeq 0$ of program SDP and solve the resulting linear program $L_{\mathcal{S}}$ finding a minimizer X^*.
while X^* has a negative eigenvalue **do**
  Find a eigenvector v corresponding to the most negative eigenvalue of X^*. Add v to $\mathcal{S}$.
  Solve the updated linear program to find a new minimizer X^*.
end while
return X^*.

sparsity, with the idea that the resulting linear programs will be easier to solve. Bundle methods, such as the *spectral bundle method* of Helmberg and Rendl [72] work with the dual of (SDP), under the further restriction that X has a constant trace. [90] presents a unifying framework for the latter and similar methods. In [18], the constraint $X \succeq 0$ is replaced for infinitely many constraints of the form $f(X, Y) \leq 0$ which must hold for every Y in some convex set $\mathcal{Y}$ and where f is a Lipschitz continuous function. The authors further argue that one should instead solve a second-order cone relaxation, adding the constraints

$$\left\| \begin{pmatrix} 2X_{i,j} \\ X_{i,i} - X_{j,j} \end{pmatrix} \right\|_2 \leq X_{i,i} + X_{j,j}, \ \forall i \in [n], \ \forall j \in [n],$$

which are valid for (SDP).

It is noteworthy that mostly all of these works discuss how to update $\mathcal{S}$, but seldom consider how to initialize it. Typically $\mathcal{S}$ is set to the standard basis of $\mathbb{R}^n$, resulting in the linear constraints $X_{ii} \geq 0, \ i \in \{1, \ldots, n\}$, which are implied by the constraint $X \succeq 0$. Interestingly, under mild conditions, there exists a finite set $\mathcal{S}$ that ensures that the optimal values of the SDP and the linear relaxation $L_{\mathcal{S}}$ match, supporting the approach of using Algorithm1.

Observation 1.1. *Suppose that both (SDP) and its dual, given by the following semidefinite optimization program*

$$\begin{aligned} &\max_{y \in \mathbb{R}^r} \ b^\top y \\ &\text{s.t: } C - \sum_{i=1}^{r} y_i A_i \succeq \ 0 \end{aligned} \tag{DSDP}$$

are strictly feasible. Let y^* be an optimal solution to (DSDP). Let $v_1, \ldots, v_n$ be an orthonormal basis of $\mathbb{R}^n$ of eigenvectors of $C - \sum_{i=1}^m y_i^* A_i = S^*$ with $S^* = \sum_{i=1}^n \beta_i v v^\top$, and β_i the eigenvalues of S^*. Let $\mathcal{S}^* = \{v_1, \ldots, v_n\}$. Then, $L_{\mathcal{S}^*}$ is solvable, and its optimal value matches the optimal value of (SDP).

The proof of this observation is deferred to the section 1.2. Similar versions of Observation 1.1 can be found in [90] and [146].

In fact, [146] proves that if $\mathcal{S} = \{u_1, \ldots, u_l\}$ are the vectors generated by the spectral bundle method of [72] of Rendl et al. to solve DSDP, the objective value of $(L_{\mathcal{S}})$ matches that of (SDP), but this is hardly surprising: if we knew in advance the set of vectors $\mathcal{S}^*$ given by Observation 1.1, we could set $\mathcal{S} = \mathcal{S}^*$ and solve (SDP) as a linear program. More importantly, we emphasize that finding the sets $\mathcal{S}^*$ and $\{u_1, \ldots, u_l\}$ requires solving another comparable SDP, namely DSDP.

In this chapter, we tackle the task of finding a better set $\mathcal{S}$ to initialize Algorithm 1 under certain computational restrictions by drawing inspiration from the question of when - if ever- one can avoid the iterative procedure suggested by Kelley and exactly solve the semidefinite program with a linear program. By “exactly solving" we mean finding a linear relaxation of the SDP whose optimal value equals that of the SDP.

Technically, Observation 1.1 indicates that the question of *exactly* solving an SDP with a linear problem is ill-posed if one does not restrict the set of *algorithms* one is allowed to use to process the instance. We can consider at least three possible approaches to amend this issue. First, restricting the access one has to the given instance. For example, say we are not shown a full SDP instance, but one is allowed to sample a small subset of the entries of the objective and constraints matrices. Second, to only have access to algorithms with at most a certain computational complexity, say matrix multiplication complexity. However, this would require fixing a concrete computational model and proving lower bounds for the complexity of the algorithms to be used, which are typically very hard to obtain. A third approach, which we take in this chapter, is to fix an oracle $\mathcal{O}$, that we can query at most a constant number of times. Concretely, we will assume that we have at our disposal an oracle that can compute a eigenvector decomposition of a symmetric matrix, and that can solve linear programs of polynomial size. If the SDP can in fact be solved with such an oracle, we say it is *solvable under* $\mathcal{O}$.

Hardness of approximation of the max cut problem

The question of finding a good set $\mathcal{S}$ to initialize Algorithm 1 amounts to finding a linear approximations to a semidefinite programs together with a guarantee that the approximation is good. This line of research is motivated by the question of whether the maximum cut (max cut henceforth) problem can be approximated using a linear program by a factor strictly better than 2. This problem consists in finding a bipartition of the nodes of a given graph that maximizes the number of edges with one end in both parts. The results of Poljak, Rendl, Goemans and Williamson [64, 132] show that max cut can be approximated to within a factor of 1.13 by an SDP relaxation. Therefore, a linear approximation of factor at most 1.769 to that SDP would result in a linear approximation the the max cut problem with an approximation better than 2 [1]. Such a result would be striking as the common belief is that max cut cannot be approximated within a factor better than 2 with a linear program in the restricted case that the feasible region of the program is independent of the graph and solely depends on the number of vertices [26, 32, 33, 89, 152]. In Section 1.3, we explore in detail the hardness of approximation results for max cut.

Drawing inspiration from the study of exact solvability of an SDP with an LP, we make the case that we can obtain “good starting" linear approximations for semidefinite programs if one is allowed to let $\mathcal{S}$ depend on the dual of the semidefinite program. The heart of the argument is that the obstructions mentioned for max cut emerge specifically when the polytopes being optimized are determined solely by the number of variables (node pairs for max cut) in a given instance. Hence, we propose to let $\mathcal{S}$ depend on the matrices C and $A_1, \ldots, A_r$ which determine the objective and the constraints of (SDP), and consequently on the feasible region of DSDP. Crucially, such formulations trivially avoid the results in [26] and [89]. We call linear approximations with such dependence “instance-specific". Notice that making some assumption on the algorithms that we can use to interact with the instance is essential here. To illustrate this point, imagine we wish to write a linear program to find the max cut value $mc(G)$ of a graph G. To do so, we can compute a max cut of the graph using brute force and then write an LP with a linear constraint insisting that the objective equals $mc(G)$.

Exact linear relaxations under O

To find candidate sets $\mathcal{S}$ that guarantee that the linear program $L_{\mathcal{S}}$ is a strong relaxation of SDP we first explore sufficient conditions under which the SDP is solvable under the oracle O. Although Observation 1.1 suggests an answer, such a set of vectors

[1] Since $1.77 \cdot 1.13 = 2$.

cannot, as far as we are aware, be obtained with the oracles we are considering. In Section 2, we present Theorems 1.1 and 1.2 which will provide solvability under $\mathcal{O}$ without requiring the solution of a semidefinite program. Our results are tied to the geometry of the dual feasible region of SDP, and a relevant case is when the dual feasible region is a polyhedron. If such is the case and an explicit description of it is available, then program DSDP can be solved as a linear program. Theorem 1.1 shows that under the same condition the *primal* SDP can be solved with a linear program as well. Unfortunately, this theorem is not very useful as it requires enumerating the vertices of the feasible region, which may grow exponentially. The polyhedral assumption has received attention from the literature in the context of *quadratically constrained quadratic problems* (QCQPs) [155], and perhaps more so a weakening of it: simultaneous diagonalizability.

Definition 1.1. *A set of matrices $\{A_i\}_{i\in I} \subseteq \mathbb{R}^{n\times n}$ where I is some set of indices which may be infinite, is said to be simultaneously diagonalizable (SD) if there exists an invertible, orthogonal matrix $U \in \mathbb{R}^n$ such that every element of the set $\{U^\top A_i U\}_{i\in I}$ is a diagonal matrix. Note that $U^\top U = UU^\top = I_n$ as U is orthogonal.*

It turns out that if the set of matrices defining the dual feasible region Γ of SDP is simultaneously diagonalizable, then Γ is a polyhedron [155].

Observation 1.2. *Let Γ be a spectrahedron given by the representation $\Gamma = \{y \in \mathbb{R}^n : C - \sum_i^r A_i y_i) \succeq 0\}$. If the set of matrices $\{C, \{A_i\}_{i\in[r]}\}$ is simultaneously diagonalizable, then Γ is polyhedral.*

We prove this fact in Section 1.2, and point out that the given condition is sufficient but not necessary. Under this more stringent condition, we prove in Theorem 1.2 that $\mathcal{O}$ can be used to solve SDP.

It will typically not be the case that the dual feasible set Γ is polyhedral, and much less that the matrices $C, \{A_i\}_{i\in[r]}$ are simultaneously diagonalizable. In Section 1.2 we prove that this condition is equivalent to the simultaneous diagonalizability of matrices $C - \sum_i A_i p_i$ and $C - \sum_i A_i q_i$ for all p and q in $\mathbb{R}^r$. This characterization suggests that we only insist of the commutativity of the matrices $C - \sum_i A_i p_i$ and $C - \sum_i A_i q_i$ for some p and q. It turns out that this is the key idea to initialize the set $\mathcal{S}$ in Algorithm 1. In Section 1.2 we set the theoretical background of these considerations, and in the following sections we explore their applications to three families of semidefinite optimization problems: the max cut problem, The Lovász theta number and the more generic Shor SDP relaxation of quadratically constrained quadratic problems.

Overview and outline

In this chapter, we explore the question of when an SDP can be solved with a linear program with the intention of improving existing cutting plane approaches to solve SDPs (such as the conservative methods described in [105]). We point out that this family of methods is *not* the de-facto choice to solve large scale semidefinite programs, and very strong methods exist which can scale substantially such as [124, 156, 158, 160, 163]. Nevertheless, we hope these methods might come with their own limitations and in settings where SDPs appear naturally, such as in the sum-of-squares hierarchy for polynomial optimization [159], or whenever optimal solutions to the SDPs are not low rank. In these regimes, polyhedral approximations might be a good alternative. In addition, developing stronger polyhedral approximations to SDPs has consequences in approaches to integer semidefinite programs, which has received attention recently [38, 61, 62, 76, 161] and in spatial branch-and-bound algorithms for non-convex quadratic problems. We explore this later avenue in Chapter 2. The rest of this chapter is organized as follows.

(a) In Section 1.2 we derive two sufficient conditions for solvability of an SDP under $\mathcal{O}$. These conditions are then weakened to produce a strategy to provide candidate starting sets for outer polyhedral approximation algorithms to solve SDPs.

(b) In Section 1.3, we study the setting of finding a maximum cut of a graph G using the semidefinite relaxation of Poljak, Rendel, Goemans and Williamson [64, 132]. Even though the conditions for exact solvability are not met, we use the relaxed version to provide a linear program that certifies a spectral bound in contrast to previous linear relaxations for the maximum cut problem. We then derive a solvability result under $\mathcal{O}$, recovering and generalizing a theorem of Alon and Sudakov [7].

(c) In Section 1.4 we introduce linear relaxations of the Lovász theta number SDP and Shor's semidefinite relaxation for quadratically constrained quadratic programs. We recall as well our linear relaxation of max cut, and introduce a linear strengthening of the max cut SDP.

(d) In Section 1.5 we extensively test our methods empirically on random instances of the problems introduced in Section 1.4. We discuss solving times of the proposed programs.

(e) In Section 1.6 we show the performance of our linear program in the case where the original SDP is itself a relaxation of an underlying optimization problem. We study the case of the max cut problem and the *sparse PCA* problem, where both the SDPs and our linear relaxations can be used to recover a solution to the underlying problem. We show that the quality of our linear programs is competitive with that of the SDPs. For max cut, we compare with results obtained by Mirka and Williamson in [114].

(f) In Section 1.7 we conclude with some remarks.

1.2 Instance-specific linear relaxations of semidefinite optimization problems

In this section we explore the question of exact solvability of semidefinite programs given access to an oracle $\mathcal{O}$, with the following properties:

- Given a set of simultaneously diagonalizable matrices $\{A_1, \ldots, A_r\}$, $\mathcal{O}$ can be called once to compute an orthogonal matrix U such that $U^\top A_i U$ are diagonal matrices for $i = 1, \ldots r$. For an implementation of such an oracle see [65].
- $\mathcal{O}$ can be called a constant number of times to find an optimal solution to a linear program of polynomial size in the bit representation of the information of the SDP, namely the objective and constraint matrices.

In case we can find the optimal value of program SDP by querying $\mathcal{O}$ at most a constant number of times, we say that the SDP is solvable under $\mathcal{O}$, and our intention is to derive sufficient conditions that guarantee solvability of the SDP. It is to be expected that such conditions are not applicable except in some rare cases. We posit that we can derive weakenings of them to provide a starting set $\mathcal{S}$ for Algorithm 1. Recall that a generic SDP is given by

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \ & \langle C, X \rangle \\ \text{s.t: } & \langle A_i, X \rangle = b_i, \ \forall i \in [r], \\ & X \succeq 0. \end{aligned} \tag{SDP}$$

The dual of this program is:

$$\max_{y\in\mathbb{R}^n} \; b^\top y$$
$$\text{s.t: } C - \sum_{i=1}^{m} y_i A_i \succeq 0. \tag{DSDP}$$

Throughout this chapter, we will assume "generic SDPs" and their duals are strictly feasible, and therefore strong duality holds. A spectrahedron Γ is the intersection of the cone of positive semidefinite matrices and an affine subspace. If we identify the affine subspace with $\mathbb{R}^r$ then we can write Γ as:

$$\Gamma = \{y \in \mathbb{R}^r : y_1 A_1 + \cdots + y_r A_r + A_{r+1} \succeq 0\}$$

where $A_1, \ldots, A_r,\ A_{r+1}$ are symmetric $n\times n$ matrices. In general, the map $\mathcal{A} : \mathbb{R}^r \longrightarrow \mathbb{S}^n$ given by $\mathcal{A}(y) = y_1 A_1 + \cdots + y_r A_r + A_{r+1}$ is called an affine symmetric matrix map. Through duality, one can see that spectrahedrons are to semidefinite programs what polyhedra are to linear programs [153].

As we have already pointed out, the necessity of restricting to an oracle with these properties comes from Observation 1.1, as otherwise we could always find an appropriate polyhedron to optimize over. We now prove the content of the observation.

Proof. We first describe the dual of program $L_{\mathcal{S}}$ for a generic set $\mathcal{S} = \{s_1, \ldots, s_k\}$. This program is given by:

$$\max_{y\in\mathbb{R}^n,\ \alpha\in\mathbb{R}^n_+} \; b^\top y$$
$$\text{s.t: } C - \sum_{i=1}^{r} y_i A_i = \sum_{i=1}^{k} \alpha_i s_i s_i^\top. \tag{$DL_{\mathcal{S}}$}$$

Notice that for any set $\mathcal{S} = \{s_1, \ldots, s_k\}$, program $(DL_{\mathcal{S}})$ is a restriction of $(DSDP)$ as the matrices $C - \sum_i y_i A_i$ are restricted to belong to the convex cone generated by the PSD matrices $s_i s_i^T, i \in [k]$, rather than the whole set of positive semidefinite matrices. It follows that the optimal value of (DSDP) upper bounds the optimal value of $(DL_{\mathcal{S}})$ for any set $\mathcal{S}$. By hypothesis, both (SDP) and its dual are strictly feasible and therefore solvable by strong conic duality. Hence, we let $\mathcal{S}^*$ be the elements of a basis of $\mathbb{R}^n$ of orthonormal eigenvectors of an optimal solution S^* of program DSDP. The dual of $L_{\mathcal{S}^*}$ is then $\max_{y\in\mathbb{R}^n\ \alpha\in\mathbb{R}^n_+} b^\top y$ subject to $C - \sum_{i=1}^m y_i A_i = \sum_{i=1}^k \alpha_i v_i v_i^\top$. Hence, letting $y_i = y_i^*$ and $\alpha_i = \beta_i$ gives a feasible solution to $DL_{\mathcal{S}^*}$ which matches the optimal value of $DSDP$ and hence is optimal. To conclude, observe that strong linear duality holds and therefore $L_{\mathcal{S}^*}$ is solvable and its optimal value equals that of $DSDP$.

□

It is clear that whenever Γ is polytope and we have an explicit representation of it given by a system of linear equations $Ax \leq d$, then program DSDP reduces to a linear program. More interestingly perhaps is that the primal problem SDP can also be solved as a linear program, albeit on potentially an exponential number of constraints.

Theorem 1.1. *Consider a generic semidefinite optimization problem SDP, with dual given by DSDP. Suppose that the set*

$$\Gamma = C - \sum_{i=1}^{r} y_i A_i \succeq 0$$

is a polytope with extreme points $p_1 \ldots, p_k$, and define $\mathcal{S} := \bigcup_{i=1}^{k} \mathcal{E}(C - \mathcal{A}(p_k))$. Then, $L_{\mathcal{S}}$ is a linear program and solves SDP.

Proof. The maximum value of the function $b^\top y$ over Γ is achieved at some vertex p of Γ. By strong duality and the solvability of DSDP, there exists some $X^* \succeq 0$ which solves program SDP. In particular, program $L_{\mathcal{S}}$ with $\mathcal{S} := \bigcup_{i=1}^{k} \mathcal{E}(C - \mathcal{A}(p_k))$ where $p_1, \ldots, p_k$ are the vertices of Γ is feasible. Let $\hat{X}$ be an optimal solution to this program. Let $\{v_1 \ldots v_n\} \subseteq \mathcal{S}$ be an orthonormal eigenbasis for the matrix $C - \mathcal{A}(p)$. Since this matrix is positive semidefinite, we can write $C - \mathcal{A}(p) = \sum_{i=1}^{n} \beta_i v_i v_i^\top$ where the $\beta_i, i \in [n]$ are the (non-negative) eigenvalues of $C - \mathcal{A}(p)$. By feasibility of $\hat{X}$, $v_i^\top \hat{X} v_i \geq 0$ for all $i \in [n]$. Multiplying each term by $\beta_i \geq 0$ we derive

$$\sum_{i=1}^{n} \beta_i \left\langle \hat{X}, v_i v_i^T \right\rangle = \left\langle \hat{X}, \sum_{i=1}^{n} \beta_i v_i v_i^T \right\rangle = \left\langle \hat{X}, C - \mathcal{A}(p) \right\rangle \geq 0.$$

This implies that $\left\langle \hat{X}, C \right\rangle \geq \left\langle \hat{X}, \mathcal{A}(p) \right\rangle$.

To conclude, recall that for $j \in [r]$, $\left\langle X, A_j \right\rangle = b_j$ giving the inequality $\left\langle \hat{X}, C \right\rangle \geq b^\top p$. Again by strong duality and since the LP is a relaxation of the SDP, we have $b^\top p = \langle C, X^* \rangle \geq \left\langle C, \hat{X} \right\rangle$ yielding the desired equality $b^\top p = \langle C, X^* \rangle$.

□

In [136] is its shown that deciding if a spectrahedron is a polyhedron is in co-NP, and an algorithm for deciding polyhedrality is given. [20] generalizes and improves the previous results. The algorithm presented in the latter paper runs in exponential time, as it requires enumerating the vertices of a certain polyhedron. Even if we knew that Γ is polyhedral, we do not have exact solvability under O, as the previous problem has an exponential number of constraints. A particular case in which Γ is polyhedral and that has received attention in the literature is whenever the matrices C and $A_i,\ i \in [r]$ are simultaneously diagonalizable. This is the content of observation 1.2, which we now prove.

Proof of Observation 1.2. Also see [155], Lemma 9 . Let U be a matrix that simultaneously diagonalizes matrices C and $A_i,\ i \in [r]$ i.e. the matrices $C' = U^\top C U$ and $A'_i = U^\top A_i U$ are all diagonal. By Silvester's law of inertia [78], we have that $C - \mathcal{A}(y) \succeq 0$ if and only if $U^\top [C - \mathcal{A}(y)] U \succeq 0$ if and only if $C' - \sum_{i=1}^r y_i A'_i \succeq 0$. Hence, we have

$$\Gamma = \{y \in \mathbb{R}^r : C' - \sum_{i=1}^r y_i A'_i \succeq 0\}$$

which is a polyhedral set since all matrices involved are diagonal. □

For a clear exposition of the implications of this observation to QCQPs see [155] and the references therein. In addition, the authors show that the region Γ might be polyhedral even if the matrices C and $\{A_i\}_{\ i\in[r]}$ are not simultaneously diagonalizable. Although the latter condition is much more stringent, it allows us to avoid the need to have the vertices of Γ given to us explicitly, as Theorem 1.1 requires.

Theorem 1.2. *Let SDP be a semidefinite program with dual DSDP. Suppose that the set of matrices $\{C, A_1, \dots, A_r\}$ is simultaneously diagonalizable. Then, SDP is solvable under O.*

Proof. Let U be a orthogonal matrix that simultaneously diagonalizes C and A_i for each $i \in [r]$. Let $v_1, \dots .v_n$ denote the columns of U and set $\mathcal{S} = \{v_1, \dots, v_n\}$. Let p^* be a dual optimal solution with $C - \mathcal{A}(p^*) = S^*$ where S^* is positive semidefinite. Since U diagonalizes each $A_i,\ i \in [r]$ and C, it is clear that the matrix $U^\top [C - \mathcal{A}(p)] U$ is diagonal. In other words, the matrix $U^\top S^* U = D$ for some diagonal matrix D with non-negative entries. This means that we can express S^* as

$$S^* = \sum_{i=1}^{n} \beta_i^* v_i v_i^\top, \beta_i^* \in \mathbb{R}_+ \; \forall\, i \in [n].$$

We turn our attention the linear relaxation of SDP defined by $\mathcal{S}$, defined in Section 1.1, which we recall is given by

$$\begin{aligned} &\min_{X \in \mathbb{S}^n} \langle C, X \rangle \\ \text{s.t: } &\langle A_i, X \rangle = b_i,\ \forall i \in [r], \\ &v^\top X v \geq 0 \; \forall\, v \in \mathcal{S}. \end{aligned} \tag{$L_{\mathcal{S}}$}$$

This program is linear and is a relaxation of SDP as any feasible solution to it is feasible for $L_{\mathcal{S}}$. Its dual is given by

$$\begin{aligned} &\max_{y, \in \mathbb{R}^n, \beta \in \mathbb{R}^n_+} b^\top y \\ \text{s.t: } &C - \sum_{i=1}^{r} y_i A_i = \sum_{i=1}^{n} \beta_i v_i v_i^\top. \end{aligned} \tag{$DL_{\mathcal{S}}$}$$

Observe that this program is a strenghening of program DSDP, and that S^* is feasible for this program. Therefore, their optimal values must match, and in particular the optimal value of $DL_{\mathcal{S}}$ is finite. By strong duality of linear programs, $L_{\mathcal{S}}$ is solvable and its optimal value equals the optimal value of both DSDP and $DL_{\mathcal{S}}$. Again by our strong duality assumption of programs SDP and DSDP, program $L_{\mathcal{S}}$ solves SDP.

□

A class of problems that has been extensively studied in the literature and where the hypothesis of our previous theorem applies are simultaneously-diagonalizable QCQPs. Recall that a QCQP is a problem of the form

$$\inf_{x \in \mathbb{R}^n} q_0(x) : q_i(x) \leq \; 0 \; \forall\, i \in [r]. \tag{QCQP}$$

where $q_0(x) = x^\top C x + d_0^\top x + b_0$ and where $q_i(x) = x^\top A_i x + 2 d_i^\top x + b_i$ with $C, A_i \in \mathbb{S}^n$, $d_0, d_i \in \mathbb{R}^n$ and $b_0, b_i \in \mathbb{R}$ for all $i \in \{1, \ldots, r\}$. QCQPs are NP-hard to solve in general but admit tractable convex relaxations. The SDP relaxation of a QCQP is given by the following semidefinite program [12, 145]:

$$
\begin{aligned}
&\inf_{x\in\mathbb{R}^n, X\in\mathbb{S}^n} \langle C, X\rangle + 2d_0^\top x + b_0 \\
\text{s.t } : &\langle A_i, X\rangle + 2d_i^\top x + b_i \leq \ 0 \ \forall i \in [r] \\
&\begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix} \succeq 0.
\end{aligned}
\tag{1.1}
$$

Whenever the C and the A_i are simultaneously diagonalizable and we have access to a matrix U such that $C = U^\top D_0 U$ and $A_i = U^\top D_i U$ for $i \in \{1, \ldots, r\}$, we can perform the change of variables $y = Ux$ and $\tilde{d}_i = Ud_i,\ i \in \{0, \ldots, r\}$ to obtain the a diagonalized version of the problem

$$
\inf_{y\in\mathbb{R}^n} q_0(y) : q_i(y) \leq \ 0 \ \forall \ i \in [r] \tag{1.2}
$$

However, we have $q_i(y) = a_i^\top y^2 + 2\tilde{d}_i^\top y + d_i$, $a_i \in \mathbb{R}^n$, $\tilde{d}_i \in \mathbb{R}^n$ and $c_i \in \mathbb{R}$ for each $i \in [0, \ldots, r]$. Here, $y^2 \in \mathbb{R}^n$ is the vector whose entries are the squared entries of the vector $y \in \mathbb{R}^n$. Ben-Tal and den Hertog [15] and Locatelli [98] study a certain second order cone relaxation of this problem, and show that the optimal value of that relaxation and that of the SDP relaxation match. Our results imply that in fact, given access to a matrix U that simultaneously diagonalizes the C and $A_i,\ i \in [1, \ldots, r]$ we can solve the SDP relaxation (1.1) with the linear program $L_{\mathcal{S}}$ where $\mathcal{S}$ is the set of columns of U.

Corollary 1.1. *Consider a quadratically constrained quadratic problem given as in QCQP and such that the matrices $C, \{A_i\}_{i\in\{1,\ldots,r\}}$ are simultaneously diagonalizable by an orthogonal matrix U. Let opt be the optimal value of relaxation (1.1) of QCQP. Let $\mathcal{S}$ be the set of columns of U. Then, the objective value z of the linear relaxation $SP_{\mathcal{S}}$ of (1.1) equals opt.*

Proof. The proof is immediate from Theorem 1.2. □

Finding initial sets

As we have seen in Theorem 1.2, we know some vectors whose inclusion in $\mathcal{S}$ guarantees solvability under $\mathcal{O}$. The reason this worked was that we were able to produce a feasible solution to $DL_{\mathcal{S}}$ which matches the objective of an *optimal* solution to DSDP. Nonetheless, the previous argument still holds for a generic feasible solution to DSDP: any dual feasible solution will generate sets $\mathcal{S}$ that satisfy the corresponding dual bound.

Lemma 1.1. *Consider a generic SDP problem and let $\hat{y}$ be a feasible solution to the dual of the SDP with objective value $b^\top \hat{y}$. Let $\mathcal{S} = \mathcal{E}(C - \mathcal{A}(\hat{y}))$. Then, the objective value z^* of program $L_{\mathcal{S}}$ satisfies*

$$z^* \geq b^\top \hat{y}.$$

The proof of this lemma is very similar to that of Theorem 1.2. This result indicates that finding a good set $\mathcal{S}$ amounts to finding feasible solutions to the dual of SDP whose objective value is close to optimal. This task is akin to finding good feasible solutions to SDP, or at worse to solve a semidefinite feasibility problem, which in principle may be as hard as solving the original problem. However, the results of the previous subsection suggest a way to get around this issue by exploiting simultaneous diagonalizability. Under a weakening of this assumption, we will be able to construct solutions, which will be automatically feasible for the the DSDP. To begin, we give in Proposition 1.1 a characterization of simultaneous diagonalizability which we will then relax.

Lemma 1.2. *Let $\{A_i\}_{i\in I} \subseteq \mathbb{S}^n$ be a set of symmetric matrices. Then, there exists a basis of orthonormal vectors $\{u_1, \ldots, u_n\}$ that simultaneously diagonalizes $\{A_i\}_{i\in I}$ if and only if A_i and A_j commute for every i and $j \in I$, i.e, $A_iA_j = A_jA_i$ $\forall i, j \in I$.*

See [40] for a proof.

Proposition 1.1. *The set of matrices $\{A_1, \ldots A_r\} \subseteq \mathbb{S}^n$ is simultaneously diagonalizable if and only if for every p and $q \in \mathbb{R}^r$ the matrices $\mathcal{A}(p) = \sum_{i=1}^r p_iA_i$ and $\mathcal{A}(q) = \sum_{i=1}^r q_iA_i$ commute, and hence are simultaneously diagonalizable.*

Proof. Necessity is trivial by having p and q range over the standard basis of $\mathbb{R}^r$ and Lemma 1.2. For sufficiency, Let U be an orthonormal matrix such that the matrices $U^\top A_iU = D_i$ are diagonal $\forall i \in [r]$. Given p and $q \in \mathbb{R}^r$ we have:

$$U^\top \mathcal{A}(p)U = U^\top \left(\sum_{i=1}^r p_iA_i\right) U = \sum_{i=1}^r p_iD_i.$$

Similarly we have $U^\top \mathcal{A}(q)U = \sum_{i=1}^m q_iD_i$. Since diagonal matrices commute we have

$$\left(\sum_{i=1}^r p_iD_i\right)\left(\sum_{i=1}^r q_iD_i\right) = \left(\sum_{i=1}^r q_iD_i\right)\left(\sum_{i=1}^r p_iD_i\right).$$

Given that $U^\top U = I$, pre-and post-multiplying by U and $U^\top$ respectively gives:

$$U\left(\sum_{i=1}^{r} p_i D_i\right) U^\top U \left(\sum_{i=1}^{r} q_i D_i\right) U^\top = U\left(\sum_{i=1}^{r} q_i D_i\right) U^\top U \left(\sum_{i=1}^{r} p_i D_i\right) U^\top$$

and finally

$$\mathcal{A}(p)\mathcal{A}(q) = \mathcal{A}(q)\mathcal{A}(p).$$

Since these matrices commute, they are simultaneously diagonalizable.

□

Given that commutativity of the set $\{C, A_1, \ldots, A_r\}$ will typically not hold, we relax the equivalent condition given by the previous lemma to require that commutativity holds only for special class of p's and q's. In particular we will set $p = e_{r+1}$ and q such that for some subset $J \subseteq [r]$ we have $\sum_{j \in J} q_j A_j = I_n$. The idea is that if we have a point $y \in \mathbb{R}^n$, not necessarily dual feasible for which the matrices C and $\mathcal{A}(y)$ commute, then taking $\mathcal{S}$ to be the columns of a matrix that diagonalizes them will yield a linear program with objective value as good as the best dual feasible solution that lies on the set

$$\{A \in \mathbb{S}^n : \exists\, x, t \in \mathbb{R} : A = tI_n + x \sum_{j \in [r] \setminus J} q_j A_j\}.$$

Theorem 1.3. *Consider a generic semidefinite optimization problem of the form SDP, with dual DSDP. Suppose that there exists vectors $q^1, q^2 \in \mathbb{R}^r$ whose support is disjoint such that $\sum_{j=1}^{r} q_j^1 A_j = I_n$ and such that the matrices C and $\sum_{j=1}^{r} q_j^2 A_j$ commute and therefore are simultaneously diagonalizable by some orthogonal matrix U. Let $\mathcal{S}$ to be the set of columns of such an U. Then, the optimal value z of program $L_{\mathcal{S}}$ satisfies the bound*

$$\left(\sum_{j=1}^{r} b_j q_j^2\right) x + \left(\sum_{j=1}^{r} b_j q_j^1\right) t \leq z$$

for any x and t such that the matrix $C - x\left(\sum_{j=1}^{r} q_j A_j\right) + tI_n$ is positive semidefinite.

Proof. Let U be a matrix that simultaneously diagonalizes C and $\mathcal{A}(q^2) = \sum_{j=1}^{r} q_j^2 A_j$. Let z be the optimal value of program $L_{\mathcal{S}}$ where $\mathcal{S}$ is the set of columns $v_1, \ldots, v_n$ of U. Recall that the dual of this program is given by

$$\begin{aligned} \max_{y,\in\mathbb{R}^n,\beta\in\mathbb{R}^n_+} \quad & b^\top y \\ \text{s.t: } C - \sum_{i=1}^{r} y_i A_i &= \sum_{i=1}^{n} \beta_i v_i v_i^\top. \end{aligned} \tag{$DL_{\mathcal{S}}$}$$

Since U diagonalizes C, any column v of U is an eigenvector of C with some corresponding eigenvalue λ, and the same holds for $\mathcal{A}(q)$ with some eigenvalue γ. Hence, v is a eigenvector of $C - x\mathcal{A}(q) + tI_n$ with corresponding eigenvalue $\lambda - x\gamma + t$. Since we are looking for x and t values such that $C - x\mathcal{A}(q) + tI_n$ is psd, this gives rise to the equation $\lambda - x\gamma + t \geq 0$, and we have such one equation for every column of U. This system is always feasible as the t variable is free. Hence, there exists x^*, t^* for which the matrix $C - x^*\mathcal{A}(q) + t^*I_n$ is positive semidefinite. As U diagonalizes C, $\mathcal{A}(q)$ and I_n as $U^\top I_n U = U^\top U = I_n$, $C - x^*\mathcal{A}(q) + t^*I_n$ is diagonalizable by U and thus can be written as $\sum_i \eta_i v_i v_i^\top$ with $\eta_i \geq 0$ for $i \in [n]$. Thus, setting $y_j = x^* q_j^2$ if j belongs to the support of q^2 and $y_j = t^* q_j^1$ if j belongs to the support of q^1 (here recall that q^1 and q^2 have disjoint support) gives a feasible solution to program $DL_{\mathcal{S}}$ by setting $\eta_i = \beta_i$ for $i \in [n]$. The objective value of this solution is

$$\left(\sum_{j=1}^{r} b_j q_j^2\right) x^* + \left(\sum_{j=1}^{r} b_j q_j^1\right) t^*. \tag{1.3}$$

□

We make a few observations about this theorem. First and foremost, we didn't require that the matrix $I_n + \sum_{j=1}^{r} q_j A_j$ is feasible for program DSDP. Second, notice that we have required that we can aggregate some of the A_j to form the identity matrix. Although this seems quite constraining, it is always the case that such a combination exists by our assumption that $DSDP$ is strictly feasible, i.e if there exists $q \in \mathbb{R}^r$ such that $C - \mathcal{A}(q) \succ 0$. In principle, finding such q would require finding finding a point in the interior of the dual feasible region, which might be non-trivial. This suggests that our theorem is easier to apply in regimes where it is more directly “obvious " which combination of the A_j forms the identity. This is the case in the max cut problem, the Lovász theta number, the sparse PCA problem, the extended trust region SDP relaxation and many others. Finally, we observe that even though the bound given in Equation 1.3

is the best bound we can *prove*, there might be other “hidden " dual feasible solutions that certify a better bound for $L_{\mathcal{S}}$.

Observation 1.3 (Hidden basis property)**.** *Let $\hat{y}$ be a dual feasible solution for program DSDP with objective value $b^\top \hat{y}$. Suppose that $y \in \mathbb{R}^r$ is a point such that the matrices $C - \mathcal{A}(\hat{y})$ and $C - \mathcal{A}(y)$ (which is not necessarily PSD) share a basis of orthonormal eigenvectors. Let $\mathcal{S} = \mathcal{E}(C - \mathcal{A}(y))$ then, the optimal value z of program $L_{\mathcal{S}}$ satisfies*

$$b^\top \hat{y} \leq z.$$

The proof of this observation is straightforward, but note that we have required $\mathcal{S}$ to be some eigenbasis of $C - \mathcal{A}(y)$ rather than the set of columns of some orthogonal matrix that simultaneously diagonalizes C and $\mathcal{A}(y)$. Clearly, if U diagonalizes both of those matrices it diagonalizes any linear combination of them. As we will see in the max cut experiments, Theorem 1.3 will certify a spectral bound, but the LP relaxation will actually have a better objective than the bound of Theorem 1.3 guarantees in practice.

Finding commuting matrices

To apply Theorem 1.3, we need first to find a combination of the constraints matrices which commutes with the objective matrix C of SDP. This can be accomplished using a linear program. Picking L to be an arbitrary linear function on y gives the program

$$\begin{aligned} \min_{y \in \mathbb{R}^n} \ & L(y) \\ \text{s.t: } & C\mathcal{A}(y) = \mathcal{A}(y)C. \end{aligned} \tag{1.4}$$

To select L, we propose a function that trades off between the ℓ_1 norm of the matrix $C - \mathcal{A}(y)$ and the dual objective function $b^\top y$. The intention of the ℓ_1 term is to promote solutions where $C - \mathcal{A}(y)$ is sparse, rendering the computation of an eigenbasis easier. The term $-b^\top y$ encourages having solutions with good dual objective value. This yields the program

$$\begin{aligned} \min_{y \in \mathbb{R}^n} \ & \sum_{i,j} \left| [C - \mathcal{A}(y)]_{ij} \right| - b^\top y \\ \text{s.t: } & C\mathcal{A}(y) = \mathcal{A}(y)C. \end{aligned} \tag{CG}$$

Note that the null vector is always a feasible solution to this program. In Section 1.5 we experimentally test this idea.

1.3 Linear Relaxations of the max cut semidefinite program

The question of exactly - or approximately - solving an SDP with a linear program finds one of its historical roots in the max cut problem, where in a given undirected graph, we seek a bipartition of the nodes to maximize the number of edges with one end in both parts. Since linear programming has been one of the main paradigms to tackle NP-hard combinatorial optimization problems through the relax-and-round paradigm, substantial efforts were dedicated to find a linear programming relaxation of the max cut problem. A graph with m edges has always a cut of size at least $\frac{1}{2}m$ and any cut can cut at most m edges, so it is trivial to provide an algorithm with integrality gap[2] 2. For example, a randomized algorithm picking vertices at random or a greedy algorithm will have this guarantee. The question was then if there exists a linear program that could have an approximation ratio better than 2.

The starting point of this line of research was perhaps the linear relaxation for max cut given by [13, 133]. Let $G = (V, E)$ be an undirected, simple graph on m edges and W its adjacency matrix. We define

$$\alpha(G) := \max\langle W, X\rangle$$

$$\begin{aligned} X_{ij} + X_{ik} + X_{kj} &\leq 2 \ \forall i, j, k \in V \\ X_{ij} - X_{ik} - X_{jk} &\leq 0 \ \forall i, j, k \in V \\ 0 \leq X_{ij} &\leq 1 \ \forall i, j \in V. \end{aligned} \tag{1.5}$$

Here we use a binary variable X_{ij} for each pair of vertices $\{i, j\}$ to denote if the edge between them is cut. The first set of 'triangle' constraints specify that at most two edges can be picked in a cut from any triangle, while the second set rules out exactly one edge from any triangle from being selected in a cut. In [133], Poljak and Tuza prove that for sparse and dense versions of Erdős-Rényi random graphs, the integrality gaps of this LP tend to $2 - o(1)$ and $\frac{4}{3} - o(1)$ respectively. Here, $G_{n,p}$ denotes the class of random graphs on n nodes where every edge is included independently of others with probability p.

Theorem 1.4. *(Poljak, Tuza) [133] Let $mc(G)$ denote the size of the max cut of G.*

- *(Sparse graphs). Let $p(n)$ be a function such that $0 < p < 1$, $p(n) \cdot n \to \infty$ and $p \cdot n^{1-a} \to 0$ for every $a > 0$, then the expected relative error $\frac{\alpha(G_{n,p}) - mc(G_{n,p})}{mc(G_{n,p})}$ tends to 1 as $n \to \infty$ with probability $1 - o(1)$.*

[2] In this thesis, we employ the convention that the integrality gap is a number that is at least 1 and hence is the ratio of the value of the relaxation to the optimal value of the max cut.

- *(Dense graphs). Let $p(n)$ be a function such that $0 < p < 1$, $p(n) = \Omega\left(\sqrt{\frac{\log(n)}{n}}\right)$. Then the expected relative error $\frac{\alpha(G_{n,p})-mc(G_{n,p})}{mc(G_{n,p})}$, tends to $\frac{1}{3}$ as $n \to \infty$ with probability* $1 - o(1)$.

Such integrality gap lower bounds for the basic LP encouraged two distinct approaches to solve the problem. The first one focused on adding valid constraints to formulation (1.5), such as "hypermetric", and "gap" constraints. See [45, 123] for more details. Nonetheless, a long line of research culminated in showing that such direct strengthenings will fail to provide an approximation factor better than 2 [32, 33, 152]. In particular, Kothari et al. [89] prove that this problem - and more generally Constraint Satisfaction Problems - is resistant to this strategy by showing that extended linear formulations are as powerful as the Sherali-Adams hierarchy, which in turn requires an exponential number of rounds (in ε) to certify an integrality gap better than $2 - \varepsilon$. The second approach, perhaps much more influential, considered stronger optimization relaxations, such as the vector optimization relaxation of Poljak and Rendel [132], shown to be SDP-representable and providing an approximation ratio of ~ 1.13 in the seminal work of Goemans and Williamson [64]. Naturally, this leads to the question if linear programs can well approximate semidefinite ones. In [26] Braun et. al. show that in principle one needs an exponential number of a constraints in an LP to correctly approximate an SDP. These two combined results extinguish the hope that linear programming may be used to approximate max cut. Since the question of finding a good set $\mathcal{S}$ to initialize Algorithm 1 amounts to finding a linear approximation to a semidefinite program, these results suggest that no systematic procedure can generate a good set $\mathcal{S}$ as in particular they would provide an approach to obtain a low-gap linear programming approximation to the max cut problem. In this sense, we propose to use instance-specific information to avoid the hardness of approximation results, in particular by exploiting bounds relating the spectrum of the graph to the value of the max cut, resulting in linear relaxations with better approximation ratios.

For a graph $G = (V, E)$ we set $m = |E|$ and denote by W its adjacency matrix. Recall that the semidefinite relaxation for max cut due to of Poljak, Rendl, Goemans and Williamson [64, 132] is given by

$$\begin{aligned} &\frac{1}{2}m + \frac{1}{4}\max_{X}\ \langle -W, X\rangle \\ \text{s.t: } &X \succeq 0,\quad X_{ii} = 1,\ \forall\, i \in [n]. \end{aligned} \tag{GW}$$

with dual

$$\begin{aligned} &\frac{1}{2}m + \frac{1}{4}\min_{\gamma\in\mathbb{R}^n} \sum_{i=1}^{n} \gamma_i \\ &\text{s.t: } W + diag(\gamma) \succeq 0. \end{aligned} \tag{DGW}$$

It is known that strong duality holds for this pair of programs: both (GW) and (DGW) are solvable and their objectives coincide. Delorme and Poljak show [42] that the max cut value of G on n nodes is upper bounded by the quantity

$$\min_{u\in\mathbb{R}^n:\sum_i u_i=0} \frac{n}{4}\lambda_1(\mathcal{L}(G) + diag(u)).$$

It turns out that this program is equivalent to program DGW [64]. In their seminal work, Goemans and Williamson show that this program achieves an approximation ratio of roughly $\frac{1}{0.878} \sim 1.138$. Through this equivalence, one can show that the semidefinite program GW satisfies a series of eigenvalue bounds. For instance, one may take u such that $\sum_{i=1}^n u_i = 0$ and all of the diagonal entries of the matrix $\mathcal{L}(G) + diag(u)$ equal $\frac{2m}{n}$. This results in what is usually known as *the* eigenvalue bound for max cut due to Mohar and Poljak [117]

$$mc(G) \leq \frac{1}{2}m + \frac{n}{4}\lambda_1(-W) = \frac{1}{2}m - \frac{n}{4}\lambda_n(W) \tag{1.6}$$

To see the second equality, recall that for any matrix A, $\lambda_n = \lambda_1(-A))$. See [7, 117] for an elementary proofs of this inequality. As mentioned in [123], conventional wisdom is that LPs cannot certify even the eigenvalue bound, and we are not aware of a polynomially sized linear program that certifies this bound.

Instance-specific linear relaxations.

The specialization of program $L_{\mathcal{S}}$ to the max cut problem results in a polynomially sized linear program that explicitly depends of the adjacency matrix W on G, allowing us to circumvent the theoretical limitations of linear relaxations described in the introduction of this section. Using Theorem 1.3 this LP will be shown to satisfy the eigenvalue bound (1.6) whenever $\mathcal{S}$ is chosen appropriately. Fixing $\mathcal{S} = \{v_1, \ldots, v_k\}$, program $L_{\mathcal{S}}$ specializes to a linear program which we denote by program $SP_{\mathcal{S}}$.

$$\begin{aligned} &\max_{X\in\mathbb{S}^n} \frac{1}{2}m + \frac{1}{4}\langle -W, X\rangle \\ &\text{s.t: } v^\top X v \geq 0 \; \forall\, v \in \mathcal{S}, X_{ii} = 1, \; \forall\, i \in [n], \|X\|_\infty \leq 1. \end{aligned} \tag{$SP_{\mathcal{S}}$}$$

In this program we have included the constraint $\|X\|_\infty \leq 1$. As the following observation shows, this is a valid constraint for GW. Adding it is useful because it will guarantee that the dual of $SP_{\mathcal{S}}$ is always feasible, regardless of G.

Observation 1.4. *Let X be feasible for program (GW). Then, it is feasible for program $SP_{\mathcal{S}}$ for any set $\mathcal{S} \subseteq \mathbb{R}^n$.*

Proof. Let X be feasible for (GW). This means X is positive semidefinite, and that there exists a set of vectors $x_1, \ldots, x_n$ such that $X_{ij} = x_i^\top x_j$ for all $i, j \in [n]$. For each $i \in [n]$ we have $X_{ii} = 1$ and thus we see that $\|x_i\|_2 = 1$. It follows that each entry of the vectors x_i is bounded by 1 and therefore that X_{ij} is bounded by 1 for all i and j. The other two constraints of the linear program are clearly satisfied by X.

□

It will be also be useful to consider the following strenghening of program GW depending of $\mathcal{S} = \{v_1, \ldots, v_k\}$.

$$\begin{aligned} &\frac{1}{2}m + \frac{1}{4}\max_{\eta \in \mathbb{R}^k} \left\langle -W, \sum_{i=1}^{k} \eta_i v_i v_i^\top \right\rangle \\ \text{s.t: } &diag\left(\sum_{i=1}^{k} \eta_i v_i v_i^T\right) \leq 1, \eta_i \geq 0,\ v_i \in \mathcal{S}\ \forall i \in [k],\ k = |\mathcal{S}|. \end{aligned} \tag{$SD_{\mathcal{S}}$}$$

Here, and for the rest of the chapter, we denote by $z_{SP_{\mathcal{S}}}, z_{GW}, z_{DGW}, z_{SD_{\mathcal{S}}}$ the optimal values of $SP_{\mathcal{S}}$, GW, DGW and $SD_{\mathcal{S}}$ ignoring the additive constant $\frac{1}{2}m$ and the multiplicative constant $\frac{1}{4}$, respectively. For illustration, we have:

$$\begin{aligned} z_{GW} &= \max\ \langle -W, X\rangle \\ \text{s.t: } &X \succeq 0,\ \ X_{ii} = 1,\ \forall\, i \in [n]. \end{aligned}$$

By duality, we get the following relationships between these optimal values

$$z_{SD_{\mathcal{S}}} \leq z_{GW} = z_{DGW} \leq z_{SP_{\mathcal{S}}}.$$

Observe that we may employ different sets $\mathcal{S}$ to define SP and SD and the above relations will continue to hold. As a sanity check, we first observe that program $SP_{\mathcal{S}}$ satisfies the trivial bound for max cut.

Lemma 1.3. *Let $\mathcal{S}$ be an arbitrary subset of $\mathbb{R}^n$. Then $z_{SP_{\mathcal{S}}}$ satisfies:*

$$z_{SP_{\mathcal{S}}} \leq 2m$$

and therefore $\frac{1}{2}m + \frac{1}{4}z_{SP_{\mathcal{S}}} \leq m$.

Proof. Let $\mathcal{S} = \{v_1, \dots, v_k\}$. The dual of program $SP_{\mathcal{S}}$ is given by

$$\begin{aligned} \min_{\lambda,\alpha,\delta,\beta,\Lambda} \ & \frac{1}{2}m - \frac{1}{4}\left[tr(\Lambda) - \sum_{i \neq j}\delta_{ij} - \sum_{i\neq j}\alpha_{ij}\right] \\ \text{s.t: } & W - \Lambda = \sum_{i=1}^{k}\beta_i v_i v_i^\top, \\ & \delta_{ij} \geq 0 \ \forall i \neq j \in [n], \\ & \alpha_{ij} \geq 0 \ \forall i \neq j \in [n], \\ & \lambda_i \in \mathbb{R} \ \forall i \in [n], \\ & \beta_i \geq 0 \ \forall i \in [n], \\ \Lambda \in \mathbb{S}^n, \Lambda_{ij} = & \delta_{ij} - \alpha_{ij} \ \forall i \neq j \in [n], \Lambda_{ii} = \lambda_i \ \forall i \in [n]. \end{aligned} \tag{$DSP_{\mathcal{S}}$}$$

To see this we ignore the constant $\frac{1}{2}m$ in the objective together with the multiplicative term $\frac{1}{4}$. Introduce dual variables $\lambda_i \in \mathbb{R}$ for $i \in [n]$ corresponding to the constraints $X_{ii} = 1$, $\beta_i \in \mathbb{R}^n_+$ for $i \in [k]$ corresponding to $v^\top X v \geq 0$ and $\alpha_{i,j}, \delta_{i,j} \geq 0$ for $i \neq j \in [n]$, corresponding to $X_{ij} \leq 1$ and $X_{ij} \geq -1$ respectively, for $i \neq j \in [n]$ (in fact we need only to consider the indices $i < j$ since X is symmetric but we will ignore this as it only complicates the proof). Multiplying the dual variables with the constraints accordingly gives the inequality

$$\sum_{i=1}^{n}\lambda_i X_{ii} + \sum_{i=1}^{k}\beta_i\langle X, v_i v_i^\top\rangle - \sum_{i\neq j}\alpha_{ij}X_{ij} + \sum_{i\neq j}\delta_{ij}X_{ij} \geq \sum_{i=1}^{n}\lambda_i - \sum_{i\neq j}\delta_{ij} - \sum_{i\neq j}\alpha_{ij}$$

Let $\Lambda_{ij} = \delta_{ij} - \alpha_{ij}$ for $i \neq j$ and $\Lambda_{ii} = \lambda_i$ for all $i \in [n]$. This gives the inequality

$$\langle \Lambda, X\rangle + \sum_{i=1}^{k}\beta_i\langle X, v_i v_i^\top\rangle \geq \sum_{i=1}^{n}\Lambda_{ii} - \sum_{i\neq j}\Lambda_{ij}.$$

If we let $\Lambda + \sum_{i=1}^{k}\beta_i v_i v_i^\top = W$ we get

$$\langle -W, X\rangle \leq \sum_{i\neq j} \Lambda_{ij} - \sum_{i=1}^{n} \Lambda_{ii}$$

Letting $\beta_i = 0\ \forall i \in [n]$, $\Lambda = W$ where $\delta_{ij} = 1, \alpha_{ij} = 0$ whenever $W_{ij} = 1$ and 0 otherwise, we obtain a feasible solution for the previous program with $tr(\Lambda) - \sum_{i\neq j} \delta_{ij} - \sum_{i\neq j} \alpha_{ij} = -2m$.

□

It can be checked that for an arbitrary graph G, the feasible region of program DGW, namely $\Gamma = \{\gamma \in \mathbb{R}^n : W + diag(\gamma) \geq 0\}$ is not necessarily polyhedral. However, we can exploit Theorem 1.3 to derive a set $\mathcal{S}$ for the relaxation $SP_{\mathcal{S}}$ that has a good objective value. Although this statement can be proven directly by simply giving a judicious choice of $\mathcal{S}$, we derive the result in a way that explicitly uses the theorem.

Theorem 1.5. *Let G be a graph on n vertices and W its adjacency matrix. Let λ_n denote the smallest eigenvalue of W. Set $\mathcal{S} = \mathcal{E}(W)$. Then*

$$z_{SP_{\mathcal{S}}} \leq -n\lambda_n := \chi(G).$$

Proof. For $i = 1, \ldots, n$ let the matrix A_i denote the matrix of all zeros but with a single 1 in its i-th diagonal entry. Hence, Γ can be expressed as:

$$\Gamma = \{\gamma \in \mathbb{R}^n : W + \sum_{i}^{n} \gamma_i A_i \geq 0\}.$$

To apply Theorem 1.3, we express the identity as some combination of the A_i. concretely, we let $\hat{\gamma} = \vec{1}$ be the vector of all ones in $\mathbb{R}^n$ so that we have that $\sum_{i=1}^{n} \hat{\gamma}_i A_i = I_n$. By Theorem 1.3, it follows that if $\mathcal{S} = \mathcal{E}(W)$ then the optimal value $z_{SP_{\mathcal{S}}}$ of program $SP_{\mathcal{S}}$ satisfies

$$z_{SP_{\mathcal{S}}} \leq t \cdot n$$

for any t such that $W + tI$ is positive semidefinite. Observe that $W - \lambda_n I$ is positive semidefinite. In particular, we obtain

$$z_{SP_{\mathcal{S}}} \leq -n\lambda_n.$$

□

We next provide an alternate, direct proof of this result by directly setting $\mathcal{S} = \mathcal{E}(W)$ and using the dual of program $SP_{\mathcal{S}}$.

Alternative proof of Theorem 1.5. The inequality holds if we are able to show a feasible solution of the dual program of $(SP_{\mathcal{S}})$ whose objective value equals $-n\lambda_n$. Let $\mathcal{S} = \{v_1, v_2, \ldots, v_n\} = \mathcal{E}(W)$. Consider an eigenvector v of W with corresponding eigenvalue λ, so that $Wv = \lambda v$. Observe that $\lambda - \lambda_n \geq 0$ since λ_n is the most negative eigenvalue of W. Clearly, the vector v is a eigenvector of the matrix $W - \lambda_n I_n$ with corresponding eigenvalue $\lambda - \lambda_n$. By the spectral theorem, we have $W - \lambda_n I_n = \sum_{i=1}^{n} (\lambda_i - \lambda_n) v_i v_i^T$, where $v_1, \ldots, v_n$ are an orthonormal eigenbasis of W. In other words, we have:

$$\lambda_n I_n + \sum_{i=1}^{n} (\lambda_i - \lambda_n) v_i v_i^T = W.$$

This yields the desired feasible solution $\Lambda = -\lambda_n I_n$ which has an objective value $\frac{1}{2}m - \frac{n}{4}\lambda_n$ for $DL_{\mathcal{S}}$. $\square$

Interestingly, this result allows us to show that the linear relaxation $SP_{\mathcal{S}}$ is strictly stronger than the linear formulation for max cut given in program (1.5), in the sense that it gives -in contrast to the previous LP- the correct value of max cut for the graphs considered in Theorem 1.4. Perhaps more interestingly, we show that for random $d-$regular graphs the linear program $SP_{\mathcal{S}}$ with $\mathcal{S} = \mathcal{E}(W)$ approximates max cut with an approximation factor of $1 + O(\frac{1}{\sqrt{d}})$. This result is quite striking as it is precisely for random $d-$ regular graphs (with $d \in O(1)$) that the hardness of approximation for max cut using the Sherali-Adams hierarchy was shown [32, 33, 89, 152]. These two claims are the content of the next two corollaries.

Corollary 1.2. *Let $G = G(n, p)$ be sampled according to the Erdős-Rényi model [59] where p is a function of n. Let $g(n)$ be a non-decreasing function of n. Then, the ratio $\frac{\frac{1}{2}m + \frac{1}{4}\chi(G)}{\frac{1}{2}m}$ is at most $1 + \sqrt{\frac{2}{g(n)}}$ as long as np is at least $\frac{g(n)}{2}\log(n)$, with high probability. In particular, for all dense graphs of Theorem 1.4, $np \geq g(n) \geq \sqrt{n}$ and the quotient converges to 1. For sparse graphs of Theorem 1.4 where $np = c\log(n)$, the quotient is at most $1 + \sqrt{\frac{2}{c}}$.*

Proof. Let $p = p(n)$ and $G = G(n, p)$ be sampled according to the Erdős-Rényi model with $np \in \Omega(\frac{g(n)}{2}\log(n))$. Letting $\varepsilon = \frac{1}{n}$ and applying Theorem 1 of [37] we have that with probability at least $1 - \frac{1}{n}$

$$-\lambda_n \leq \sqrt{4np\ln\left(\frac{2n}{\varepsilon}\right)} + p.$$

Recalling that the number of edges m of G is $\theta(n^2p)$ with high probability, a direct computation of the quantity $\frac{\frac{1}{2}m-\frac{1}{4}n\lambda_n}{\frac{1}{2}m}$ gives the result. □

Corollary 1.3. *Suppose that G is a d-regular graph with $-\lambda_n \leq c \cdot \sqrt{d}$ for some constant c and $\mathcal{S} = \mathcal{E}(W)$. Then, the following inequality holds:*

$$\frac{z_{SP_{\mathcal{S}}}}{z_{SD_{\mathcal{S}}}} \leq 1 + \frac{c}{\sqrt{d}}.$$

Proof. Recall that a d-regular graph has $m = \frac{nd}{2}$ edges. This gives $n = \frac{2m}{d}$. Suppose $-\lambda_n \leq c \cdot \sqrt{d}$. Then, by Theorem 1.5 and that $z_{GW} \geq 0$ for any graph G we get

$$\frac{\frac{1}{2}m + \frac{1}{4}z_{SP_{\mathcal{S}}}}{\frac{1}{2}m + \frac{1}{4}z_{GW}} \leq \frac{\frac{1}{2}m - \frac{1}{4}n\lambda_n}{\frac{1}{2}m} = \frac{\frac{1}{2}m - \frac{1}{4}\lambda_n\frac{2m}{d}}{\frac{1}{2}m} = 1 - \frac{\lambda_n}{d} \leq 1 + \frac{c}{\sqrt{d}}.$$

□

It is known that random $d-$regular graphs satisfy the hypothesis of the theorem [55, 58, 149], justifying our previous claim on the guarantees of our linear relaxation on random $d-$regular graphs. Another class of graphs which satisfies the hypothesis of the theorem are the Ramanujan expander graphs [104], where $c = 2$. We contrast this result with the fact that the relative error of $\alpha(G)$- defined above in LP(1.5)- relative to the max cut of G tends to 1 for Ramanujan graphs [133].

To the best of our knowledge, this is the first linear relaxation of max cut with these two guarantees.

Hidden basis property and stronger guarantees

In the previous subsection, we considered a bound given by Theorem 1.3 using the fact that $-\lambda_n\vec{1}$ is feasible for program DGW. However, it might very well be the case that there are other "hidden" dual feasible solutions. Although we are not aware of any such solutions, it is illustrative to check whether or not program $SP_{\mathcal{S}}$ gives better solutions than the eigenvalue bound. This raises the question of the quality of our linear relaxation in the setup where the eigenvalue bounds fails to give an approximation factor better than 2 for the maximum cut value of a graph. Indeed, the eigenvalue bound is not powerful enough to provide an approximation factor better than $2 - \varepsilon > 0$ for any given $\varepsilon > 0$ in general. As a matter of fact, for any given $\varepsilon > 0$, there exist a family of graphs whose maximum cut is bounded above by $\frac{1}{2}m + \varepsilon m$, but the eigenvalue bound cannot certify a bound better than $2 - \varepsilon$. We give an example of such as class in our next definition, which is inspired by a remark in [123].

Definition 1.2. *We say that a graph G is sampled from the class of random graphs $\mathcal{G}(n, d, l)$ if G has n vertices and two disjoint components G_1 and G_2 where G_1 is a random d-regular graph, G_2 is complete bipartite graph where each side of the bipartition has $\sqrt{n}$ nodes, and l random edges connect the G_1 and G_2.*

Observe that the absolute value of most negative eigenvalue of the adjacency matrix of a graph sampled from $\mathcal{G}(n, d, l)$ is $\Omega(\sqrt{n})$ due to the bipartite component. If $d \in O(1)$ then the number of edges in G is linear in n and so is the maxcut of G. However, the eigenvalue bound is weak: it certifies that the maxcut size is at most $O(n^{1.5})$ (notice that this is worse even than the trivial upper bound of m). This class of graphs is suggested as an example in [123] as a class of graphs where the eigenvalue bound behaves poorly. However, our LP certifies a much better value, when $l = 0$, as the next observation shows:

Lemma 1.4. *Let G be a graph with two disconnected components G_1 and G_2, where $|V(G_1)| = n_1$, $|V(G_2)| = n_2$, λ^1 is the smallest eigenvalue of the adjacency matrix of graph G_1 and λ^2 is the smallest eigenvalue of the adjacency matrix of G_2. Let $\mathcal{S} = \mathcal{E}(W)$. Then, $SP_{\mathcal{S}}$ certifies:*

$$z_{SP_{\mathcal{S}}} \leq n_1\lambda^1 + n_2\lambda^2.$$

Proof. The proof is basically the same as the proof of Theorem 1.5 by observing that the support of eigenvectors corresponding to disjoint components of a graph are disjoint. □

This result may seem artificial in the sense that G is a disconnected graph. However, we show through extensive experiments in Tables 1.1 and 1.2 in Section 1.5 that the quotient of the optimal value $SP_{\mathcal{S}}$ to the GW relaxation is significantly better than the quotient of $\chi(G)$ to the GW relaxation, even when edges are added between the components in these difficult examples for the eigenvalue bound.

Solvability of maxcut under $\mathcal{O}$

In the previous subsection, we have seen that we can derive a good starting set $\mathcal{S}$. In general, program $L_{\mathcal{S}}$ does not solve the max cut SDP. In this subsection we will show that whenever G is a *distance regular graph* then we have solvability of the max cut SDP under $\mathcal{O}$. The class of distance-regular graphs contains strongly regular graphs, which have been extensively studied for their algebraic, combinatorial and spectral properties [27, 147]. Famous graphs such as the Petersen graph belongs to this class. In what follows, we give a sufficient condition that ensures that the value of $SP_{\mathcal{S}}$ equals the optimal value of the GW relaxation, provided that $\mathcal{S}$ includes an orthonormal eigenbasis of W.

Definition 1.3 (Distance-regular graphs)**.** *For a graph G and u, v vertices in $V(G)$ define $G_j(u)$ to be the set of vertices of G at distance exactly j of u, i.e., the vertices $v \in V(G)$ such that the shortest path joining u and v has length j. We say G is distance regular if it is connected, d-regular for some d and there exists integers c_i, b_i, $i \in \mathbb{N}$ such that for any two vertices u, v at distance $i = d(u, v)$ there are precisely c_i neighbours of v in $G_{i+1}(u)$ and b_i neighbours of v in $G_{i-1}(u)$.*

Examples of such graphs are all strongly regular graphs, Hamming graphs, complete graphs, cycles, and odd graphs (such as the Petersen graph) [27]. The next theorem will allow us to prove that our linear relaxations are tight for this class of graphs.

Theorem 1.6. *Let G be a graph and W its adjacency matrix. Let $\mathcal{S} = \mathcal{E}(W)$ and W_n be the eigenspace of W corresponding to λ_n. Suppose the dimension of W_n is k with $n > k \geq 1$. Suppose there exists an orthonormal basis $\mathcal{U} = \{u_1, \ldots, u_k\}$ of W_n such that the matrix A with rows $u_1, \ldots, u_k$ has columns with constant 2- norm, i.e. there exists some $c \in \mathbb{R}^+$ such that $\|A_j\|_2 = c \; \forall j \in [n]$ where A_j denotes the j-th column of A. Then, $z_{SD_{\mathcal{S}}}$ equals $-n\lambda_n$ and in particular*

$$z_{SP_{\mathcal{S}}} = z_{GW} = z_{SD_{\mathcal{S}}}.$$

Proof. The proof requires two steps. First, we show that if such basis $\mathcal{U}$ exists and we let $\mathcal{S} = \mathcal{U}$ then the theorem holds. Second, we show that we may set $\mathcal{S}$ to be an arbitrary orthonormal basis of W_n. This is necessary since the dimension of $W_n \geq 2$ and hence orthonormal bases are not unique. This might break the theorem if we choose any other orthonormal basis for $\mathcal{S}$ instead of $\mathcal{U}$. We begin with the first step. Notice that $c = \sqrt{\frac{k}{n}}$. Indeed, since the u_i are unitary vectors we have that for all $i \in [k]$ $\sum_{j=1}^{n} A_{i,j}^2 = 1$. Summing over i gives $\sum_{i=1}^{k} \sum_{j=1}^{n} A_{i,j}^2 = k$. By our assumption of constant sum of the column vectors, we get $\sum_{i=1}^{k} A_{i,j}^2 = c^2 \ \forall j \in [n]$. Summing over j gives $\sum_{j=1}^{n}\sum_{i=1}^{k} A_{i,j}^2 = nc^2$ and we get $k = nc^2$. Let $B = \sqrt{\frac{n}{k}}A^\top$ and $Y = BB^\top$. Let v_i denote the ith row of B, and recall that v_i has norm $\sqrt{\frac{k}{n}}$. This implies that $Y_{ii} = v_i \cdot v_i = \frac{n}{k}\cdot\frac{k}{n} = 1$. Finally, observe that $Y = \frac{n}{k}\sum_i^k u_i(u_i)^\top$. It follows that Y is feasible for $SD_{\mathcal{S}}$ with $\mathcal{S} = \mathcal{U}$. This solution has an objective value

$$Z_{SD_{\mathcal{S}}} \geq \left\langle -W, B^T B\right\rangle \geq \left\langle -W, \frac{n}{k}\sum_{i=1}^{k} u_i u_i^T\right\rangle = -n\lambda_n.$$

For the second part, we show that we can take $\mathcal{S}$ to be any arbitrary orthonormal basis of W_n. Notice that the only fact that we used from $\mathcal{U}$ is that the matrix A formed by stacking the vectors u_i as rows has constant column norm. Therefore, it suffices to show that any matrix A' formed in the same way from an arbitrary basis $\mathcal{U}'$ has this same property. Hence, let $\mathcal{U}' = \{w_1, \ldots, w_k\}$ be an arbitrary basis of W_n and suppose that the basis $\mathcal{U}$ exists.

Since the vectors$\{u_1, \ldots, u_k\}$ are an orthonormal basis of W_n which is a lineal subspace of $\mathbb{R}^n$, we can extend this set of vectors to a full orthonormal basis $\{u_1, \ldots, u_k, u_{k+1}, \ldots u_n\}$ of $\mathbb{R}^n$. Further, observe that $\sum_{i=1}^{n} u_i(u_i)^\top = I_n$ where I_n is the $n \times n$ identity matrix. To see this, let $v = r_1u_1 + \cdots + r_nu_n \in \mathbb{R}^n$ be an arbitrary vector expressed in the u_i, $i \in [n]$ basis. We have

$$\left(\sum_{i=1}^{n} u_i u_i^\top\right) v = \sum_{i=1}^{n}\langle u_i, v\rangle u_i = \sum_{i=1}^{n} r_i u_i = v. \tag{1.7}$$

We derive that $\sum_{i=1}^{n} u_i u_i^\top$ equals the identity matrix. Notice that this equation remains true if we replace the u_i for any arbitrary orthonormal basis of $\mathbb{R}^n$. Since the diagonal entries of $A^\top A$ equal $\frac{k}{n} = c^2$ we see that the diagonal entries of $\sum_{i=k+1}^{n} u_i u_i^\top$ equal $1 - c^2$. Finally it follows that $\{w_1, \ldots, w_k, u_{k+1}, \ldots, u_n\}$ is as well a basis for $\mathbb{R}^n$ and thus by Equation (1.7) we have $\sum_{i=1}^{k} w_i w_i^\top + \sum_{i=k+1}^{n} u_i u_i^\top = I_n$. This shows that every diagonal entry of the matrix $\sum_{i=1}^{k} w_i w_i^\top$ must equal c, and hence the matrix A' formed by stacking the vectors w_i as rows has constant column norm. The conclusion of the theorem follows from the inequality $Z_{SD_{\mathcal{S}}} \leq Z_{GW} \leq Z_{SP_{\mathcal{S}}} \leq -n\lambda_n$. □

Alon and Sudakov proved something similar to the first part of our proof in [7]. In the paper, the authors prove that $z_{GW} = \frac{1}{2}m - \frac{1}{4}n\lambda_n$ under the hypothesis that there exists a feasible solution $Y = B^\top B$ for the (GW) relaxation such that the columns of B are unitary vectors $v_1, \ldots, v_n$ and its rows $u_1, \ldots u_k$, $1 \leq k \leq n$ are eigenvectors of W

corresponding to λ_n. We conclude this section with the corollary for distance-regular graphs.

Corollary 1.4. *Let G be a distance-regular graph. Let $\mathcal{S} = \mathcal{E}(W)$. Then*

$$z_{SP_{\mathcal{S}}} = z_{GW} = z_{SD_{\mathcal{S}}}.$$

Proof. The results follows from the following theorem. It states that the eigenspaces of distance regular graphs satisfy the hypothesis of Theorem 1.6.

Theorem 1.7 ([27], Theorem 4.1.4)**.** *Let G be a distance regular graph and λ an eigenvalue of G. Then, there exists a symmetric matrix whose columns span the eigenspace corresponding to λ and that have a constant norm.*

□

1.4 Applications to semidefinite programs

To verify the applicability of the ideas presented, we consider three families of semidefinite optimization problems, each illustrating an aspect of our work. The first problem considered is the semidefinite relaxation of the maxcut problem which we presented in Section 1.3. We present our experimental results in Section 1.5.

Maximum Cut

The max cut problem is a prime example of how our methodology can be applied as it is a hard combinatorial problem that linear programs fail to approximate. We will test our ideas using two linear programs, already introduced in Section 1.3.

$$\begin{aligned} &\max_{X \in \mathbb{S}^n} \ \frac{1}{2}m + \frac{1}{4}\langle -W, X\rangle \\ &\text{s.t: } v^\top X v \geq 0 \ \forall \ v \in \mathcal{S}, X_{ii} = 1, \ \forall \ i \in [n], \|X\|_\infty \leq 1. \end{aligned} \tag{$SP_{\mathcal{S}}$}$$

By the results of Section 1.3, we know that as $n \to +\infty$, the optimal value of this program will converge to the optimal value of max cut for Erdős-Rényi graphs and random d-regular graphs whenever $\mathcal{S}$ contains a basis of eigenvector of the matrix W. We test the quality of the linear relaxation on such graphs, as well as on graphs of the family $\mathcal{G}(n, l, k)$ which was introduced in Section 1.3. This family was designed to have a trivial eigenvalue bound. In Section 1.6 we include as well experiments on the quality of relaxations on 16 graphs taken from TSPLIB [138] and 14 graphs from the network repository [141]. Furthermore, we consider program

$$\begin{aligned} &\frac{1}{2}m + \frac{1}{4}\max_{\eta\in\mathbb{R}^k}\left\langle -W, \sum_{i=1}^{k}\eta_i x_i x_i^\top \right\rangle \\ \text{s.t: } & diag\left(\sum_{i=1}^{k}\eta_i x_i x_i^T\right) \leq 1, \eta_i \geq 0,\ x_i \in \mathcal{S}\ \forall i \in [k],\ k = |\mathcal{S}|. \end{aligned} \tag{$SD_{\mathcal{S}}$}$$

This program is useful as we can obtain graph cuts from its solution using the rounding procedure of Goemans and Williamson [64]. Since the focus of this chapter is comparing the optimal value of the different linear relaxations versus the optimal value of the SDPs, we defer results on rounded solutions to Section 1.6.

Lovász theta number

The second problem we consider is the Lovász theta number $\vartheta(G)$ introduced by Lovász in the seminal paper [101] as a convex relaxation for the stability number of a graph G. ϑ can be computed in polynomial time using a semidefinite program. Since $\vartheta(\bar{G})$ -where $\bar{G}$ is the complement of G- is lower and upper bounded resp. by the clique number and the chromatic number of G, it allows one to compute those numbers in polynomial time for graphs for which these two quantities coincide e.g., perfect graphs.

$\vartheta(G)$ can be computed by the following semidefinite optimization program:

$$\begin{aligned} &\max_{S\in\mathbb{R}^n} \langle J, X\rangle \\ \text{s.t: } & tr(X) = 1,\ X_{i,j} = 0\ \forall (i,j) \in E, \\ & X \succeq 0. \end{aligned} \tag{Tn}$$

This problem is related to our setup, as it is known that the feasible region of the dual program is polyhedral whenever the considered graph is perfect. This striking results coincides with the fact that it is precisely for these graphs where the theta number coincides with the *independence number* of the graph.

We apply the ideas developed in Section 1.2 on two families of graphs. The first class is that of *regular* graphs. Notice that the constraints $X_{i,j} = 0\ \forall (i,j) \in E$ can be expressed as $\left\langle X, A^{ij}\right\rangle = 0$ where A^{ij} is matrix of all zeros except it has a 1 in its ij,ji entries whenever G contains edge ij. It is clear that if A is the adjacency matrix of G, we have $A = \sum_{ij,\in E} A^{ij}$. Regular graphs are interesting in our setting as it is easy to check that whenever G is regular graphs, A its adjacency matrix, and J the matrix of all ones, we have $JA = AJ$. The second class of graphs we consider are Erdős-Rényi random graphs, which are typically not regular and it is not obvious how to combine the A^{ij} to obtain a matrix that commutes with J. We will use program (CG) to find such matrices.

Given a finite set $\mathcal{S}$, we obtain the linear relaxation of program (Tn):

$$\begin{aligned} &\max_{X \in \mathbb{S}^n} \langle J, X \rangle \\ \text{s.t: } & tr(X) = 1,\, X_{i,j} = 0 \;\forall (i,j) \in E, \\ & v^\top X v \geq 0 \;\forall\, v \in \mathcal{S} \end{aligned} \tag{LTn}$$

In Section 1.5, we compare the objective value of programs (Tn) and (LTn), on Erdős-Rényi random graphs and $d-$regular graphs. Interestingly, this problem is much more resistant to the the cut generation strategy for solving the corresponding SDP proposed in Algorithm 1. As we will see, generating cuts through the separation oracle of the semidefinite cone fails completely on both Erdős-Rényi graphs and $d-$regular graphs. On the contrary, setting $\mathcal{S}$ to be the columns of a matrix that simultaneously diagonalizes J and A -where A is the adjacency matrix of G in the case of regular graphs or a matrix given by program (CG) in the case of Erdős-Rényi graphs - performs significantly better.

In our discussion on the max cut problem we showed that there is a eigenvalue bound for the max cut value that every graph satisfies, and one might wonder if there such a bound for the theta number. This is indeed the case, albeit only for regular graphs.

Remark 1.1. *Let G be a $d - regular$ graph with n vertices. Let W be the adjacency matrix of G with largest eigenvalue λ_1 and smallest eigenvalue λ_n, then the Lovász theta number $\vartheta(G)$ satisfies*

$$\vartheta(G) \leq \frac{-n\lambda_n}{\lambda_1 - \lambda_n} \tag{1.8}$$

For a proof of this result, see [101]. We conjecture that the objective value of the linear program (LTn) is also upper bound by $\frac{-n\lambda_n}{\lambda_1 - \lambda_n}$ as this was the case in all the experiments we performed for $d-$regular graphs.

QCQPs

We consider more general SDPs obtained as the Shor relaxation [145] of certain QCQPs to test the proposed methodology in three different settings, each highlighting an interesting point. General QCQPs were introduced in Section 1.2, but in this section and Section 1.5 we will consider a more specialized version of them, following [12], of the form

$$
\begin{aligned}
&\inf_{x\in\mathbb{R}^n} x^\top Cx + d_0^\top x + b_0 \\
\text{s.t: } & x^\top A_i x + d_i^\top x \le b_i \;\forall i \in [r], \\
& Dx = t, \\
& l \le x \le u,
\end{aligned} \tag{1.9}
$$

where r denotes the number of quadratic constraints and is at least 1. C, A_i, $i = \{1, \ldots, r\}$ are symmetric matrices, not necessarily PSD, d_i, $i = \{0, \ldots r\}$ are vectors in $\mathbb{R}^n$, D is a $q \times n$ real matrix and $t \in \mathbb{R}^q$. l and u are vectors in $\mathbb{R}^n$ and we assume that $-\infty < l \le u < +\infty$ so that the bounding boxes are non-empty and bounded. If the bounding boxes are of the form $[l, u]^n$ we can do a linear change of variables so that $x \in [0, 1]^n$. Such problems admit the following SDP relaxation:

$$
\begin{aligned}
&\inf_{x\in\mathbb{R}^n, X\in\mathbb{S}^n} \langle C, X\rangle + d_0^\top x + b_0 \\
\text{s.t: } & \langle A_i, X\rangle + d_i^\top x \le \; b_i \;\forall i \in [r], \\
& Dx = t, \\
& 0 \le x_i \le 1 \;\forall i \in [n], \\
& 0 \le X_{i,j} \le 1 \;\forall i, j \in [n], \\
& \begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix} \succeq 0.
\end{aligned} \tag{1.10}
$$

By letting

$\hat{C} := \begin{bmatrix} C & d_0 \\ d_0^\top & b_b \end{bmatrix}$, $\hat{A}_i := \begin{bmatrix} A_i & b_i \\ b_i^\top & c_i \end{bmatrix}$, $i \in \{1, \ldots, m\}$, $\hat{X} := \begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix}$ and by $\hat{X}_{n+1}$ the $n+1$'th column of $\hat{X}$ we can write the previous problem in the SDP form

$$
\begin{aligned}
&\inf_{\hat{X}\in\mathbb{S}^{n+1}} \left\langle \hat{C}, \hat{X} \right\rangle \\
\text{s.t } : & \left\langle \hat{A}_i, \hat{X} \right\rangle \le \; 0 \;\forall i \in [r], \\
& D\hat{X}_{n+1} = t, \\
& 0 \le \hat{X}_{i,j} \le 1 \;\forall i, j \in [n+1], \\
& X_{n+1,n+1} = 1, \\
& \hat{X} \succeq 0.
\end{aligned} \tag{$QSDP$}
$$

In Section 1.5 we test our methodology on random QCQPs using instances generated as in [12].

Quadratic knapsack problem

An interesting point arises whenever the quadratic forms determining the objective and the constraints do not have linear and constant terms, i.e. $d_i = b_i = 0\ \forall i \in \{0, \ldots, r\}$. In that case, our methodology takes $\mathcal{S} = \{v_1, \ldots, v_{n+1}\}$ to be the eigenvectors of a matrix in $\mathbb{S}^{n+1}$ whose $n+1$'th row and column are 0. Hence, the constraints $v^\top \hat{X} v \geq 0$ in program $QSDP$ essentially ignore the last row and column of $\hat{X}$ and amount to the constraints $u_i^\top X u_i \geq 0$ where $u_1, \ldots, u_n$ are a basis of eigenvectors of an aggregation of the $A_i,\ i \in [r]$. This is a weaker constraint than what we actually want, which is $u_i^\top \left(X - xx^\top x\right) u_i \geq 0,\ i \in [n]$.

There are a few approaches we can consider to deal with this issue. For instance, we could choose to overlook it entirely and proceed by relaxing $QSDP$ to an LP, ignoring that the d_i are 0. Alternatively, if we have a linear constraint $d_i^\top x = \alpha_i$, we may set $\hat{C} = \begin{bmatrix} C & d_i \\ d_i^\top & -2\alpha_i \end{bmatrix}$ which shifts the objective by a constant. Finally, and perhaps more interestingly, we may use the constraints $u_i^\top \left(X - xx^\top\right) u_i \geq 0,\ i \in [n]$ directly, which can be equivalently rewritten as:

$$u_i^\top X u_i \geq u_i^\top \left(xx^\top\right) u_i^\top = (u_i^\top x)^2\ \forall i \in [n]. \tag{1.11}$$

These are *second order cone constraints* which result in a second order cone relaxation of program QCQP depending on a set $\mathcal{S}$ of vectors u in $\mathbb{R}^n$. Such a program is both a relaxation of $QSDP$, and a strenghening of the linear relaxation that changes the constraint $\hat{X} \succeq 0$ for $u^\top X u \geq 0$ with $u \in \mathcal{S}$, for any finite set $\mathcal{S}$.

We test these different possibilities in Section 1.5 on instances of the *Quadratic Knapsack problem* [131] which is a QCQP of the form

$$\begin{aligned} &\max_{x \in \mathbb{R}^n} x^\top C x \\ s.t: &\sum_{j=1}^{k} w_j x_j \leq c,\ x \in \{0,1\}^n \end{aligned} \tag{QKP}$$

where $w \in \mathbb{R}^n$, $C \in \mathbb{S}^n$, $c \in \mathbb{R}_+$. It has been noted in the literature that the usual Shor semidefinite relaxation of this program is not very strong [73, 131] and one may add certain valid inequalities which result in the following tighter SDP:

$$\begin{aligned}
&\max_{X\in\mathbb{S}^n} \langle C, X\rangle \\
s.t: &\sum_{j=1}^{n} w_j X_{ij} - cX_{ii} \leq\ 0\ \forall\, i \in [n], \\
&X - diag(X)diag(X)^\top \succeq 0.
\end{aligned} \tag{QKPSDP}$$

Using the idea before and a finite set $\mathcal{S}$ one may further relax this problem to obtain the second order cone program

$$\begin{aligned}
&\max_{X\in\mathbb{S}^n} \langle C, X\rangle \\
s.t: &\sum_{j=1}^{n} w_j X_{ij} - cX_{ii} \leq\ 0\ \forall\, i \in [n], \\
&u^\top X u \geq \left(u^\top diag(X)\right)^2\ \ \forall u \in \mathcal{S}.
\end{aligned} \tag{QKSSOC}$$

Extended trust region

In the previous problems it is not obvious how to linearly combine the matrices $A_i, i \in [r]$, that determine the quadratic forms to form the identity matrix, and hence we cannot apply Theorem 1.3 directly to arbitrary QCQPs. This motivates us to consider a variation where the identity matrix is explicitly one of the constraint matrices. This is the case of the *generalized trust region* problem [98]. That type of QCQPs consists in minimizing a quadratic function over the intersection of the unit ball and some half-spaces:

$$\begin{aligned}
&\min_{x\in\mathbb{R}^n} x^\top C x + 2d^\top x, \\
&\text{s.t: } x^\top x \leq 1 \\
&\quad\ \ Dx \leq t
\end{aligned} \tag{TR}$$

with $C \in \mathbb{S}^n$, $d \in \mathbb{R}^n$, $D \in k \times n$ for some $k \in \mathbb{N}$ and $t \in \mathbb{R}^k$. Notice that the constraint $x^\top x \leq 1$ can be written as $x^\top I_n\ x \leq 1$. In Section 1.5 we test our methodology on a slightly more general version of this problem, where we keep some quadratic constraints. Abusing the language, we still refer to this family of problems as extended trust region problems.

1.5 Experimental results

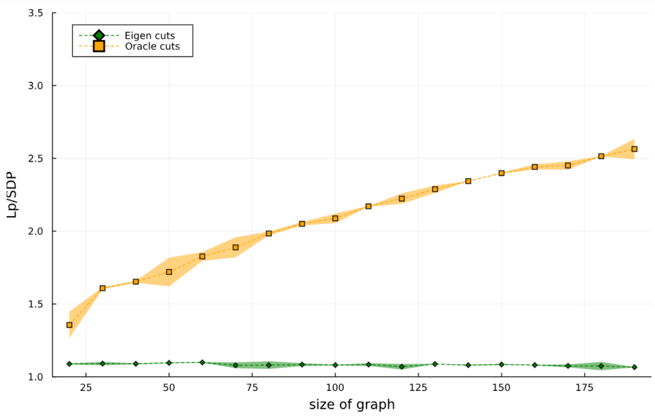
In this section we present experimental results exhibiting the quality of our linear relaxations for the semidefinite relaxation of max cut, Lovász's theta number and on the SDP relaxations of families of QCQPs described in Section 1.4. For each of these problems, we will compare the optimal value of the linear relaxations to the optimal value of the SDP which they respectively relax by means of the quotient of the objective values. We contrast these quotients to the alternative of using Algorithm 1, starting with $\mathcal{S} = e_1, \ldots, e_n$ and iteratively generating cuts using the SDP separation oracle. Whenever we fix an semidefinite program with some label SDP, we denote by $Iter_k(SDP)$ the linear program obtained at the $k-th$ iteration of Algorithm 1. For instance, $Iter_0(SDP)$ is simply dropping the semidefinite constraint of the SDP instance. We define z_n as the optimal value of $Iter_n(SDP)$. For each family of experiments, where we consider a certain SDP, we will denote by $z_{\mathcal{S}}$ the objective value of the corresponding linear relaxation obtained by following the ideas of Section 1.1. z_{sdp} will denote the objective value of the SDP instance. Although we consider different SDPs, there will not be danger of confusion as we caption of the figures and tables indicate which SDP we are addressing.

All of the code used is available at `https://github.com/dderoux/Instance_specific_relaxations`. To solve the resulting optimization programs we have used Mosek [8]. [3]

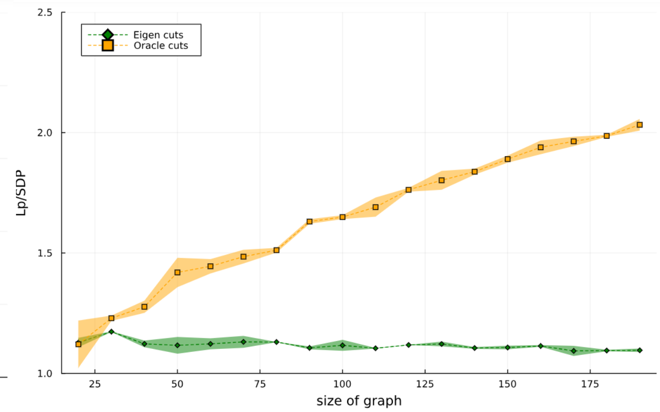
Max cut

Denote by z_{sdp}, z_n and $z_{\mathcal{S}}$ the objective values of programs (GW), $Iter_n(GW)$ and $(SP_{\mathcal{S}})$ with $\mathcal{S}$ chosen as in Subsection 1.2. In this particular case, since the(GW) semidefinite program does not have linear constraints beyond the ones of the diagonal, the identity I is the only constraint matrix and it commutes with the objective matrix W. This means that $\mathcal{S}$ is simply a eigenbasis for the matrix W. As we proved in Theorem 1.5, $(SP_{\mathcal{S}})$ satisfies the eigenvalue bound for max cut. For each n ranging from 20 up to 200 in steps of 10, we generate 5 random graphs and plot the maximum, minimum and median of the quotients $\frac{z_{\mathcal{S}}}{z_{sdp}}$. We present our results for Erdős-Rényi random graphs and $d-$regular random graphs in Figures 1.1 and 1.2 respectively.

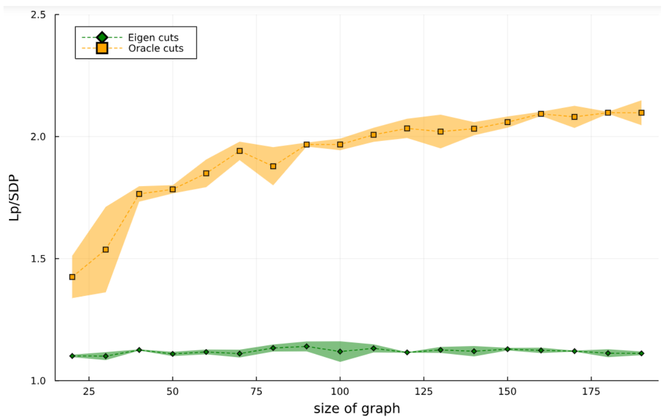
[3] The experiments were performed on a 32 GB RAM ThinkPad Lenovo T490s machine running windows 10 with a Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz.



(a) $p = 0.5$



(b) $p = 0.9$

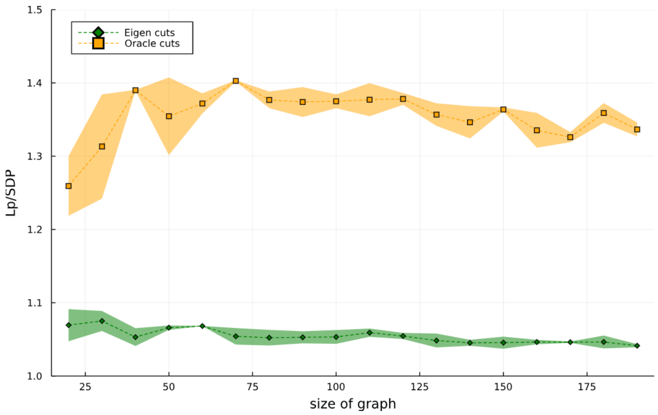


(c) $p = \frac{3\log(n)}{n}$

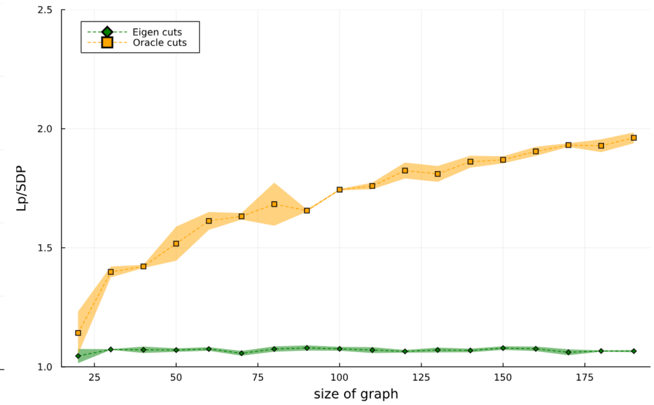
Figure 1.1: Ratio of $\frac{z_S}{z_{sdp}}$ (Eigen cuts) and $\frac{z_n}{z_{sdp}}$ (Oracle cuts) for instances of max cut where the graph has been sampled according to the Erdős-Rényi random model, for different values of p, as n grows.

Comparison with the Eigenvalue Bound

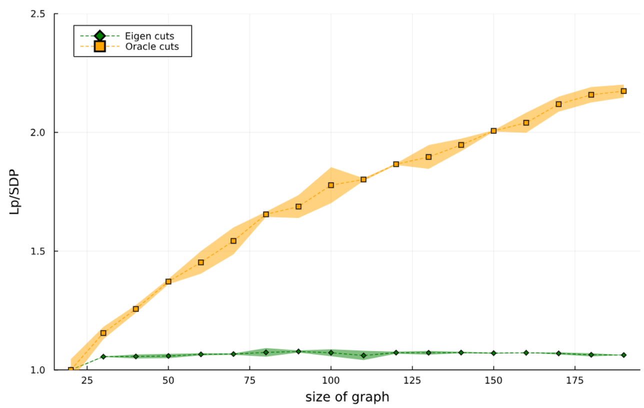
In Tables 1.1 and 1.2, we compare the performance of z_S and the eigenvalue bound $\chi(G) := -n\lambda_n(G)$ on the graphs $\mathcal{G}(n, k, l)$ which we introduced Section 1.3, for different values of n, k and l. Since all of our experiments are random, we present averaged values over 5 instances, as well as the standard deviations of our results. Notice that the eigenvalue bound fails to give a small upper bound on the max cut value for this family of graphs. For the case $n = 400$, the bound fails completely, by giving a worse bound that the trivial upper bound of m for max cut. However, the linear program succeeds, in all of our experiments, to have a quotient of at most 1.04 within the optimal value of the Goemans and Williamson relaxation.



(a) $d = 5$



(b) $d = \sqrt{n}$



(c) $d = \frac{n}{10}$

Figure 1.2: Ratio of $\frac{z_{\mathcal{S}}}{z_{sdp}}$ (Eigen cuts) and $\frac{z_n}{z_{sdp}}$ (Oracle cuts) for instances of max cut where the graph is a random $d-$regular graph, for different values of d, as n grows.

Table 1.1: Ratio of $\chi(G)$ to z_{sdp} and ratio of $z_{SP_{\mathcal{S}}}$ to z_{sdp} for $k = 4$ and $l = 5$.

n	**χ(G)/z_{sdp}: average(sd)**	**$z_{\mathcal{S}}/z_{sdp}$: average(sd)**
64	1.241 (0.008)	1.020 (0.002)
100	1.417 (0.007)	1.017 (0.002)
196	1.760 (0.003)	1.012 (0.001)
400	2.289 (0.003)	1.010 (0.001)

Table 1.2: Ratio of $\chi(G)$ to z_{sdp} and ratio of $z_{\mathcal{S}}$ to z_{sdp} for $k = 6$ and $l = 10$.

n	$\chi(\mathbf{G})/\mathbf{z_{sdp}}$: **average(sd)**	$\mathbf{z}_{\mathcal{S}}/\mathbf{z_{sdp}}$: **average(sd)**
64	1.137 (0.008)	1.029 (0.002)
100	1.278 (0.007)	1.024 (0.001)
196	1.546 (0.005)	1.020 (0.001)
400	1.962 (0.002)	1.013 (0.001)

Lovász theta number

Denote by z_{sdp}, z_n and $z_{\mathcal{S}}$ the objective values of programs Tn, $Iter_n(Tn)$ and LTn with $\mathcal{S}$ chosen as in Subsection 1.2, respectively. For each n ranging from 20 to 200 in steps of 10, we generate 5 random graphs and plot the maximum, minimum and median of the quotients $\frac{z_{\mathcal{S}}}{z_{sdp}}$ and $\frac{z_n}{z_{sdp}}$ for these five instances. In the following subsections, we present these plots for Erdős-Rényi and random d-regular graphs.

Erdős-Rényi random graphs

In Figure 1.3 We plot the mentioned quotients for Erdős-Rényi random graph while we vary p, the probability of connecting two edges.

d-regular random graphs

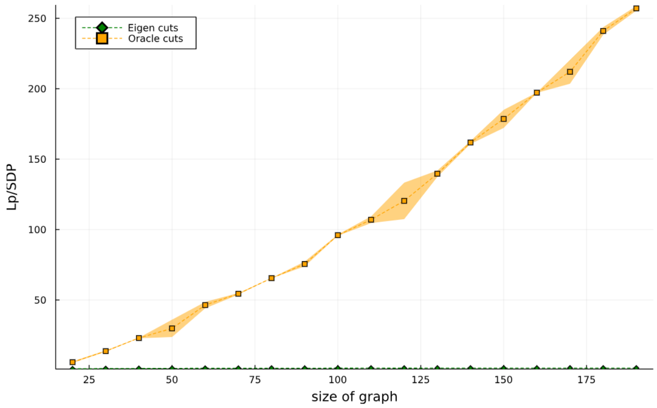
In Figure 1.4 we plot the mentioned quotients for $d-$regular random graph while we vary d.

Quadratically constrained quadratic problems

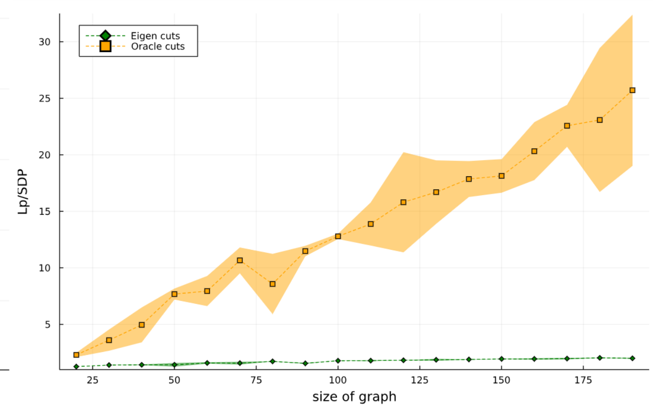
In this subsection we test the proposed methodology on the different QCQPs introduced in Section 1.4.

Random QCQPs

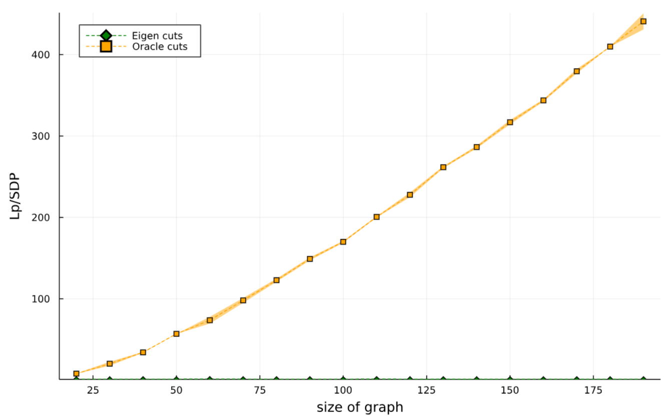
We generate random QCQPs following the review [12], where the authors compare various SDP relaxations of QCQPs in terms of percentage distance to the objective and solution time. For these instances, the x variables are bounded in an unit box $[0, 1]^n$ and the number of variables is varied from 20 up to 100 in steps of 10. The vectors b, t in $\mathbb{R}^{r+1}$ and $\mathbb{R}^q$ respectively and the matrices $D \in \mathbb{R}^{q\times n}$ and $C, A_i \in \mathbb{S}^n$, $i \in \{1, \ldots, r\}$ have entries drawn uniformly and independently at random from an uniform distribution supported in $[-1, 1]$. The vector $d \in \mathbb{R}^{r+1}$ has entries sampled uniformly at random from an uniform distribution supported in $[0, 100]$. Since QCPQs are highly sensitive



(a) $p = 0.5$



(b) $p = 0.9$



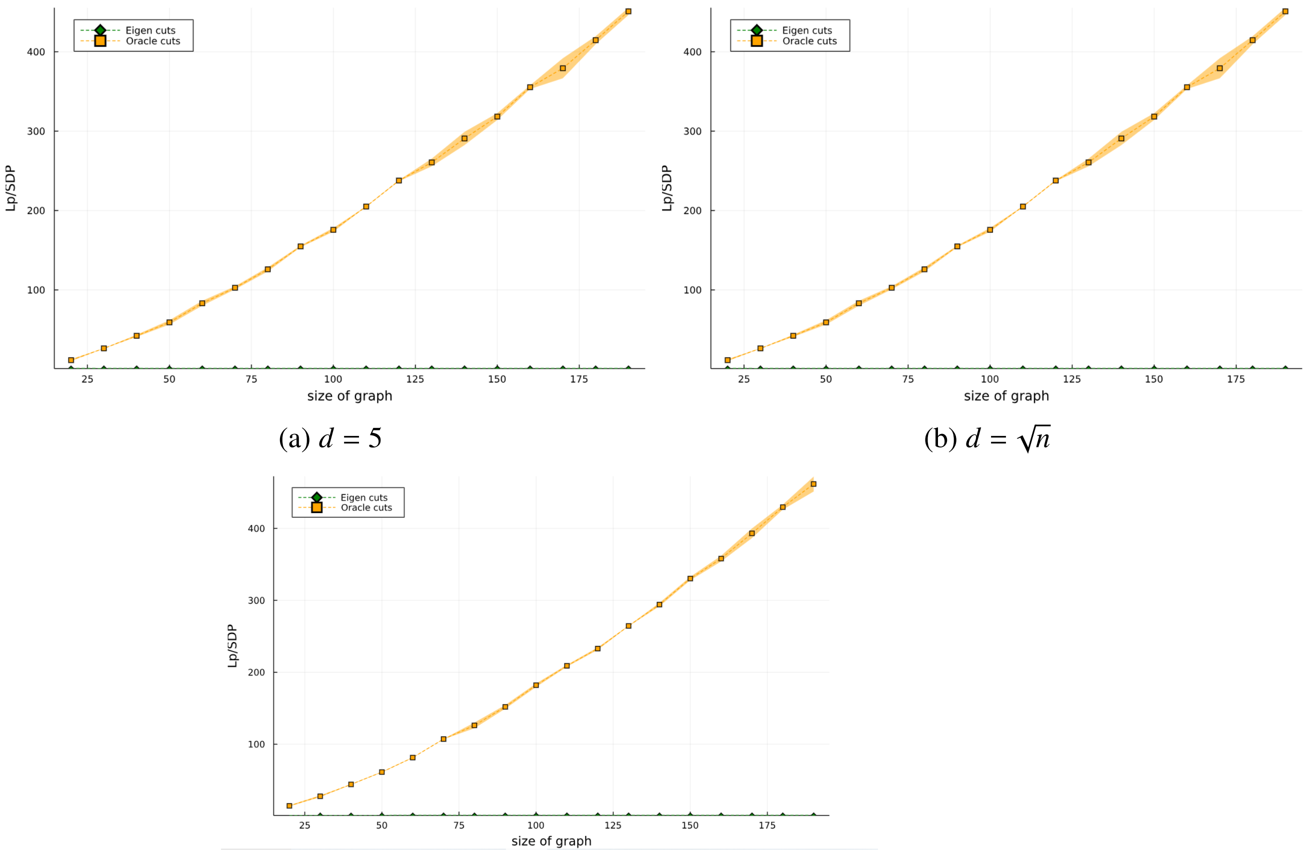
(c) $p = \frac{3\log(n)}{n}$

Figure 1.3: Quotients for the Lovász theta number $\frac{z_S}{z_{sdp}}$ (Eigen cuts) and $\frac{z_n}{z_{sdp}}$ (Oracle cuts) as n grows for Erdős-Rényi random graphs with different values of p.

to the number of quadratic constraints, we test different combinations of number of quadratic and linear constraints, according to the following combinations:

- QCQPs with $r = 1, q = \frac{n}{10}$.
- QCQPs with $r = 1, q = \frac{n}{5}$.
- QCQPs with $r = \frac{n}{2}, q = \frac{n}{10}$
- QCQPs with $r = n, q = \frac{n}{10}$.

Furthermore, we consider different densities Δ for the matrix C, which corresponds to the percentage of nonzero elements of the matrix, on average. For a given combination of these parameters and a value of n we generate 5 random instances and solve the following optimization programs for each:



(a) $d = 5$

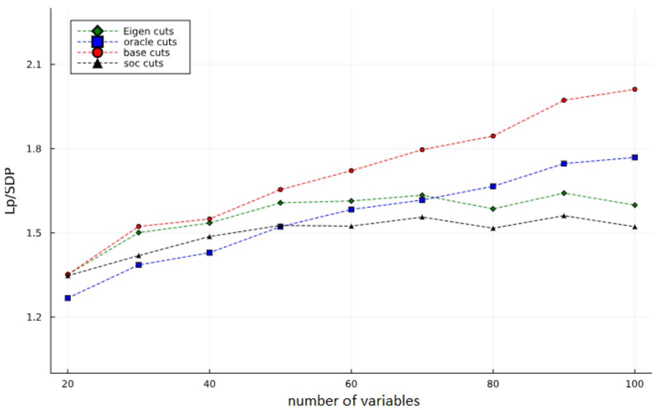
(b) $d = \sqrt{n}$

(c) $d = \frac{n}{10}$

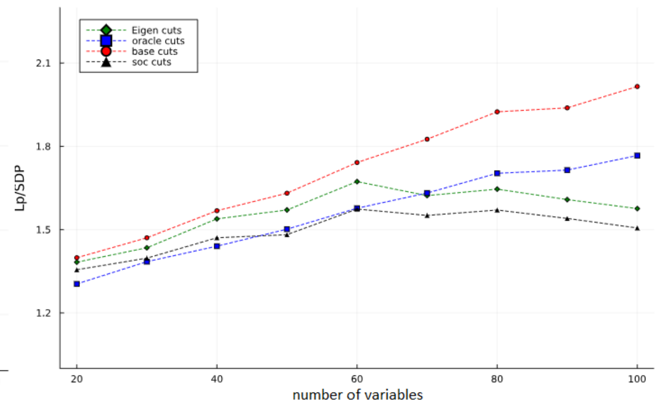
Figure 1.4: Quotients for the Lovász theta number $\frac{z_{\mathcal{S}}}{z_{sdp}}$ and $\frac{z_n}{z_{sdp}}$ as n grows for random $d-$regular graphs with different values of d, as n grows.

- Problem $QSDP$. We denote the objective value of this semidefinite program by z_{sdp}.
- The linear relaxation $L_{\mathcal{S}}$ of $QSDP$ where we let $\mathcal{S}$ the elements of a eigenvector basis of the matrix A_0. We denote the objective value of this problem by $z_{\mathcal{S}}$.
- The LP $Iter_n(QSDP)$. We denote by z_n the objective value of this program.
- The LP $Iter_0(QSDP)$. We denote by z_0 the objective value of this program.
- The second order cone program obtained by dropping the constraint $\hat{X} \succeq 0$ from $QSDP$ adding the constraints (1.11) with $\mathcal{S}$ the elements of a eigenvector basis of the matrix A_0. We denote the objective value of this problem by z_{soc}.

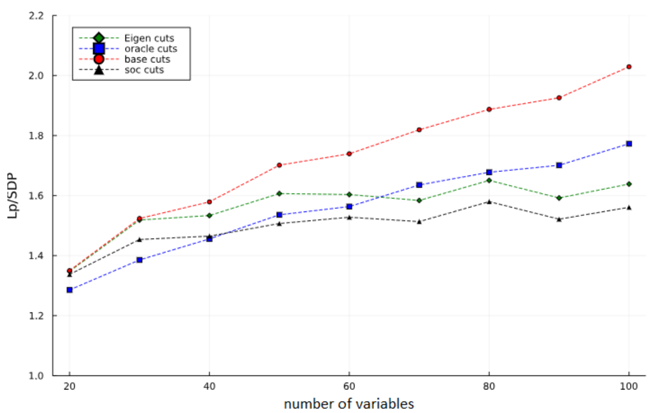
We average the values of ratios $\frac{z_{\mathcal{S}}}{z_{sdp}}$, $\frac{z_n}{z_{sdp}}$, $\frac{z_0}{z_{sdp}}$ and $\frac{z_{soc}}{z_{sdp}}$ over the five instances, and plot the results in Figures 1.5, 1.6, 1.7 and 1.8. We observe that due to randomness, it will not be possible to linearly combine the matrices $A_1, \ldots, A_m, A_{m+1}$ so that they commute with the objective matrix A_0. Therefore, program CG will typically return the 0 matrix, and $\mathcal{S}$ will simply be a basis of eigenvectors of A_0.



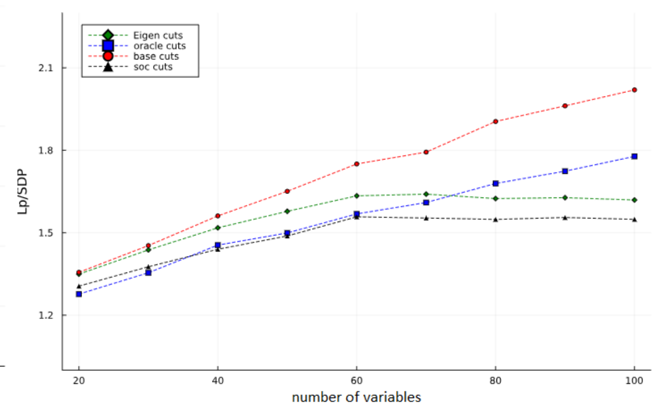
(a) $r = 1, \ q = \frac{n}{5}$



(b) $r = 1, \ q = \frac{n}{10}$



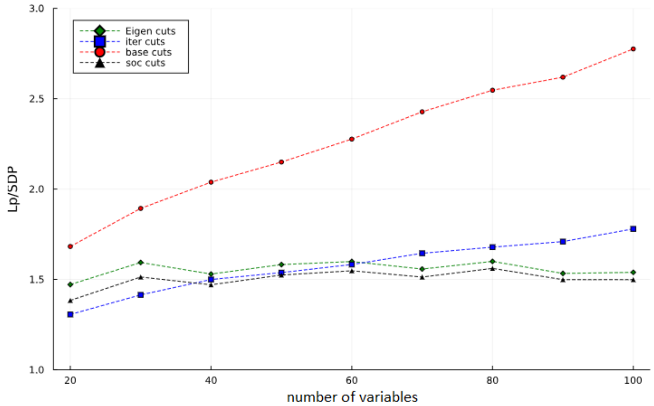
(c) $r = \frac{n}{2}, \ q = \frac{n}{10}$



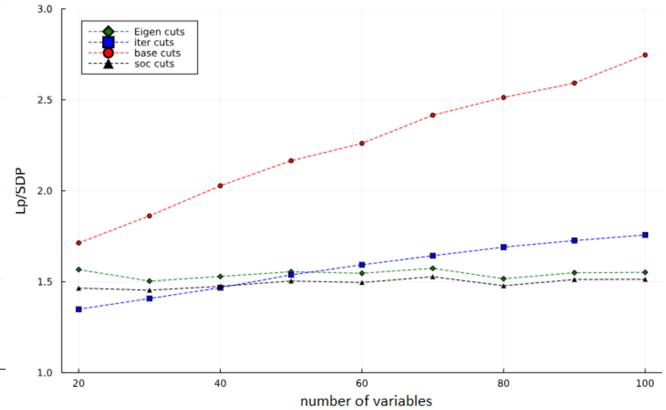
(d) $r = n, \ q = \frac{n}{10}$

Figure 1.5: Quality of the ratios $\frac{z_{\mathcal{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.25.

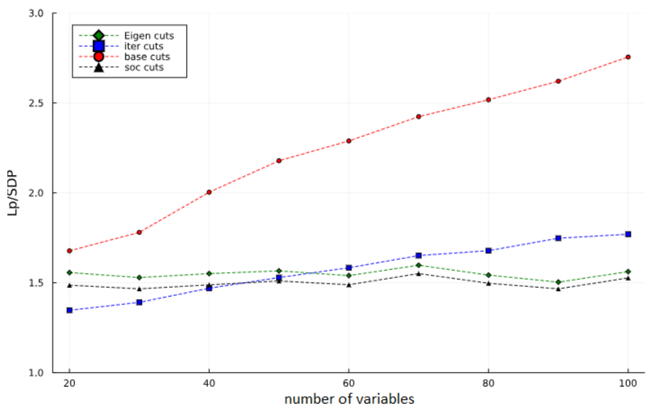
For these instances, the quality of all the relaxations is encouraging, with the trivial relaxation obtained by dropping the semidefinite constraint getting a ratio of at most 4 in all of our experiments. The second order cone relaxation is typically the better as soon as n exceeds 50. Whenever the density increases, we notice that the ratios $\frac{z_{\mathcal{S}}}{z_{sdp}}$ and $\frac{z_{soc}}{z_{sdp}}$ get closer and closer, hinting at that the second order cone relaxation is not much stronger than the linear relaxation. Although the LP $Iter_n$ achieves a better ratio for small n, this is no longer true for larger values of n. In addition, notice that for a value of n this LP requires solving n LPs and n eigenvector decompositions.



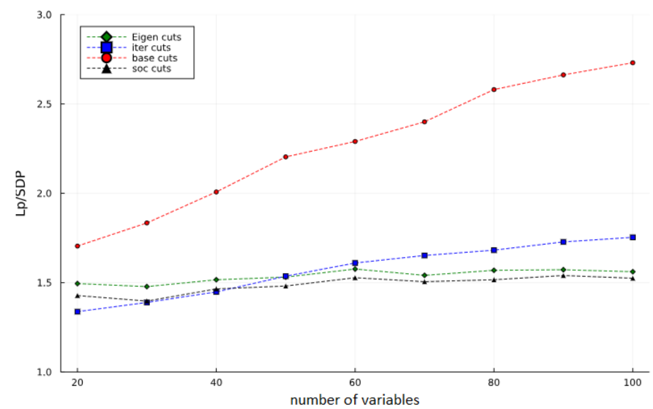
(a) $r = 1,\ q = \frac{n}{5}$



(b) $r = 1,\ q = \frac{n}{10}$



(c) $r = \frac{n}{2},\ q = \frac{n}{10}$



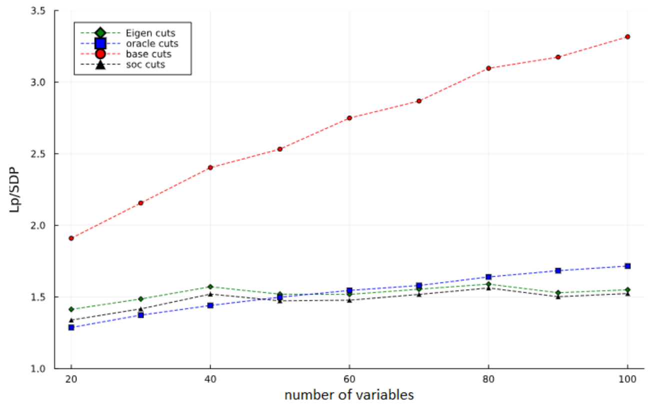
(d) $r = n,\ q = \frac{n}{10}$

Figure 1.6: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.5.

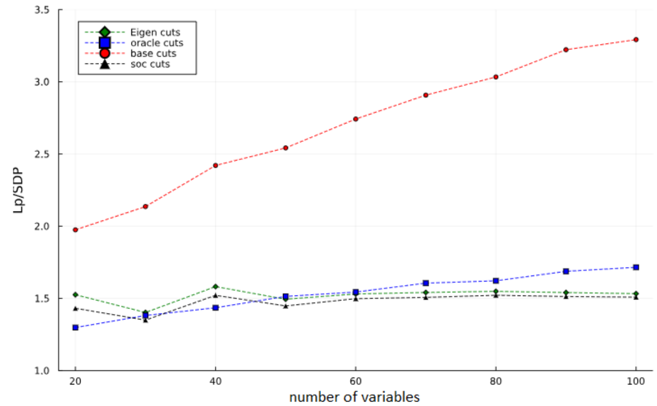
Extended trust region problems

We now consider instances of the extended trust region problem with extra quadratic constraints, as presented in Subsection 1.4. These instances are the same as in the previous subsection, but with the added quadratic constraint $x^\top I_n x \leq 1$. We present our results in figures 1.9, 1.10, 1.11 and 1.12.

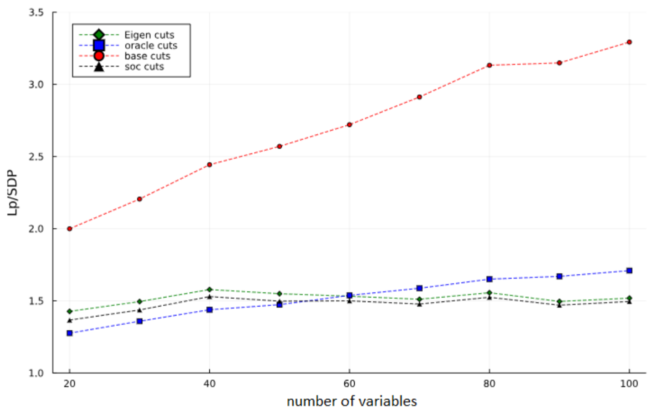
The results for these experiments are similar across the different densities. In all of our experiments, the second order cone relaxation and the linear relaxation L_S of the extended trust region problem are very strong with the ratio to the SDP relaxation being very close to 1. Moreover, this ratio does not get worse as n increases, quite in sharp contrast to the base relaxation $Iter_0$ of objective value z_0 and the LP $Iter_n$, which gets a ratio worse than 50 whenever n exceeds 100. for these instances, program L_S



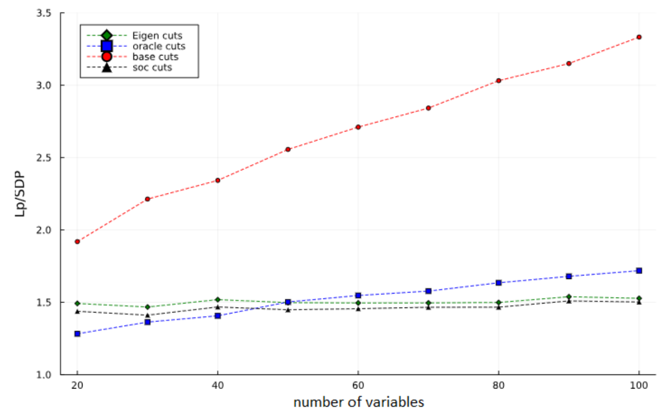
(a) $r = 1,\ q = \frac{n}{5}$



(b) $r = 1,\ q = \frac{n}{10}$



(c) $r = \frac{n}{2},\ q = \frac{n}{10}$



(d) $r = n,\ q = \frac{n}{10}$

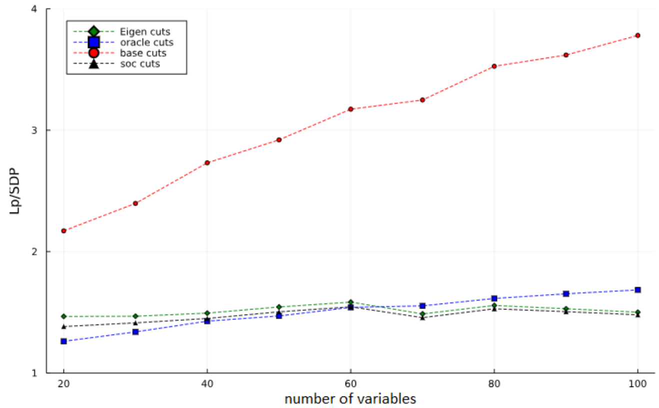
Figure 1.7: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.75.

specialized to the extended trust region problem and program QKSSOC certify the dual bounds provided by Theorem 1.3, which we believe is the reason of the effectiveness of these relaxations.

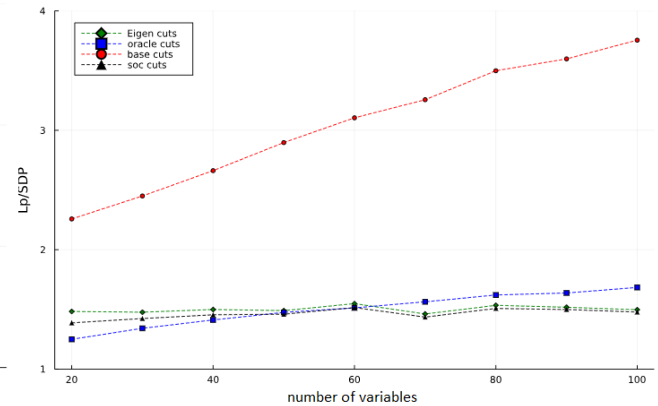
Quadratic knapsack problem

We now consider instances of the quadratic knapsack problem as presented in Subsection 1.4. In this family of problems, the linear term d_0 in the objective is 0, and therefore we can consider the different strategies mentioned in Section 1.4. Hence, for each instance we solve 5 programs, as follows:

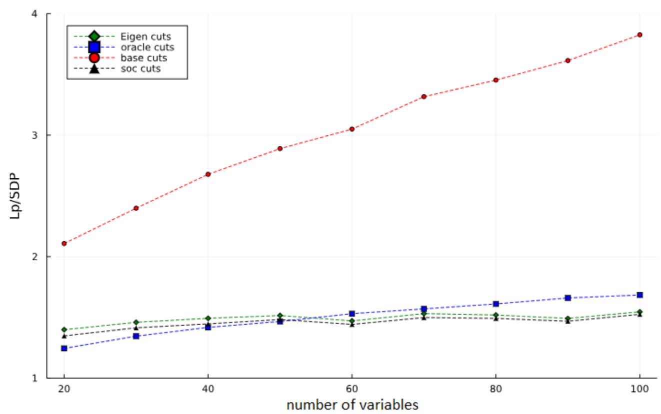
- Problem QKPSDP. We denote the objective value of this semidefinite program by z_{sdp}.



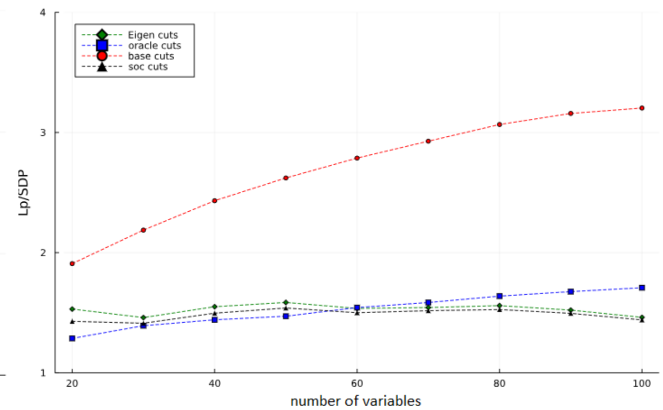
(a) $r = 1,\ q = \frac{n}{5}$



(b) $r = 1,\ q = \frac{n}{10}$



(c) $r = \frac{n}{2},\ q = \frac{n}{10}$

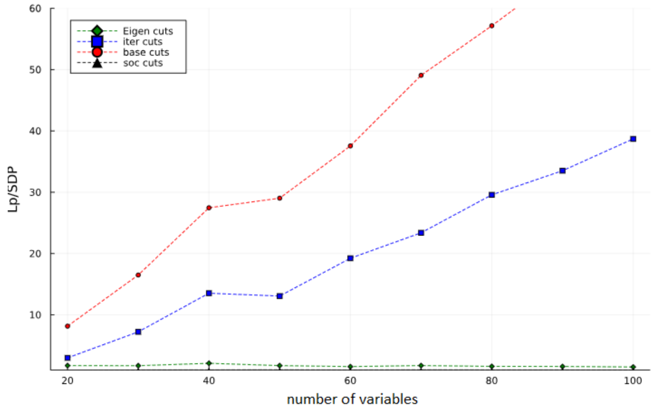


(d) $r = n,\ q = \frac{n}{10}$

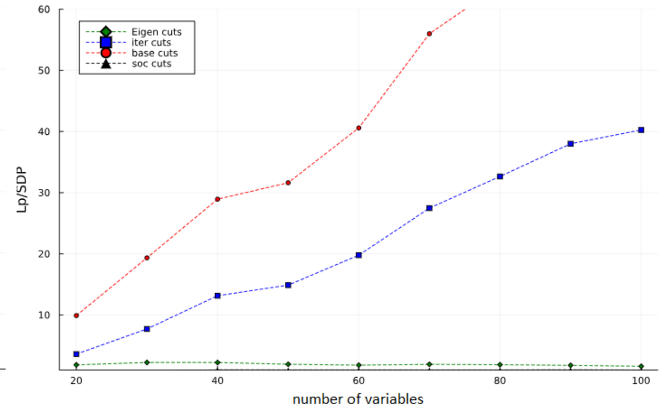
Figure 1.8: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 1.

- The linear relaxation L_S of QKPSDP where we let S the elements of a eigenvector basis of the matrix A_0. We denote the objective value of this problem by z_S.
- The LP $Iter_n(QSDP)$. We denote by z_n the objective value of this program.
- The LP $Iter_0(QSDP)$. We denote by z_0 the objective value of this program.
- The second order cone relaxation of QKPSDP given by program QKSSOC.

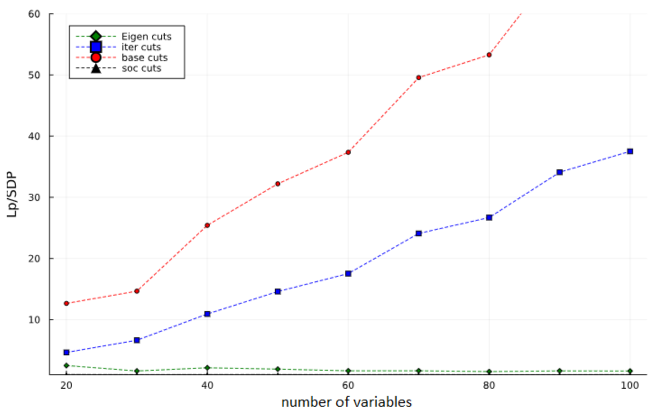
The instances were generated following [131], who specify instances that have become the standard to computationally test this optimization problem. Namely, we first set a *density* value $\Delta \in [0, 1]$, which corresponds to the percentage of nonzero elements of the matrix C. Each weight $w_j,\ j \in [n]$ is uniformly randomly distributed in $[1, 50]$.



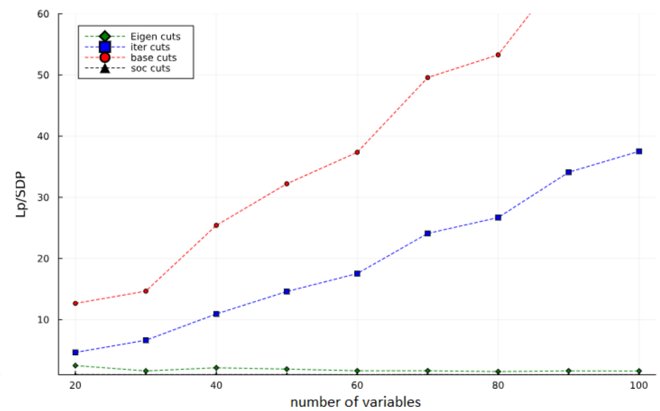
(a) $r = 1,\ q = \frac{n}{5}$



(b) $r = 1,\ q = \frac{n}{10}$



(c) $r = \frac{n}{2},\ q = \frac{n}{10}$

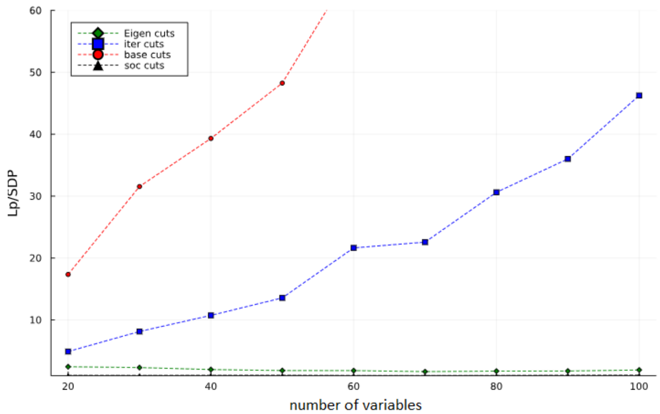


(d) $r = n,\ q = \frac{n}{10}$

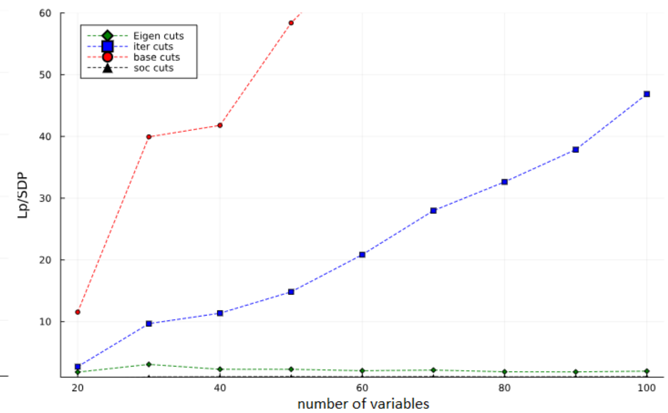
Figure 1.9: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.25.

The ij entry of A_0 equals the ji entry and is nonzero with probability Δ, in which case it is uniformly distributed in $[1, 100]$, $i, j \in [n]$. The capacity c of the knapsack is taken uniformly at random from the interval $[50, \sum_{j=1}^{n} w_j]$. We present our results in Figure 1.13.

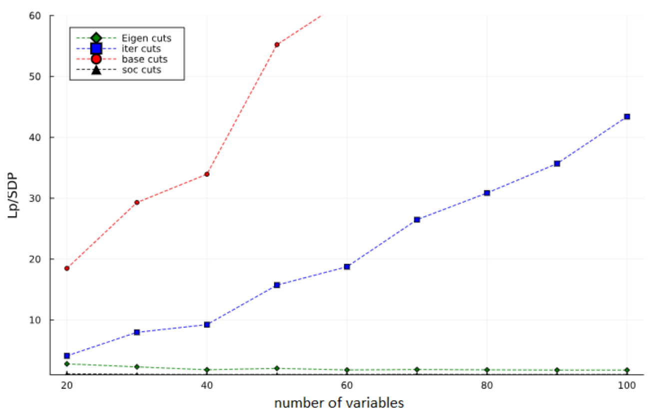
For this family of problems, all relaxations are within reasonable bounds of the SDP objective value. It is nonetheless appealing that the second order cone relaxation performs very well, with the ratio to the objective of the SDP nearly 1, regardless of the value of n. The relaxation L_S seems to perform similarly to $Iter_n$.



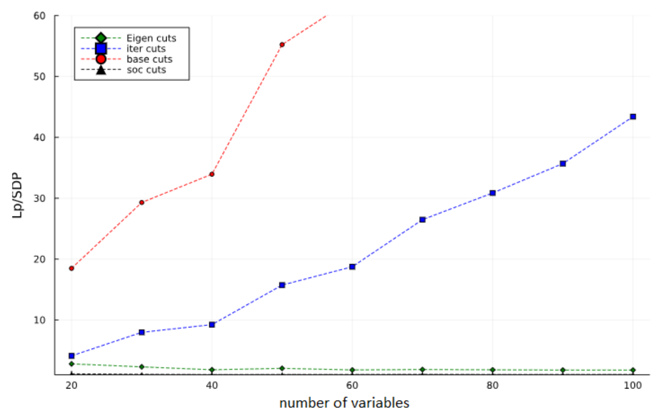
(a) $r = 1,\ q = \frac{n}{5}$



(b) $r = 1,\ q = \frac{n}{10}$



(c) $r = \frac{n}{2},\ q = \frac{n}{10}$



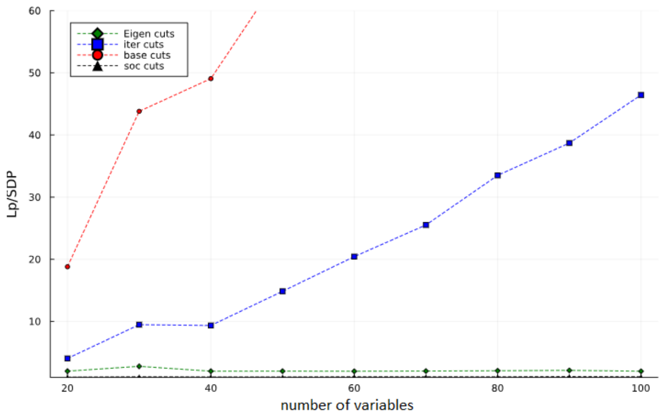
(d) $r = n,\ q = \frac{n}{10}$

Figure 1.10: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.5.

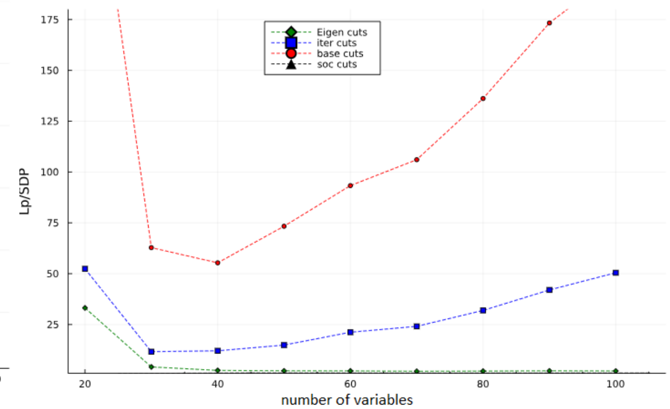
Computational time considerations

Algorithm 1 offers a meta-algorithm to solve semidefinite programs. Ideally, choosing appropriate starting sets $\mathcal{S}$ to initialize the algorithm will result in better solving times. It is critical then that solving program $L_{\mathcal{S}}$ or a second order cone strenghening takes significantly less time than solving the SDP. In what follows, we report solving times of the different programs proposed.

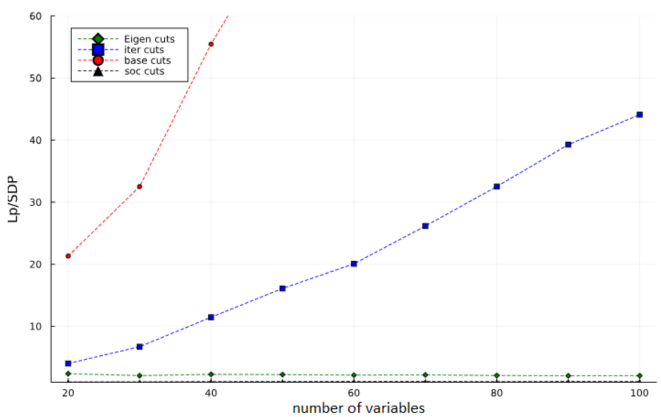
For the max cut and the Lovász theta number we consider Erdős-Rényi random graphs on 270 and 200 vertices respectively. The probability of adding an edge between two vertices is set to $p = 0.75$. We repeat the experiments for 3 instances and report the average solving time and worst ratio of the LP to the SDP objective value among the three instances.



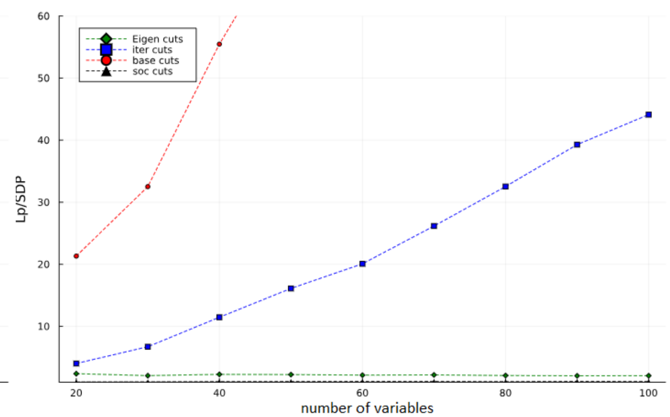
(a) $r = 1, \ q = \frac{n}{5}$



(b) $r = 1, \ q = \frac{n}{10}$



(c) $r = \frac{n}{2}, \ q = \frac{n}{10}$



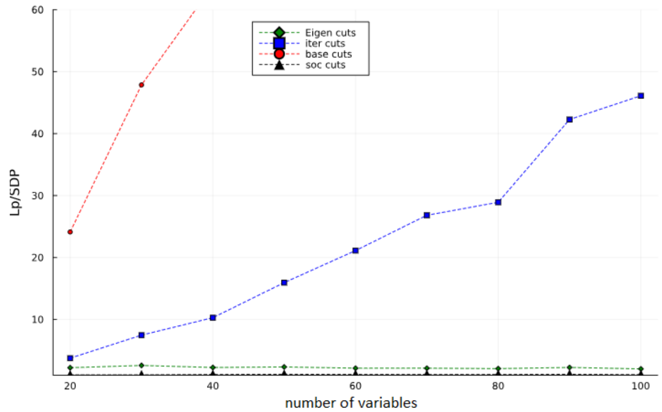
(d) $r = n, \ q = \frac{n}{10}$

Figure 1.11: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.75.

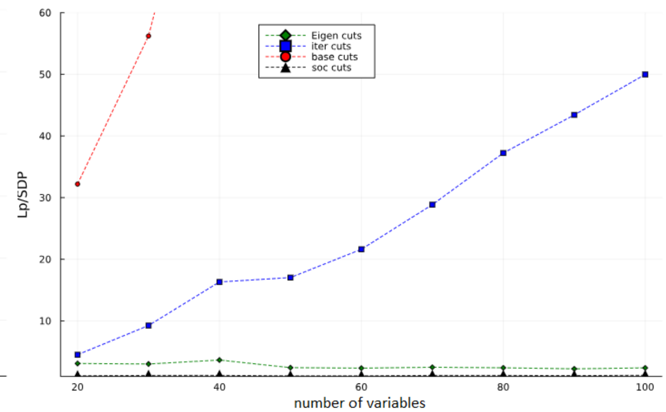
Max cut : The worst ratio found was 1.08. The average solving time of the SDP was 0.47 seconds. The average solving time of the LP was 9.77 seconds.

Theta number : The worst ratio found was 6.7. The average solving time of the SDP was 2994 seconds. The average solving time of the LP was 39 seconds.

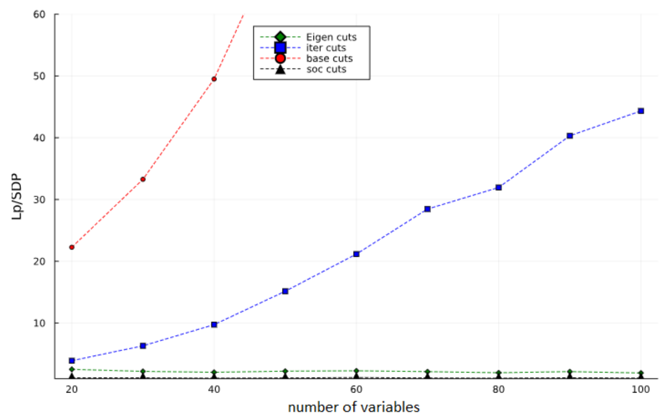
We proceed by reporting the solving times for the quadratic knapsack, random QCQPs and the Extended Trust Region problem. We consider problems with 270 variables. For the Trust Region and random QCQPs we set the number of quadratic constraints to 10, and the number of linear constraints to 20. For each problem, we generate 3 instances as described previously, setting the density Δ to 0.75. We report the average



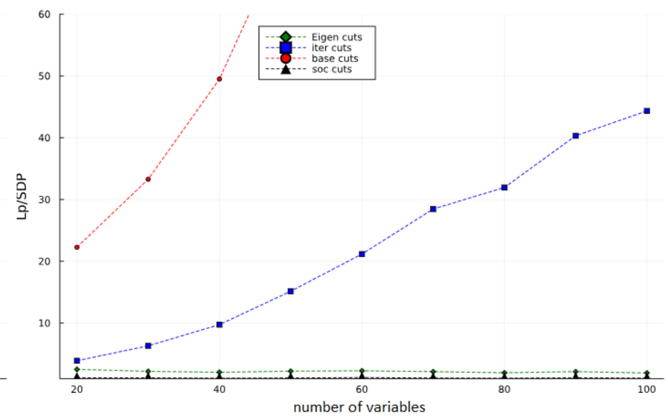
(a) $r = 1, \ q = \frac{n}{5}$



(b) $r = 1, \ q = \frac{n}{10}$



(c) $r = \frac{n}{2}, \ q = \frac{n}{10}$



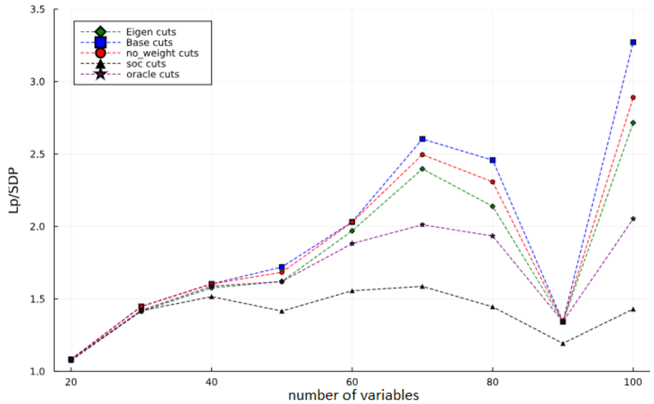
(d) $r = n, \ q = \frac{n}{10}$

Figure 1.12: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 1.

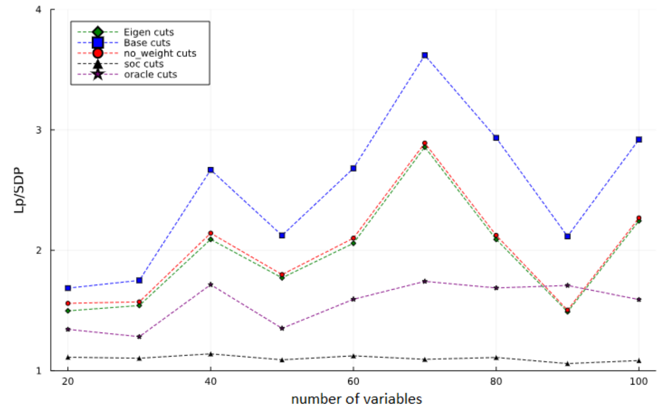
solving time, worst ratio of the LP to the SDP objective and worst ratio of the SOC to the SDP value among the three instances.

Trust region : The average solving time of the SDP was 7476 seconds. The average solving time of the LP was 216 seconds. The average solving time of the SOC was 8.9 seconds. The worst ratio found for the LP was 2.49, and the worst ratio found for the SOC was 1.17.

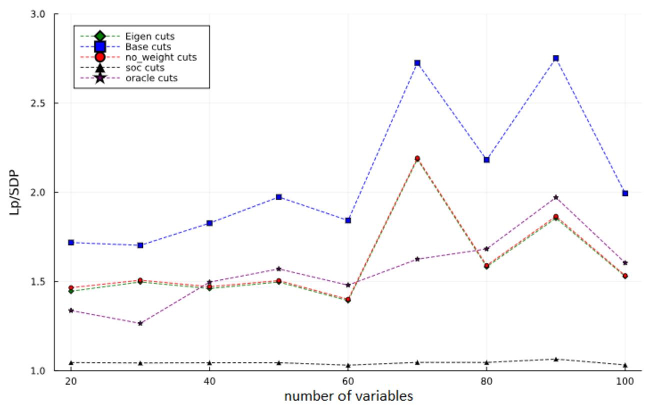
Random QCQPs : The average solving time of the SDP was 6510 seconds. The average solving time of the LP was 13 seconds. The average solving time of the SOC was 21 seconds. The worst ratio found for the LP was 1.49, and the worst ratio found for the SOC was 1.48.



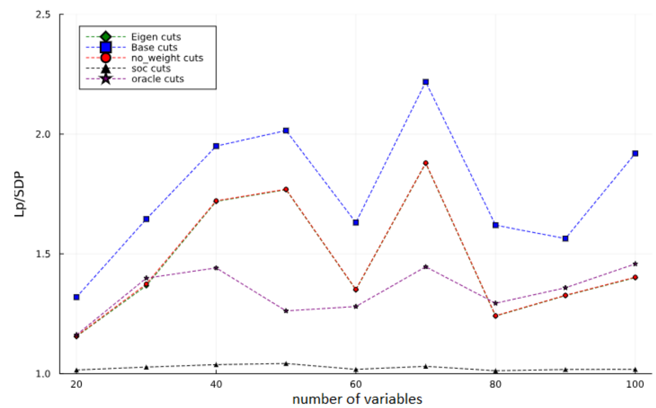
(a) $\Delta = 0.05$



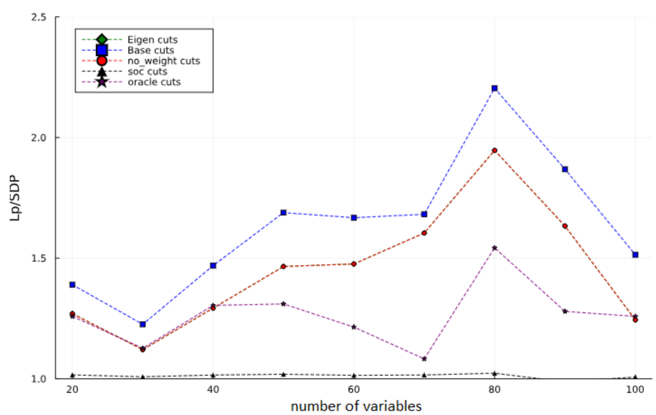
(b) $\Delta = 0.25$



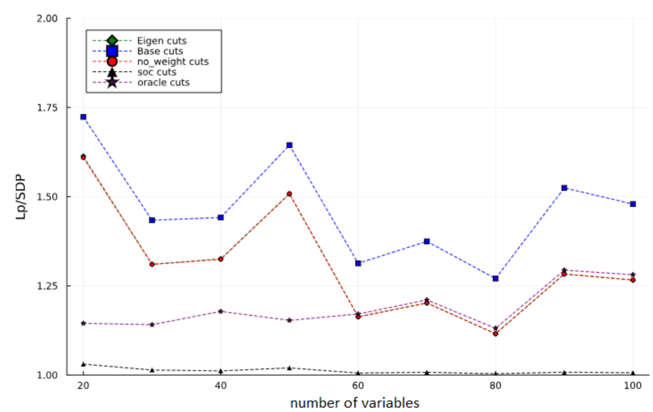
(c) $\Delta = 0.5$



(d) $\Delta = 0.75$



(e) $\Delta = 0.95$



(f) $\Delta = 1$

Figure 1.13: Quality of the ratios $\frac{z_S}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$, (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts) and $\frac{z_{soc}}{z_{sdp}}$ for instances of the quadratic knapsack problem with density different densities.

Knapsack : The average solving time of the SDP was 7422 seconds. The average solving time of the LP was 15 seconds. The average solving time of the SOC was 20 seconds. The worst ratio found for the LP was 1.844, and the worst ratio found for the SOC was 1.01.

It is noteworthy that solving the max cut SDP is faster by 4 orders of magnitude than the all of the other semidefinite programs considered in this section. In addition, it is quite surprising that the SOC relaxations of the QCQPs have solving times comparable to that of the LPs. In particular, the solving time of the SOC is two orders of magnitude faster than the LP for the trust region problems. We point out that very strong, fast and scalable, specialized algorithms for semidefinite programs such as the max cut problem and the Lováz theta number exist, such as [63, 156, 158], and therefore alternatives such as an outer approximation algorithm as 1 might not be appealing for these problems.

1.6 Underlying optimization problems

In Section1.5, we focused on comparing the objective of the semidefinite relaxation to that of a LP or second order cone relaxation. However, the semidefinite relaxation GW is the relaxation of combinatorial problem, and hence a natural question is whether the linear programs proposed give good solutions for the actual underlying problem. As far as we are aware, there is no algorithm to round the Lovász theta number to obtain an independent sub-graph in a graph, but we can certainly round solutions of the LPs for the max cut problem and for a problem which is not combinatorial: the sparse PCA problem.

In what follows, we present experimental results showing the quality of actual graph cuts obtained using programs $SP_{\mathcal{S}}$ and $SD_{\mathcal{S}}$ and the subsequent rounding using vectors for $\mathcal{S}$ which we describe in the next subsection. We present as well the ratios $\frac{\frac{1}{2}m+\frac{1}{4}Z_{SP_{\mathcal{S}}}}{\frac{1}{2}m+\frac{1}{4}Z_{SD_{\mathcal{S}}}}$ and $\frac{\frac{1}{2}m+\frac{1}{4}Z_{SP_{\mathcal{S}}}}{\frac{1}{2}m+\frac{1}{4}Z_{GW}}$ which we call *LP-gap* and *optimality gap*, respectively.

These experiments are done for for Erdős-Rényi random graphs, 16 graphs taken from TSPLIB, 14 graphs from the network repository. We compare them with graph cut values obtained by Mirka and Williamson on the same graph instances [4]. For the Trevisan's algorithm see [150]. The simple and the sweep algorithms are modifications of Trevisan's algorithm presented in [114]. The greedy algorithm for max cut is

[4]The results included here were obtained by direct communication with the authors, and will be included in a future version of [114].

folklore, and the specifics are detailed in the previous reference as well. In the second subsection, we include results for the sparse PCA problem.

Finding cuts from $SD_{\mathcal{S}}$

A particular advantage of program $SD_{\mathcal{S}}$ is that its solutions are also feasible for GW and hence can be employed using the rounding algorithm in [64] to obtain feasible cuts, as the next observation shows.

Observation 1.5. *Let X be feasible for program $SD_{\mathcal{S}}$ where $\mathcal{S}$ is finite and contains the standard basis $e_1, \ldots, e_n$ of $\mathbb{R}^n$. Then, we can find a cut of value at least $0.878\left(\frac{1}{2}m + \frac{1}{4}z_{SD_{\mathcal{S}}}\right)$.*

Proof. Notice that since that for i in $1, \ldots, n$, the matrix $e_i(e_i)^T$ is the matrix of all zeros with a 1 on its ii entry. Further, since W has 0 on its diagonal we may assume that any optimal solution to $SD_{\mathcal{S}}$ has all diagonal entries equal to 1. It follows that such a solution is feasible for the Goemans and Williamson semidefinite program, and hence we can use the rounding procedure described in [64] to obtain a cut with the claimed value. □

In our experiments, we will be using different vectors for programs $SD_{\mathcal{S}}$ and $SP_{\mathcal{S}}$, so to avoid any confusion we denote by $\mathcal{S}'$ the set of vectors used for the relaxation $SD_{\mathcal{S}'}$. An interesting source of vectors for $\mathcal{S}'$ for program $SD_{\mathcal{S}'}$ are the eigenvectors of an optimal solution $\hat{X}$ to $SP_{\mathcal{S}}$ where $\mathcal{S} = \mathcal{E}(W)$. Although $\hat{X}$ is not PSD in general, we can take the eigenvectors $x_i, i \in [k]$ that correspond to positive eigenvalues of $\hat{X}$, and let $\mathcal{S}' = \{x_i, \ldots, x_k\}$. We observe that the computational cost of this procedure comes from solving the LP (or the SDP), whereas producing a random vector in the unit ball to find a cut is computationally cheap. Hence we produce 100 random vectors and report the value of the best cut found using those vectors in all of our experiments. We note that in [114] the same method is used to find cuts from a solution to GW.

Erdős-Rényi random graphs

Letting $\mathcal{S} = \mathcal{E}(W)$, we know, thanks to Corollary 1.2 ,that the LP gap and hence the optimality gap converge to 1 as n grows with high probability for Erdős-Rényi graphs when np is not very small. We empirically evaluate the size of cuts produced by $SD_{\mathcal{S}}$ and the subsequent rounding and present these results in table 1.3, together with the results obtained by Mirka and Williamson on the same graphs. Surprisingly, our procedure generates the best cut value on 15 out of the 20 instances reported in

Table 1.3: Optimality gap, LP-gap, and other algorithms for Erdős-Rényi random graphs for the max cut problem.

Graph	Optimality gap	LP Gap	LP cut value	Greedy	Trevisan	Simple	sweep	GW	OPT
$G(50,0.1)$	1.076	1.275	114	104	**116**	112	113	113	116
$G(50,0.25)$	1.021	1.119	208	199	**210**	209	**210**	199	211
$G(50,0.5)$	1.019	1.071	**373**	357	363	372	**373**	371	377
$G(50,0.75)$	1.014	1.053	**522**	510	510	510	520	513	524
$G(100,0.1)$	1.054	1.213	**304**	272	284	294	296	301	311
$G(100,0.25)$	1.023	1.102	**772**	732	747	766	766	769	782
$G(100,0.5)$	1.013	1.048	**1394**	1350	1356	1391	1391	1375	1416
$G(100,0.75)$	1.006	1.032	**2025**	1978	2008	2014	2019	1982	2035
$G(200,0.1)$	1.031	1.151	**1275**	1204	1242	1257	1257	1267	1296
$G(200,0.25)$	1.015	1.074	**2926**	2796	2892	2920	2922	2861	2975
$G(200,0.5)$	1.007	1.031	5473	5388	5397	5441	5451	**5489**	5542
$G(200,0.75)$	1.005	1.025	**7860**	7731	7743	7848	7852	7835	7904
$G(350,0.1)$	1.018	1.105	3645	3542	3548	3661	**3667**	3513	3735
$G(350,0.25)$	1.009	1.050	**8613**	8344	8426	8535	8553	8349	8709
$G(350,0.5)$	1.006	1.027	**16327**	16110	16225	16253	16298	15904	16482
$G(350,0.75)$	1.003	1.017	**23818**	23613	23678	23791	23811	23674	23967
$G(500,0.1)$	1.0143	1.087	7326	7174	7174	7314	7314	**7372**	7532
$G(500,0.25)$	1.007	1.040	**17177**	16833	17014	17045	17075	16766	17399
$G(500,0.5)$	1.004	1.022	**32978**	32557	32862	32952	32960	32713	33234
$G(500,0.75)$	1.003	1.014	**48326**	47995	47995	48244	48255	47597	48576

[114]. Furthermore, we obtain cuts better than the ones produced by the Goemans and Williamson rounding procedure -which first solves a semidefinite program- on 18 out of the 20 instances. Our results are reported in Table 1.3.

Relevant instances

We test our algorithm on 16 complete graphs from TSPLIB [138], an online library of sample instances for the Travelling Salesman Problem and related graph problems. These graph are complete weighted graphs and hence we do not report the number of edges of each graph. In Table 1.4 we present the optimality gap and the LP gap found for these graphs. We report as well the size of the cuts obtained following the cut generation technique presented in 1.6. Our algorithm finds the best cut in 7 of 16 instances, and a better (or equal) cut than the GW relaxation on 14 out of the 16 instances. We then present the same quotients on 14 graph instances taken from the Network Repository [141] in Table 1.5. Since of these graphs are weighted and some are not, we do not report the number of edges of each graph. Our algorithm finds the best cut in 8 of the 14 instances, and a better (or equal) cut than the GW relaxation on

10 out of the 14 instances.

Sparse PCA

Principal component analysis (PCA) is a popular tool in the statistical and machine learning literature used for dimensionality reduction, data visualisation and analysis. The core idea is to find linear combinations of the variables that correspond to directions of maximal variance, called the principal components. Finding these can be accomplished by means of a singular value decomposition. For more details about applications we refer the reader to [1]. One of the main disadvantages of PCA is that the weights in the linear combination of the variables are typically non-zero, thus hindering interpretation and applicability to certain problems, such as biology or finance. In these cases it is desirable to have components that are linear combination of just a few variables. Such components are called *sparse* components, and many different techniques have been proposed to obtain them. Cadima and Jolliffe [30] propose an ad-hoc technique consisting in setting to 0 loadings that are small enough. Zou, Hastie, and Tibshirani [164] write the PCA problem as a regression optimization problem, and then impose an ℓ_1 penalization term to encourage sparse solutions. Following the ideas of the previous section, we relax by dropping the constraint $X \succeq 0$ and imposing $v^\top X v \geq 0$ for all $v \in \mathcal{S}$ where we set $\mathcal{S} = \mathcal{E}(C)$. This yields the linear program

$$\begin{aligned} \max_{X \in \mathbb{S}^n} \ & \langle C, X \rangle \\ \text{s.t: } & tr(X) = 1,\ \vec{1}^\top |X| \vec{1} \leq k, \\ & v^\top X v \geq 0 \ \forall v \in \mathcal{S} \\ & X_{ii} \geq 0,\ X_{ii} + X_{jj} - 2\alpha X_{ij} \geq 0 \ \forall\, i, j \in [n] \\ & -1 \geq X_{ij} \geq 1,\ \forall i, j \in [n]. \end{aligned} \tag{LSPCA}$$

The linear constraints on X added on the last two lines are valid for X positive semidefinite since the cone of positive semidefinite matrices is self dual, as long as $\alpha \in [0, \sqrt{2}]$. We mention that these constraints are suggested in [157].

We test the quality of our relaxation in terms of sparsity of the recovered components in the examples presented in [41] and in terms of explained variance. Explained variance is the typical way to evaluate the performance of a PCA decomposition. However, we point out that there does not seem to be a consensus in the literature for what the "explained variance" for a sparse PCA decomposition is. The reason, in a nutshell, is that components recovered in the sparse case are not mutually orthogonal [31]. In

Table 1.4: Optimality gap, LP-gap, and other algorithms for some graphs on the TSPLIB graph database [138] for the max cut problem.

Graph	Optimality gap	LP Gap	LP cut value	Greedy	Trevisan	Simple	Sweep	GW
bayg29	1.027	1.139	$\mathbf{4.269 \times 10^4}$	$3.837e^4$	$4.225e^4$	$\mathbf{4.269 \times 10^4}$	$\mathbf{4.269 \times 10^4}$	$\mathbf{4.269 \times 10^4}$
bays29	1.025	1.139	$\mathbf{5.399 \times 10^4}$	4.831×10^4	5.393×10^4	5.369×10^4	$\mathbf{5.399 \times 10^4}$	5.386×10^4
berlin52	1.049	1.186	$\mathbf{4.706 \times 10^5}$	4.532×10^5	4.616×10^5	4.465×10^5	4.681×10^5	4.522×10^5
bier127	1.036	1.170	2.323×10^7	2.162×10^7	2.300×10^7	2.322×10^7	$\mathbf{2.330 \times 10^7}$	2.320×10^7
brazil58	1.031	1.181	2.313×10^6	$\mathbf{2.319 \times 10^6}$	$\mathbf{2.319 \times 10^6}$	2.315×10^6	2.315×10^6	2.180×10^6
brg180	1.009	1.029	4.563×10^7	4.118×10^7	$\mathbf{4.616 \times 10^7}$	4.531×10^7	4.551×10^7	4.330×10^7
ch130	1.021	1.127	$\mathbf{1.888 \times 10^6}$	1.777×10^6	1.885×10^6	$\mathbf{1.888 \times 10^6}$	$\mathbf{1.888 \times 10^6}$	1.887×10^6
ch150	1.024	1.109	2.525×10^6	2.500×10^6	2.521×10^6	$\mathbf{2.526 \times 10^6}$	$\mathbf{2.526 \times 10^6}$	2.434×10^6
d198	1.055	1.279	1.289×10^7	9.635×10^6	1.286×10^7	1.292×10^7	$\mathbf{1.293 \times 10^7}$	$\mathbf{1.293 \times 10^7}$
eil101	1.0218	1.133	$\mathbf{1.071 \times 10^5}$	1.052×10^5	1.070×10^5	1.063×10^5	1.064×10^5	1.058×10^5
gr120	1.011	1.158	2.156×10^6	2.123×10^6	2.147×10^6	2.156×10^6	$\mathbf{2.157 \times 10^6}$	2.154×10^6
gr137	1.013	1.192	3.068×10^7	2.241×10^7	3.044×10^7	3.066×10^7	$\mathbf{3.070 \times 10^7}$	$\mathbf{3.070 \times 10^7}$
gr202	1.030	1.180	$\mathbf{1.599 \times 10^7}$	1.372×10^7	1.533×10^7	1.559×10^7	1.593×10^7	1.581×10^7
gr96	1.022	1.130	1.165×10^7	8.967×10^6	1.156×10^7	$\mathbf{1.166 \times 10^7}$	$\mathbf{1.166 \times 10^7}$	1.157×10^7
kroA100	1.007	1.156	$\mathbf{5.897 \times 10^6}$	5.848×10^6	5.850×10^6	$\mathbf{5.897 \times 10^6}$	$\mathbf{5.897 \times 10^6}$	$\mathbf{5.897 \times 10^6}$
a280	1.018	1.138	3.209×10^6	2.447×10^6	3.151×10^6	$\mathbf{3.21 \times 10^6}$	$\mathbf{3.21 \times 10^6}$	2.970×10^6

Table 1.5: Optimality gap, LP-gap, and other algorithms for some graphs of the Network repository graph database [141] for the max cut problem.

Graph	Optimality gap	LP Gap	LP cut value	Greedy	Trevisan	Simple	Sweep	GW
ENZYMES8	1.034	1.269	1.230×10^2	1.170×10^2	$\mathbf{1.260 \times 10^2}$	$\mathbf{1.260 \times 10^2}$	$\mathbf{1.260 \times 10^2}$	$\mathbf{1.260 \times 10^2}$
eco-stmarks	1.095	1.393	$\mathbf{1.756 \times 10^3}$	8.891×10^2	1.190×10^3	9.354×10^2	9.354×10^2	9.601×10^2
johnson16-2-4	1.000	1.000	3.012×10^3	$\mathbf{3.036 \times 10^3}$	$\mathbf{3.036 \times 10^3}$	2.958×10^3	2.986×10^3	2.918×10^3 116
hamming6-2	1.000	1.000	$\mathbf{9.920 \times 10^2}$	$\mathbf{9.920 \times 10^2}$	$\mathbf{9.920 \times 10^2}$	9.680×10^2	9.690	9.760×10^2
ia-infect-hyper	1.020	1.081	$\mathbf{1.254 \times 10^3}$	1.213×10^3	1.233×10^3	1.227×10^3	1.227	1.211×10^3
soc-dolphins	1.090	1.279	1.160×10^2	1.120×10^2	1.120×10^2	1.190×10^2	$\mathbf{1.210 \times 10^2}$	1.150×10^2
email-enron-only	1.113	1.279	4.060×10^2	3.920×10^2	$\mathbf{4.130 \times 10^2}$	3.710×10^2	3.800×10^2	3.960×10^2
dwt_209	1.054	1.176	$\mathbf{5.410 \times 10^2}$	5.250×10^2	5.270×10^2	5.250×10^2	5.270	5400×10^2
inf-USAir97	1.332	1.683	1.011×10^2	9.961×10^1	9.820×10^1	8.184×10^1	9.337×10^1	$\mathbf{1.074 \times 10^2}$
ca-netscience	1.180	1.334	5.750×10^2	5.830×10^2	5.880×10^2	5.270×10^2	5.270×10^2	$\mathbf{6.110 \times 10^2}$
ia-infect-dublin	1.110	1.247	$\mathbf{1.673 \times 10^3}$	1.648×10^3	1.659×10^3	1.550×10^3	1.558×10^3	1.664×10^3
road-chesapeake	1.106	1.313	$\mathbf{1.250 \times 10^2}$	1.230×10^2	1.230×10^2	1.210×10^2	1.230×10^2	$\mathbf{1.250 \times 10^2}$
Erdős991	1.294	1.560	$\mathbf{9.610 \times 10^2}$	9.330×10^2	9.340×10^2	7.350×10^2	7.580×10^2	9.240×10^2
dwt_503	1.049	1.174	1.805×10^3	1.822×10^3	1.822×10^3	$\mathbf{1.921 \times 10^3}$	$\mathbf{1.921 \times 10^3}$	1.909×10^3

this paper, the author propose a set of corrected formulas for the the sparse pca which reduce to the usual explained variance formula when the PCs are orthogonal.

Synthetic experiments and Pit props data set

To evaluate the recovery of sparse principal components with their semidefinite relaxation, [41] use their program on a synthetic data set and on the Pit pros data set. In this subsection, we compare our linear relaxation $LSPCA$ to their semidefinite program by checking the sparse components that both methods produce. D'Aspermont et al. [41] generate a synthetic matrix C with sparse components and empirically check that their proposed SDP can indeed recover the components. We repeat this experiment and show that the linear relaxation $LSPCA$ obtained by setting $\mathcal{S} = \mathcal{E}(C)$ recovers as well the components. We show the results in Table 1.6. In the artificial example, three hidden factors are created:

$$V_1 \sim \mathcal{N}(0, 290),\ V_2 \sim \mathcal{N}(0, 300),\ V_3 = -0.3V_1 + 0.925V_2 + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, 300)$$

with V_1, V_2 and ε independent. Then, 10 observed variables are generated as follows:

$$X_i = V_j + \varepsilon_i^j,\ \varepsilon_i^j \sim \mathcal{N}(0, 1),$$

with $j = 1$ for $i = 1, 2, 3, 4$, $j = 2$ for $i = 5, 6, 7, 8$ and $j = 3$ for $i = 9, 10$ and $\{\varepsilon_i^j\}$ independent for all $i \in [10]$ and $j \in [3]$. To recover the sparse components, a solution X_1 for program $LSPCA$ is found and truncated to keep only the dominant -sparse-eigenvector x_1. Then, the covariance matrix C is deflated to obtain

$$C_2 = C - (x_1^\top C x_1)x_1 x_1^\top$$

and iterated to obtain further components. As mentioned in [41], the ideal solution is to use (X_1, X_2, X_3, X_4) for the first principal component to recover factor V_1 and only (X_5, X_6, X_7, X_8) for the second component to recover V_2. We replicate the results of Table 1 in [41] using the true covariance matrix C and the oracle knowledge that the sparcity $k = 4$. We then run our linear relaxation by setting $\mathcal{S} = \mathcal{E}(C)$. We report the results in Table 1.6. Observe that the components that our linear relaxation are sparse, and have the same support that the ones found by the SDP and are very close in norm (ignoring the signs, which are irrelevant to this application).

Table 1.6: Loadings for the first two principal components on the synthetic data set with $k = 4$ for both PCs.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
SPCA PC1	0	0	0	0	0.5	0.5	0.5	0.5	0	0
SPCA PC2	0.5	0.5	0.5	0.5	0	0	0	0	0	0
LSPCA PC1	0	0	0	0	−0.598	−0.596	−0.457	−0.28	0	0
LSPCA PC2	0.482	0.366	0.762	0.226	0	0	0	0	0	0

Pit props dataset

We next consider the Pit pros dataset, introduced in [82]. This dataset consists of 180 observations of 13 measured variables. It is a regularly used dataset in the PCA literature, and is notorious for having hard- to-interpret principal components. We replicate the results of Table 2 in [41], where they present two sets of experiments. First, they set $k = 5$ for the first component and then $k = 2$ for components 2 and 3. In the second set of experiments, they set $k = 6$ for the first component and then $k = 2$ for components 2 and 3. For our linear relaxation, We let $\mathcal{S} = \mathcal{E}(C)$ and use the same values of k. We report the results in Table 1.7 for $k = 5, 2, 2$. These two tables show that our methods does recover sparse components.

Variance explained

In this subsection we evaluate the quality of our linear relaxation in terms of the variance explained by the recovered principal components, which is the typical metric to evaluate the quality of principal components. To compute the explained variance, we use the "The fraction of total variance computed" as defined in [31], which is a corrected formula for explained variance when the components are not orthogonal. We compute this value for both the SDP relaxation and our LP relaxation for 40 data sets contained in the *Rdataset* repository, which contains the data sets preinstalled in the core of *R* as well of some of the data sets contained in some of the most popular *R* libraries. We use only data sets which contain between 8 and 20 continuous variables. For each data set, we compute 4 sparse components. We define $ev(SDP)$ to be the largest explained variance by sparse components found by the SDP method by varying the target sparsity k ranging from 1 to 30% of the number of variables in the data set. We define $ev(LP)$ similarly but this time using the linear relaxation. We point out that the number k for which these maximal values are obtained need not be the same. We present our results in figure 1.14. Each point corresponds to the relative error (in percentages) between the explained variances for the two methods, computed as

Table 1.7: Loadings for first three principal components, for the Pit props dataset, $k = 5, 2, 2$.

	topdiam	length	moist	testsg	ovensg	ringtop	ringbud	bowmax	bowdist	whorls	clear	knots	diaknot
SPCA PC1	0.56	0.583	0	0	0	0	0.263	0.098	0.371	0.362	0	0	0
SPCA PC2	0	0	0.707	0.707	0	0	0	0	0	0	0	0	0
SPCA PC3	0	0	0	0	0	0.793	0.610	0	0	0	0	0	−0.012
LSPCA PC1	0.645	0.437	0	0	0	0.019	0.268	0	0.352	0.443	0	0	0
LSPCA PC2	0	0	0.640	0.767	0	0	0	0	0	0	0	0	0
LSPCA PC2	0	0	0	0.0464	0	0.931	0.359	0	0	0	0	0	0

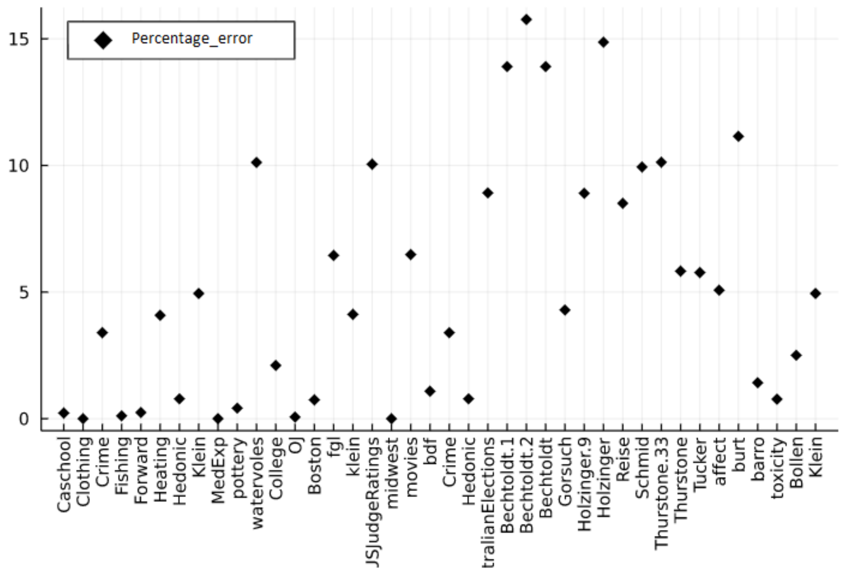


Figure 1.14: Relative error in percentages between the explained variances for the two SDP method and the LP method to recover sparse components.

$$100 \cdot \frac{ev(SDP) - ev(LP)}{ev(SDP)}.$$

We note that among the 40 data sets used, only 1 has an error larger than 15%, all but 6 have an error larger than 10% and more than half (23 out of 40) have an error of less than 5%.

1.7 Summary and future work

In this work, we introduced a generic technique to obtain linear and second order cone relaxations of semidefinite programs with provable guarantees based on the commutativity of the constraints and objective matrices. We believe that other algebraic properties of these matrices can be exploited to obtain further stronger relaxations. Although we believe solving semidefinite programs with linear programs is an interesting topic is its own right, we posit that our ideas can be exploited in settings where linear approximations of convex regions is an essential component of state-of-the-art algorithms, such as in copositive programming [29] and outer approximation algorithms for semidefinite integer programs [102].

On the theoretical side, the main remaining question regarding the max cut problem is if the proposed linear program SP_S provides a better-than-2 approximation algorithm. From our computational tests, we are not aware of any instance where the approximation factor is worse than 1.8.

For the Lovász theta number, the main theoretical question is if our proposed linear program satisfies the same inequalities that $\vartheta(\bar{G})$ does. Namely,

$$\alpha(G) \leq \vartheta(\bar{G}) \leq \chi(G)$$

where $\alpha(G)$ and $\chi(G)$ are the clique and chromatic numbers of G, respectively. It would be interesting as well to find out if program Tn satisfies the bound (1.8) for $d-$regular graphs. Finally, the second order cone relaxations for the knapsack and extended trust region problems performed well in terms of both solving time and objective value. It would be then worthwhile to explore the specialization of Algorithm 1 to these problems, and to compare its behaviour to state of the art algorithms for those problems.

Chapter 2

SPECTRAL OUTER APPROXIMATION ALGORITHMS FOR BINARY SEMIDEFINITE PROBLEMS

2.1 Introduction

In recent years, integer semidefinite optimization has received wide interest from the optimization community. These problems are appealing as they allow the formulation of many mixed integer non-linear problems due to the high expressive power of the semidefinite cone, in particular being able to capture non-differentiable functions. In addition, it allows the design of algorithms based on conic convex optimization, such as Branch-and-Bound approaches which in turn benefit from the advances in stable interior-point algorithms. This contrasts the fact that global solvers for mixed-integer non-linear problems are often unstable and are not very well suited for non-differentiable functions [48].

Even in the presence of these benefits, applications of integer semidefinite programming seem to be quite sparse in the literature. [1] Some examples of mixed-integer semidefinite optimization problems appear in truss topology optimization [62], sparse principal component analysis [97] and the computation of restricted isometry constants [61].

Very recently, exploiting results on positive semidefinite matrices with entries in $\{0, -1, 1\}$, de Meijer and Sotirov showed in [110] that binary, quadratically constrained quadratic problem can be reformulated as binary semidefinite programs (BSDPs). The former are problems of the form

$$\begin{aligned} \min_x \ & x^\top C x + 2d_0^\top x \\ \text{s.t: } & x^\top A_i x + 2d_i^\top x \leq b_i, \ \forall i \in [r], \\ & Dx = t, \\ & x \in \{0,1\}^n, \end{aligned} \tag{BQCQP}$$

with $C, A_i \in \mathbb{S}^n$ for $i \in \{1, \ldots r\}$, $D \in \mathbb{R}^{q \times n}$, $q \in \mathbb{N}$, $d_i \in \mathbb{R}^n$ for $i \in \{0, \ldots r\}$ for some $r \in \mathbb{N}$ and $t \in \mathbb{R}^q$.

[1] As a matter of fact, SDPs appeared historically as relaxations of a quadratic problem (some with binary constraints) through Shor's relaxation, and therefore imposing integer constraints on them is, at the very least, unexpected.

Binary QCQPs are a central class of optimization programs and substantial efforts have been dedicated to understand them theoretically and solve them in practice. Their relevance comes from the fact that they capture a wide array of problems. As mentioned in the introduction, they are able to express problems from many different fields, such as combinatorial optimization and computer science [23, 48, 52, 100, 131], machine learning [57, 107, 127], chemical engineering [21] and the references therein and portfolio optimization [22, 43, 139], among others. Unconstrained binary quadratic problems [14, 112, 134] are a subset of Binary QCQPs that have been extensively studied, in particular from the quantum optimization community [47, 53, 68, 83, 109], mainly due to the development of quantum/quantum-inspired methods to solve them [28].

The results of [110] are motivating as they open a new avenue to solve binary QCQPs to global optimality, a task that remains challenging even for state-of-the-art algorithms. In fact, solving problems of the form BQCQP is NP-hard [130], and they are even hard to approximate. For illustration, the problem of finding the largest independent set in a given graph G on n vertices can be cast as a binary QCQP, but the stable set number in an $n-$node graph is NP-hard to compute and even hard to approximate within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ [70].

Different approaches have been proposed to solve BQCQPs to global optimality. These algorithms usually rely on the *Branch-and-bound* framework. See [10, 25, 95, 118]. Relevant variants are *Branch-and-cut* methods that incorporate cutting planes [85] that tighten the problems arising from branching [116, 126]. A different family of exact algorithms that do not rely on the branch-and-bound framework are *outer approximation algorithms*, first introduced by Duran and Grossmann [49] and Leyffer [56, 96].

Many off-the-shelf solvers exist to solve different variants of QCQP problems. BARON [143], GloMIQO [115], Ipopt [154], Couenne [81] provide global methods for mixed-integer QCQPs. On the other hand, Gurobi [125], SCIP [2], CPLEX [81], Mosek [8] provide algorithms to solve mixed-integer quadratic problems, with some support for non-convex quadratic constraints. We point out that all of these algorithms take time exponential in the size of the optimization problem in the worst case, and unfortunately, they typically exhibit this behavior in practice.

It is not clear, however, if formulating the BQCQPs as binary SDPs and solving them with specialized algorithms for the latter is a viable alternative. In fact, earlier experiments show that the current available solvers for BSDPs are only capable of solving instances of BQCQP of very limited sizes in reasonable time. We are aware of 7 such algorithms. SCIP-SDP [62] uses a branch-and-bound strategy where strict duality of the semidefinite relaxations is inherited to the subproblems. Kobayashi and Takano [87] propose a cutting plane and branch and cut algorithm for generic mixed-integer semidefinite programs. This approach is refined in [111] where Chvatal-Gomory cuts are considered. A very similar algorithm to their cutting plane approach called CUTSDP is implemented in YALMIP [90, 99]. GravitySDP can solve integer SDPS while exploiting sparsity [75]. Finally, in [103], Lubin et al. propose an outer approximation algorithm based on the ideas of Duran and Grossmann [49] and Leyffer [56, 96], to solve mixed integer, conic optimization programs, which therefore can be specialized to mixed integer semidefinite optimization. The resulting implementation in Julia is called *Pajarito*. This algorithm is extended in [38] to a Branch-and-Bound Outer Approximation approach.

These considerations suggest the need to implement algorithms that exploit the specific structure of the integer semidefinite formulation of BQCQP. In this chapter, we exploit this special structure to propose a second order, spectral outer approximation algorithm to solve binary semidefinite programs that are exact formulations of binary QCQPs. To the best of our knowledge, this approach has yet to be tested and compared to the alternatives available to solve binary integer semidefinite programs.

Overview and outline

Our starting point are the results of de Meijer and Sotirov, who showed in [110] that binary QCQPs can be reformulated as binary semidefinite programs. Formally:

Theorem 2.1 (Theorem 9 of [110]). *Let $C, A_i \in \mathbb{S}^n, d_0, d_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $\forall i \in [r]$ and $D \in \mathbb{R}^{q\times n}, t_i \in \mathbb{R}, \forall\ i \in [q]$, where $r, q \in \mathbb{N}$. The following semidefinite, binary program is equivalent to BQCQP.*

$$\begin{aligned} \min_{X,x}\ & \langle C, X\rangle + 2d_0^\top x \\ s.t{:}\ & \langle A_i, X\rangle + d_i^\top x \le b_i \ \ i \in [r], \\ & Dx = t, \\ & X - xx^\top \succeq 0, \\ & Diag(X) = x,\ x \in \{0,1\}^n. \end{aligned} \tag{BSDP}$$

The constraint $X - xx^\top \succeq 0$ is usually written as the equivalent constraint $\begin{bmatrix} X & x^\top \\ x & 1 \end{bmatrix} \succeq$

0. Our main idea is to use an outer approximation approach to tackle this formulation. This family of algorithms work by iteratively approximating the semidefinite feasible region with polyhedrons and solving the resulting integer linear problems. Now, the performance of these algorithms depends crucially on the quality of the outer approximation. We propose to approximate the region defined by $X - xx^\top \succeq 0$ by using a set of vectors $v_1, \ldots, v_n$ where $n \in \mathbb{N}$ by the region defined by the second order cone constraints $v_1^\top(X - xx^\top)v_1 \geq 0, \ldots, v_n^\top(X - xx^\top)v_n \geq 0$. By drawing inspiration from the ideas of Chapter 1, we propose to carefully select the vectors v as the columns of a matrix that simultaneously diagonalizes the matrix C and an aggregation of matrices $A_i, i \in [r]$ from the BSDP problem. The rest of this chapter is organized as follows.

(a) In Section 2.2, we present the outer approximation algorithm for mixed-integer conic programs of [103] specialized to mixed integer semidefinite programs.

(b) In Section 2.3, we develop two spectral outer approximation algorithms for integer, semidefinite programs based on a second-order approximation of the semidefinite program's feasible region, which explicitly uses information from the objective and constraints matrices.

(c) In Section 2.4 we introduce the binary, cardinality constrained least squares problem and its formulation as a binary SDP. We then present the Quadratic knapsack problem and its formulation as a binary SDP.

(d) In Section 2.5, we briefly describe available algorithms to solve binary semidefinite optimization problems and compare them to the proposed algorithms in Section 2.3.

(e) In Section 2.6 we conclude with some remarks.

2.2 Outer approximation algorithms for Integer semidefinite problems

In this section, we introduce an outer approximation algorithm for integer semidefinite optimization. This algorithm is introduced in the more general setting of mixed-integer conic optimization [103] by Lubin et al.

We begin by introducing integer semidefinite programs. Such a problem is of the form

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \ & \langle C, X \rangle \\ \text{s.t: } & \langle A_i, X \rangle = b_i \ \forall i \ \in [r], \\ & X \succeq 0 \\ u_{ij} \leq X_{ij} \leq l_{ij}, \ & X_{ij} \in \mathbb{Z}, \ \ (i,j) \in L \subseteq [n] \times [n], \end{aligned} \tag{ISDP}$$

with $C, A_i \in \mathbb{S}^n$, $b_i \in \mathbb{R}$ for all $i \in [r]$. $L \subseteq [n] \times [n]$ indicates the entries of X which are constrained to be integer, and u_{ij}, l_{ij} upper and lower bounds of X_{ij} for $(i, j) \in L$. Notice that if we ignore the integrality constraints, the set of feasible solutions is convex and given by the equations $X \succeq 0$, $\langle A_i, X \rangle = b_i \ \forall i \ \in [r]$. A polyhedral outer approximation of the set of positive semidefinite matrices $\mathbb{S}^n$ is defined as follows.

Definition 2.1. *A polyhedron P is an outer approximation of $\mathbb{S}^n$ if P equals the intersection of a finite number of half-spaces and contains $\mathbb{S}^n$.*

Outer approximations of the semidefinite cone are obtained by fixing a finite set of positive semidefinite matrices since $\mathbb{S}^n_+$ is self-dual. That is, a polyhedral set $P = \{X \in \mathbb{S}^n : \langle T_i, X \rangle \geq 0, \ i \in [q]\}$, where $T_1, \ldots, T_q \in \mathbb{S}^n_+$ is an outer approximation of $\{X \in \mathbb{S}^n : X \succeq 0\}$. Because we will need to keep track of the matrices T_i, we set $\mathcal{T} := \{T_1, \ldots, T_q\}$ and let

$$P_{\mathcal{T}} := \{X \in \mathbb{S}^n : \langle T, X \rangle \geq 0 \ \forall \ T \in \mathcal{T}\}.$$

Consider a problem of the form of ISDP. Let $P_{\mathcal{T}}$ be a polyhedral outer approximation of $\mathbb{S}^n_+$. Then, the problem

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \ & \langle C, X \rangle \\ \text{s.t: } & \langle A_i, X \rangle = b_i \ \forall i \ \in [r], \\ & X \in P_{\mathcal{T}}, \\ s_{ij} \leq X_{ij} \leq l_{ij}, \ & X_{ij} \in \mathbb{Z}, \ \ (i,j) \in L \subseteq [n] \times [n]. \end{aligned} \tag{$OASDP(P_{\mathcal{T}})$}$$

is called the linear outer approximation problem of ISDP. Notice that this problem is a mixed-integer linear problem. Since any feasible solution to ISDP is feasible to $OASDP(P_{\mathcal{T}})$ as well, this latter problem is a relaxation of the former, and the optimal value of $OASDP(P_{\mathcal{T}})$ provides a lower bound to ISDP.

Notice that if we fix the entries of X to have a specific integer value in program ISDP, the program reduces to a semidefinite optimization problem. More formally, Let X^L

be a matrix with $X^L_{ij} \in \mathbb{Z}$ and $u_{ij} \leq X^L_{ij} \leq l_{ij}$ for all $(i, j) \in L \subseteq [n] \times [n]$. Consider the following optimization program

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \ & \langle C, X \rangle \\ \text{s.t: } & \langle A_i, X \rangle = b_i \ \forall i \ \in [r], \\ & X \succeq 0 \\ & X_{ij} = X^L_{ij} \ \ \forall (i, j) \in L. \end{aligned} \tag{$SDP(X^L)$}$$

Since we have fixed the integer coordinates to take a specific value, this program is a (convex) positive semidefinite optimization problem. Furthermore, if X' is an optimal solution to this problem, it is feasible for $ISDP$, and therefore provides a corresponding upper bound.

In summary, the outer approximation algorithm for ISDP first computes an optimal solution $\hat{X}$ to $OASDP(P_{\mathcal{T}})$. This solution necessarily has integer values in the entries specified by L, and the algorithm proceeds to solve problem $SDP(\hat{X})$. That is, problem $SDP(X^L)$ where the entries in L match the entries of $\hat{X}$ in L. If all problems are solvable and X^* denotes an optimal solution of ISDP then the following key inequalities hold

$$\left\langle C, \hat{X} \right\rangle \leq \langle C, X^* \rangle \leq \langle C, X' \rangle .$$

If the values $\left\langle C, \hat{X} \right\rangle$ and $\langle C, X' \rangle$ are equal, then X' is optimal for ISDP and the algorithm terminates. If not, the algorithm proceeds by updating the outer approximation $P_{\mathcal{T}}$ to a tighter approximation by the addition of valid linear constraints, and $OASDP(P_{\mathcal{T}})$ is re-solved. This process yields a non-decreasing sequence of lower bounds.

We now formally state the linear outer approximation algorithm.

Algorithm 2 OA(SDP)

Fix a tolerance $\varepsilon > 0$. Set $\mathcal{T} = \{(e_i \pm e_j)(e_i \pm e_j)^\top : i, j \in [n]\}$.
Solve problem OASDP($P_{\mathcal{T}}$) finding a minimizer $\hat{X}$.
Solve problem $SDP(\hat{X})$, finding a minimizer X'.
if $\left| \langle C, X' \rangle - \left\langle C, \hat{X} \right\rangle \right| > \varepsilon$ **then**
    Find a dual optimal dual solution S' of program $DSDP(\hat{X})$.
    Set $\mathcal{T} = \mathcal{T} \cup \{S'\}$. Go to step 2.
end if
return X'.

Notice that since the number of possible integer assignments for the coordinates in L are finite, this algorithm is guaranteed to terminate in a finite number of steps if there is no repetition of assignments of the integer variables. In what follows, we prove a lemma that guarantees finite convergence of the algorithm. To state this result, we need first to specify the dual of problem $SDP(X^L)$.

Observation 2.1. *The dual of problem $SDP(X^L)$ is given by*

$$\begin{aligned} \max_{\gamma\in\mathbb{S}^n, S\in\mathbb{S}^n_+, y\in\mathbb{R}^n} \quad & b^\top y + \left\langle \gamma, X^L \right\rangle \\ s.t: \gamma_{i,j} = 0\ \forall (i,j) &\in [n]\times[n] \setminus I \\ C - \sum_{i=1}^{r} A_i y_i - \gamma &= S \\ S &\geq 0. \end{aligned} \tag{$DSDP(X^L)$}$$

The key to obtain appropriate hyperplanes to update the set $P_{\mathcal{T}}$ that imply the convergence of the algorithm in finite time is conic *duality*.

Lemma 2.1. *Let X^L be fixed and denote by Z_{X^L} the optimal value of program ($SDP(X^L)$) with corresponding minimizer X'. Suppose that strong duality holds between the pair of problems $DSDP(X^L)$ and $SDP(X^L)$. Let S' be optimal for the latter program. Set $\mathcal{T} = \{S'\}$ so that*

$$P_{\mathcal{T}} = \{X \in \mathbb{S}^n : \langle X, S' \rangle \geq 0\}.$$

Let $\hat{X}$ be such that $\langle A_i, \hat{X}\rangle = b_i\ \forall i \in [m]$, $\langle \hat{X}, S'\rangle \geq 0$ and such that $\hat{X}_{ij} = X^L_{ij}$ for all $(i,j) \in L$. Then, $\langle C, \hat{X}\rangle \geq Z_{X^L}$. In addition, if $\hat{X}$ is optimal for program $OASDP(P_{\mathcal{T}})$ - or in other words, if the outer approximation $OASDP(P_{\mathcal{T}})$ returns a matrix with integer part equal to X^L-, then X' is global optimal for ISDP and the outer approximation algorithm terminates.

Proof. First observe that

$$
\begin{aligned}
0 \leq \langle \hat{X}, S' \rangle &= \langle \hat{X}, C - \sum_{i=1}^{r} A_i y_i - \gamma \rangle \\
&= \langle \hat{X}, C \rangle - \langle \hat{X}, \sum_{i=1}^{r} A_i y_i - \gamma \rangle \\
&= \langle \hat{X}, C \rangle - \sum_{i=1}^{r} y_i \langle \hat{X}, A_i \rangle - \langle \hat{X}, \gamma \rangle \\
&= \langle \hat{X}, C \rangle - \sum_{i=1}^{r} y_i b_i - \langle \hat{X}, \gamma \rangle \\
&= \langle \hat{X}, C \rangle - \sum_{i=1}^{r} y_i b_i - \langle X^L, \gamma \rangle = \langle \hat{X}, C \rangle - Z_{X^L}.
\end{aligned}
$$

The last equation is valid because γ is zero for the (i, j) entries not in L, and because $\hat{X}$ matches X^L in those entries. Hence, we derive $\langle C, \hat{X} \rangle \geq Z_{X^L}$.

To conclude, let OPT denotes the optimal value of ISDP. Observe that since program $OASDP(P_{\mathcal{T}})$ is a relaxation of ISDP we have $\langle C, \hat{X} \rangle \leq OPT$. Now, Z_{X^L} is the optimal value of program $SDP(X^L)$ whose optimizer is feasible to program ISDP so that we have $OPT \leq Z_{X^L}$. All in all we get the inequalities

$$\langle C, \hat{X} \rangle \leq OPT \leq Z_{X^L} \leq \langle C, \hat{X} \rangle.$$

Hence, we get that $Z_{X^L} = OPT$, the gap with outer approximation relaxation is 0 and X' is optimal for ISDP.

□

The main consequence of this lemma is that the outer approximation algorithm will not cycle through integer solutions. Indeed, if an integer solution is repeated in program $SDP(X^L)$, then the algorithm will terminate in the next step by proving a gap of 0 between the inner and outer approximations.

We make a few remarks on this algorithm. Altough it is guaranteed to terminate under mild assumptions, in the worst case it might need to solve an exponential number of subproblems. Second, the *initialization* of $\mathcal{T}$ to the set $\{(e_i \pm e_j)(e_i \pm e_j)^\top : i, j \in [n]\}$ amounts to the linear constraints on X given by

$$X_{ii} + X_{jj} \geq 2|X_{i,j}| \; \forall i, j \in [n].$$

which are necessary for positive semidefinitness.

2.3 Refining outer approximations

The efficiency of the outer approximation algorithm for integer, semidefinite programs depends on two main factors: How fast we can solve each integer sub-problem, and the quality of the outer polyhedral approximation. Recently, commercial solvers such as Gurobi have had success solving mixed integer linear and second-order mixed integer problems, suggesting that improvements to outer approximation algorithms are more likely to come from the polyhedral approximation side. In Chapter 1, we considered exactly the problem of finding “good" polyhedral approximations of the feasible set of a semidefinite optimization program by using spectral information of the objective coefficient matrix and the matrices determining the constraints of the SDP. Here “good" is to be understood as that the objective value of the linear problem resulting from constraining the variables to belong to the polyhedral approximation rather than the semidefinite cone is close to that of the original problem. The aim of this section is to develop an algorithm suitable to binary semidefinite programs of the form of BSDP, which is on variables x and X. Since the ambient space of the matrix $\begin{bmatrix} X & x^\top \\ x & 1 \end{bmatrix}$ is $\mathbb{R}^{n+1}$ and the matrices defining the objective C and the constraints $A_i,\ i \in [r]$ are in $\mathbb{R}^n$, we rather work with the constraint $X - xx^\top \succeq 0$ which is equivalent to $\begin{bmatrix} X & x^\top \\ x & 1 \end{bmatrix} \succeq 0$ in order to apply the ideas of Chapter 1. To explain the details, we first discuss second-order strengthenings of polyhedral outer approximations.

Second-order strengthening

We begin by recalling that the second-order cone $\mathcal{L}^{1+n}$ is given by

$$\mathcal{L} = \{(r,t) : \mathbb{R}^{1+n} : r \geq \|t\|_2^2\}.$$

In the particular case of an integer semidefinite optimization problem, second order necessary conditions for positive semidefinitness can be imposed, resulting in second-order integer problems for the outer approximation step in Algorithm 2, rather than integer linear problems. [38] takes this idea further, and proposes a version of *Pajarito* using a second-order cone outer approximation for the outer approximation step. First, we recall that the *rotated second order cone* $\mathcal{V}^{2+n}$ is given by

$$\mathcal{V}^{2+n} = (r,s,t) \in \mathbb{R}^{2+n} : r,s \geq\ 0, 2rs \geq \|t\|_2^2.$$

This cone is self dual, and can be obtained as an invertible linear transformation of the standard second-order cone $\mathcal{L}$ as $(r,s,t) \in \mathcal{V}^{n+2}$ if and only if $(r+s, r-s, \sqrt{2}t_1, \ldots, \sqrt{2}t_n) \in \mathcal{L}^{2+n}$. One can check that given $X \succeq 0$ the *rotated second order cone constraints*

$$(X_{ii}, X_{jj}, \sqrt{2}X_{ij}) \in \mathcal{V}^3$$

are valid for each i and j. Indeed, this corresponds to saying that the 2 by 2 minors of X are positive semidefinite. Equivalent cuts are also described in [18] where the authors mention that if X is positive semidefinite, then X satisfies

$$\left\| \begin{pmatrix} 2X_{i,j} \\ X_{i,i} - X_{j,j} \end{pmatrix} \right\|_2 \leq X_{i,i} + X_{j,j}, \ \forall i \in [n], \ \forall j \in [n]. \tag{2.1}$$

These cuts are also mentioned in [157] and in fact all of them can be derived using an alternative version of the Schur Complement Lemma presented in [86]. We briefly mention this idea in Subsection 2.3. However, it is not clear that enforcing the PSD constraints on 2 by 2 minors is necessarily beneficial. Although they provide a tighter approximation, the additional $\frac{n^2-n}{2}$ cuts added place a heavy burden on the integer, second-order solver. Therefore, if second order cone cuts are to be added, they must be few and significantly improve quality of the approximation. In what follows, we describe a derivation of such cuts, using the specific structure that binary SDPs arising from binary QCQPs have. Indeed, in Chapter 1, we considered *quadratically constrained quadratic problems* of the form

$$\begin{aligned} &\min_x x^\top C x + 2d_0^\top x \\ &\text{s.t: } x^\top A_i x + 2d_i^\top x \leq b_i, \ \forall i \in [r] \end{aligned} \tag{2.2}$$

and their corresponding Shor semidefinite relaxation

$$\begin{aligned} &\inf_{x \in \mathbb{R}^n, X \in \mathbb{S}^n} \langle C, X \rangle + d_0^\top x \\ &\text{s.t: } \langle A_i, X \rangle + d_i^\top x \leq \ b_i \ \forall i \in [r], \\ &\qquad X - xx^\top \succeq 0. \end{aligned} \tag{2.3}$$

By fixing a finite set $\mathcal{S} \subseteq \mathbb{R}^n$, this latter problem can be relaxed to the second order cone program

$$\begin{aligned}
&\inf_{x\in\mathbb{R}^n, X\in\mathbb{S}^n} \langle C, X\rangle + d_0^\top x\\
&\text{s.t: } \langle A_i, X\rangle + d_i^\top x \leq\ b_i\ \forall i \in [r],\\
&\quad v^\top \left(X - xx^\top\right) v \geq 0\ \forall\ v \in \mathcal{S}.
\end{aligned} \tag{2.4}$$

Notice that the cuts $v^\top \left(X - xx^\top\right) v \geq 0$ are tailored to the special structure of program BSDP, and are second-order cuts, resulting in a much tighter approximation of the convex region $X - xx^\top \succeq 0$ than the polyhedral alternative.

In the experimental section of Chapter 1, we showed that the second-order cone relaxation (2.4) where $\mathcal{S}$ is a basis of eigenvectors of C is within 1% of the objective value of the semidefinite program for the quadratic knapsack problem and certain extensions of the extended trust region problems.[2] In addition, we tested the methodology on the semidefinite optimization problems for max cut and the Lovász theta number and presented results that indicate that the objective of the relaxations is very close to that of the respective SDPs. Second-order strenghening can also be derived for these problems, resulting in provably stronger relaxations.

We now propose a second order cone outer approximation algorithm to solve problem BSDP. In this setting, the ambient space of the positive semidefinite matrices considered is $\mathbb{S}^{n+1}$ since our variable matrix is $\begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix} \succeq 0$. Let $\mathcal{S}$ be a finite subset of $\mathbb{R}^n$. Let $\mathcal{T} = \{T_1, \ldots, T_q\} \subseteq \mathbb{S}^{n+1}_+$ and define $P_{\mathcal{T}} := \{X \in \mathbb{S}^{n+1} : \langle X, T\rangle \geq 0\ \forall T \in \mathcal{T}\}$. Consider the second-order relaxation of BSDP given by

$$\begin{aligned}
&\min_{X,x}\ \langle C, X\rangle + 2d_0^\top x\\
&\text{s.t: } \langle A_i, X\rangle + d_i^\top x \leq b_i\ \ \forall\ i \in [r],\\
&\qquad Dx = t,\\
&\qquad Diag(X) = x,\ x \in \{0,1\}^n,\\
&\qquad v^\top \left(X - xx^\top\right) v \geq 0\ \forall v \in \mathcal{S}\\
&\qquad \begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix} \in P_{\mathcal{T}}.
\end{aligned} \tag{$SOC(\mathcal{S},\mathcal{T})$}$$

With this program at hand, we can introduce the *spectral second order outer approximation algorithm*.

[2]For the quadratic knapsack problem, there are no quadratic constraints, and therefore Theorem 1.3 suggests we simply take $\mathcal{S}$ to be a basis of orthonormal eigenvectors of C.

Algorithm 3 Spectral-second-order-OA

1: Fix a tolerance $\varepsilon > 0$. Find $q^1 \in \mathbb{R}^n$ such that $\sum_{i=1}^r q_i^1 A_i = I$.
2: Use program (CG) to find q^2 with support disjoint from q_1 such that the matrices C and $\sum_{i=1}^r A_i q_i^2$ commute. Let U be a matrix that simultaneously diagonalizes C and $\sum_{i=1}^r A_i q_i^2$. Denote its columns by $v_1, \ldots, v_n$.
3: Set $\mathcal{S} = \{v_1, \ldots, v_n\}$. Set $\mathcal{T} = \emptyset$.
4: Solve problem $SOC(\mathcal{S}, \mathcal{T})$ finding a minimizer $\bar{X}$.
5: Solve problem $SDP(\bar{X})$, finding a minimizer $\hat{X}$.
6: **if** $\left|\left\langle C, \hat{X} \right\rangle - \left\langle C, \bar{X} \right\rangle\right| > \varepsilon$ **then**
7:     Find a dual optimal dual solution $\hat{S}$ of program $DSDP(\bar{X})$.
8:     Set $\mathcal{T} = \mathcal{T} \cup \{\hat{S}\}$. Go to step 4.
9: **end if**
10: **return** $\hat{X}$.

In this algorithm, we update the outer approximation by adding linear constraints coming from a dual optimal solution of $DSDP(\bar{X}^L)$. By Lemma 2.1, this guarantees the termination of the algorithm. However, since we are now dealing with a second order cone integer program, one may think of adding second order cuts to further strengthen the outer approximation. In other words, it might be worthwhile to update the set $\mathcal{S}$. We describe this process in the next subsection.

An interesting variant of this algorithm consists is using so called *lazy constraints*. In fact, notice that Algorithm 3 solves an integer second-order cone program and hence explores a Branch-and-Bound tree at each iteration. Instead, it seems worthwhile to cut solutions that are not positive semidefinite in each of the nodes by imposing constraints of the form $v^\top X v \geq 0$ dynamically. Such dynamic constraint generation is known as lazy constraint callback in the optimization literature. Given an instance of problem $SOC(\mathcal{S}, \mathcal{T})$ with $\mathcal{T} = \emptyset$ and $\mathcal{S} = \{v_1, \ldots, v_n\}$ where the v_i are defined as in Algorithm 3, the variant algorithm maintains a pool of lazy constraints $\mathcal{LC}$ that is initially empty. Then, a Branch-and-Bound procedure is initiated. Whenever an optimal solution $\hat{X}$ of a branch is found, the algorithm tests whether $u^\top X u \geq 0$ where u ranges over the elements $\mathcal{LC}$. If all of these inequalities are satisfied but $\hat{X}$ is not PSD the algorithm computes a eigenvector w corresponding to the least eigenvalue of $\hat{X}$ and updates $\mathcal{LC}$ to $\mathcal{LC} \cup \{w\}$. This procedure is iterated until optimality is proven by the Branch-and-Bound procedure. We refer to this algorithm as the Spectral-Lazy-second-order-B&C.

Cut disaggregation

At a given iteration, the outer approximation algorithm solves problem $SDP(\bar{X})$ and finds an optimal dual solution S' of problem $DSDP(\hat{X})$. If the gap between the objective

Algorithm 4 Spectral-Lazy-second-order-B&C

1: Fix a tolerance $\varepsilon > 0$. Find $q^1 \in \mathbb{R}^n$ such that $\sum_{i=1}^r q_i^1 A_i = I$.
2: Use program (CG) to find q^2 with support disjoint from q_1 such that the matrices C and $\sum_{i=1}^r A_i q_i^2$ commute. Let U be a matrix that simultaneously diagonalizes C and $\sum_{i=1}^r A_i q_i^2$. Denote its columns by $v_1, \ldots, v_n$. Set $\mathcal{S} = \{v_1, \ldots, v_n\}$.
3: Start (or continue) the Branch-and-Bound procedure to solve problem $SOC(\mathcal{S}, \mathcal{T})$.
4: Whenever a feasible solution $\hat{X}$ is found go to 5.
5: if $\lambda_n(\hat{X}) < -\varepsilon$, add the constraint $w^\top X w \geq 0$ where w is a eigenvector corresponding to $\lambda_n(\hat{X})$ to program $SOC(\mathcal{S}, \mathcal{T})$.
6: Go to 3.
7: **return** $\hat{X}$.

of the outer approximation integral program and the inner semidefinite program with fixed integer values exceeds a threshold ε, the algorithm iterates by refining the outer approximation adding the constraints $\langle X, S' \rangle \geq 0$ which guarantees that the outer approximation algorithms terminates in finite time. This strategy can be improved by adding cuts that are implied by the positive semidefinitness of X and that in turn imply $\langle X, S' \rangle \geq 0$. The following desegregation of cuts are suggested in Chapter 1 and in [38].

Observation 2.2. *Let $S \in \mathbb{S}_+^n$ be a positive semidefinite matrix. We have that $S = \sum_{j=1}^n \lambda_j v_j v_j^\top$ where $\lambda_j, j \in [n]$ are the eigenvalues of S' and the vector v_j is a eigenvector of S corresponding to λ_j for each $j \in [n]$. Then,*

$$\langle X, v_j v_j^\top \rangle = v_j^\top X v_j \geq 0 \;\forall\; j \in [n], \textit{ implies } \langle X, S \rangle \geq 0.$$

These constraints are linear in X and therefore can be added to Algorithm 3 in step 8. Perhaps more interestingly, the disaggregation of $S = \sum_{j=1}^n \lambda_j v_j v_j^\top$ can also be used to impose second order cone constraints directly related to the structure of the semidefinite matrix $\begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix}$. Notice the optimal dual variable S^* obtained in step 7 of Algorithm (3) is of dimension $n + 1 \times n + 1$.

Lemma 2.2. *Suppose that the matrix $\begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix} \in \mathbb{S}^{n+1}$ is positive semidefinite, or equivalently $X - xx^\top \succeq 0$. Let $S = \sum_{j=1}^{n+1} \lambda_j v_j v_j^\top \succeq 0$. For each $j \in [n+1]$ denote by z_j the $n+1$st entry of v_j and by $w_j \in \mathbb{R}^n$ the vector v_j restricted to its first n entries. Furthermore, suppose that for each $j \in [n+1]$ the equation*

$$w_j^\top X w_j \geq (w_j^\top x)^2$$

holds. Then, it follows that

$$\left\langle \begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix}, S \right\rangle \geq 0.$$

Proof. Observe that for each $j \in [n+1]$ we have

$$(w_j^\top x + z_j)^2 = (w_j^\top x)^2 + 2w_j^\top x z_j + z_j^2 \geq 0.$$

This implies that $2(w_j^\top x)z_j + z^2 \geq -(w_j^\top x)^2$. Now, we have that $w_j^\top X w_j \geq (w^\top x)^2$ because $X - xx^\top \succeq 0$. Adding the two previous equations yields

$$w_j^\top X w_j + 2(w_j^\top x)z_j + z_j^2 \geq -(w^\top x)^2 + (w^\top x)^2 = 0.$$

To conclude, observe that

$$\left\langle \begin{bmatrix} X & x \\ x^\top & 1 \end{bmatrix}, v_j v_j^\top \right\rangle = w_j^\top X w_j + 2(w_j^\top x)z_j + z_j^2$$

so multiplying by $\lambda_j \geq 0$ (since $S \succeq 0$) and adding over j gives the result.

□

To use the disaggregation of $S' = \sum_{j=1}^{n+1} \lambda_j v_j v_j^\top$ we can add the cuts $w_j^\top X w_j \geq (w_j z_j)^2$ for each $j \in [n+1]$ in step 8 of algorithm 3. By our previous lemma, this ensures that the conditions of Lemma 2.1 are satisfied.

In the implementation proposed in [38], the authors propose an alternate way of generating second-order cuts using the disaggregation of S' based on the ideas of [86], which proposes a set of second-order quadratic constraints implied by positive semidefinitness. We briefly present their idea for comparison. Let $X \in \mathbb{S}^n$ be positive semidefinite. Fix $i \in [n]$ and let $w \in \mathbb{R}^n$ be an arbitrary vector. Let w_i be the $i-th$ entry of w, and $\bar{w} \in \mathbb{R}^{n-1}$ be the vector obtained by removing the $i-th$ entry of the vector w. Let X_{ii} be the $i-th$ diagonal entry of a matrix X.

Let $u \in \mathbb{R}^{n-1}$ be the vector obtained by removing the $i-th$ entry of the $i-th$ row of X, i.e., by removing X_{ii} from the $i-th$ row. Let $\bar{X} \in \mathbb{S}^{n-1}$ be the matrix obtained by removing from X its $i-th$ column and row. In [86], Kim et al. prove that $X \succeq 0$ if $X_{ii} \geq 0$ and:

$$\bar{X} \succeq 0 \;\text{ and } (X_{i,i})\bar{X} - uu^\top \succeq 0.$$

Next, observe that the constraint

$$(X_{ii}, \bar{w}^\top \bar{X}\bar{w}, \sqrt{2}\bar{w}^\top u) \in \mathcal{V}^3 \tag{2.5}$$

is equivalent to

$$w^\top \bar{X} w \geq 0 \text{ and } w^\top (X_{ii}\bar{X}) w \geq (w^\top u)^2.$$

The condition $\bar{X} \succeq 0$ implies $w^\top \bar{X} w \geq 0$ and it is direct to check that $(X_{i,i})\bar{X} - uu^\top \succeq 0$ implies that $\bar{w}^\top (X_{ii}\bar{X})\bar{w} \geq (\bar{w}^\top u)^2$. Hence, the constraint 2.5 is implied by $X \succeq 0$ and can be used to obtain a stronger, second-order cone outer approximation to the positive semidefinite cone. Setting $w = e_i \pm e_j, i > j \in [n]$ results in the cuts $\mathcal{V}^{2+n} = (r, s, t) \in \mathbb{R}^{2+n} : r, s \geq 0, 2rs \geq \|t\|_2^2$ proposed in [38]. To generate cuts using the disaggregation of S', the authors heuristically set i to be the index of the largest absolute entry of a vector v_j, and add the cut (2.5) by setting $w = v_j$.

2.4 Applications to binary semidefinite programs

In this section, we introduce two binary semidefinite problems. In Section 5, we test the different algorithms considered on the problems below.

Cardinality-constrained Boolean least squares

The cardinality-constrained Boolean least squares is a problem of the following form:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} & \|Ax - b\|_2^2 \\ s.t :& x \in \{0,1\}^n \\ & \sum_{i=1}^n x_i \leq k. \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, $k \in \mathbb{N}$. This problem appears in digital communications in the setting of maximum likelihood estimation of digital signals [127]. By writting $x_i^2 = x_i$, this problem can be written as the following QCQP:

$$
\begin{aligned}
&\min_{x\in\mathbb{R}^n} x^\top A^\top Ax - 2b^\top Ax + b^\top b \\
&s.t : x_i^2 = x_i \;\forall\, i \in [n], \\
&\quad\;\; \sum_{i=1}^n x_i \le k.
\end{aligned} \tag{2.6}
$$

In turn, this QCQP can be exactly reformulated as the following BSDP:

$$
\begin{aligned}
\inf_{x\in\mathbb{R}^n, X\in\mathbb{S}^n} \;& \left\langle A^\top A, X\right\rangle - 2b^\top Ax + b^\top b \\
\text{s.t: } & Diag(X) = x, \\
& \sum_{i=1}^n x_i \le k, \\
& X - xx^\top \succeq 0, \\
& x \in \{0,1\}.
\end{aligned} \tag{BLSSDP}
$$

This problem is closely related to the computation of *restricted isometry constants*. These quantities are relevant in the context of compressed sensing and are NP-hard to compute. In [61], Gally and Pfetsch propose a mixed-integer semidefinite program to compute them. [87] uses the same problem to test their proposed algorithms. In contrast to our work, [61, 87] let the x variables take arbitrary real values, while we fix them to be binary.

Quadratic knapsack problem

The *Quadratic Knapsack problem* [131] was introduced in Subsection 1.4. We recall that this BQCQP is of the form

$$
\begin{aligned}
&\max_{x\in\mathbb{R}^n} x^\top Cx \\
&s.t : \sum_{j=1}^k w_j x_j \le c,\; x \in \{0,1\}^n
\end{aligned} \tag{QKP}
$$

where $w \in \mathbb{R}^n$, $C \in \mathbb{S}^n$, $c \in \mathbb{R}_+$.

This problem can be formulated exactly as the following binary semidefinite program

$$
\begin{aligned}
\max_{x\in\mathbb{R}^n, X\in\mathbb{S}^n} \quad & \langle C, X\rangle \\
\text{s.t:} \quad & Diag(X) = x, \\
& \sum_{i=1}^{n} w_i x_i \leq c, \\
& X - xx^\top \succeq 0, \\
& x \in \{0,1\}.
\end{aligned}
\tag{QKPSDP}
$$

2.5 Experimental results

In this section, we present experiments to evaluate the quality of our proposed algorithms, which are designed to tackle the special structure of binary semidefinite programs arising as exact formulations of binary quadratic problems. We compare our algorithms with the majority of mixed-integer semidefinite program solvers that are known to us by means of evaluating the solving time on instances of the problems introduced in Section 2.4. These algorithm are the following.

1. The pure Outer Approximation algorithm[3] proposed by [103] Pajarito, and implemented in the Julia language [19].
2. The Branch-and-Bound-outer-approximation extension of Pajarito proposed in [38], implemented in the Julia language as well.
3. The CUTSDP[4] algorithm implementation in Yalmip [99].
4. The cutting plane algorithm of Kobayashi and Takano [87].
5. The Branch-and-Cut algorithm of Kobayashi and Takano [87].

The pure outer approximation algorithm implemented in Pajarito is similar to Algorithm 3, as the polyhedral outer approximations are updated in the same fashion. The main difference is that the default version of the outer approximation algorithm of Pajarito solves mixed-integer linear problems rather than mixed-integer second order cone problems. The Branch-and-Bound-Outer-Approximation extension of the algorithm proposed in [38] is more sophisticated, and works by maintaining a single branch tree and inner and outer approximating the problems at each node of the tree. CUTSDP

[3] `https://github.com/jump-dev/Pajarito.jl`

[4] `https://yalmip.github.io/solver/cutsdp/`

works by iteratively generating valid constraints for the SDP constraint and then by solving mixed-integer linear problems. If the solution for an integer linear problem is not PSD, an eigenvector corresponding to a negative eigenvalue is computed and added as a constraint in the problem. This algorithm is very similar to the cutting plane of Kobayashi and Takano. The Branch-and-cut algorithm by the same authors is similar to Algorithm 4, but without the initialization steps 1 and 2.

To compare the different algorithms, we implemented Algorithm 3 and 4 in Julia, and used the package Pajarito directly. We implemented the two algorithms of Kobayashi and Takano. Since CUTSDP is essentially the same algorithm as the cutting plane algorithm of these two authors, we do not test the latter algorithm.

The main metric used to compare the algorithms is the time taken to solve the optimization problems. For each combination of parameters (which we describe in the following subsections), we generate 3 random instances. Following Gally [62], we use the *shifted geometric time* to aggregate the times taken by each algorithm to solve the 3 instances. The shifted geometric mean of values $y_1, \ldots, y_n$ is defined as

$$\left(\prod_{j=1}^{n}(y_j + s)\right)^{\frac{1}{n}} - s$$

where the shift s was set to 10. The intention of this aggregation metric is that it reduces the impact of easier instances. Each algorithm was given 30 minutes per instance, and 3 random instances were generated for each configuration of parameters. We present our results in Tables 1 through 3. In these tables, the columns are defined as follows:

- n: Size of the PSD matrix variable.
- k: In Tables 2.2 and 2.1, this denotes the value of k in problem BLSSDP.
- m_b: Number of binary decision variables.
- m_c: Number of continuous decision variables.
- Method: Algorithm used to solve the instances. PAJARITO_OA, PAJARITO_TREE, KOB_1, KOB_2 correspond, in order, to the list at the beginning of this section. LAZY_SOC and PURE_OA_SOC correspond to Algorithms 4 and 3 respectively.
- Time: The shifted geometric mean time in seconds taken to solve the instances.

#Abort: The number of instances that were aborted either due to numerical or memory issues, or if the Gurobi solver reported that there are no feasible solutions.

#Limit: The number of instances that exceeded the computation time limit of 30 minutes.

To solve the optimization sub-problems we have used Mosek [8] for semidefinite programs and Gurobi [125] for the mixed integer programs. All of the code used is available at `https://github.com/dderoux`.[5]

Cardinality-constrained Boolean least squares

We generate random instances of problem BLSSDP by setting $b = 0$ and taking A to have either binary entries sampled uniformly and independently at random or entries sampled from the standard normal distribution. We set A to be a $10 \times n$ matrix, and we vary the size of n. Observe that the size of the problem only depends on n. Finally we set k to be either 3 or 5 whenever A has normally distributed entries and $k = 8, 12$ whenever A has binary entries. These instances are considered in [61] and [87] in their computations of restricted isometry constants. We present our results in Tables 2.1 and 2.2 with time averaged over three instances. The experiments for the latter table are larger as these instances seem easier to solve.

Table 2.1: Performance of the different algorithms on the cardinality-constrained binary least squares problem with normally distributed entries.

n	k	m_{b}	m_{c}	Method	Time	#Abort	#Limit
15	3	10	45	PAJARITO_OA	1.66	0	0
				PAJARITO_TREE	0.24	0	0
				KOB_1	260.24	0	0
				KOB_2	9.78	0	0
				LAZY_SOC	10.65	0	0
				PURE_OA_SOC	0.44	0	0
15	5	10	45	PAJARITO_OA	20.60	0	0
				PAJARITO_TREE	3.03	0	0
				KOB_1	601.58	0	0
				KOB_2	216.08	0	0
				LAZY_SOC	24.01	0	0

[5]The experiments were performed on a 32 GB RAM ThinkPad Lenovo T490s machine running windows 10 with a Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz.

				PURE_OA_SOC	1.04	0	0
20	3	20	190	PAJARITO_OA	21.26	0	0
				PAJARITO_TREE	2.99	0	0
				KOB_1	1800	0	3
				KOB_2	410	0	0
				LAZY_SOC	92.88	0	0
				PURE_OA_SOC	1.63	0	0
20	5	20	190	PAJARITO_OA	95.96	0	0
				PAJARITO_TREE	17.20	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	231.24	0	0
				PURE_OA_SOC	1.66	0	0
25	3	25	300	PAJARITO_OA	64.43	0	0
				PAJARITO_TREE	7.67	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	1800	0	3
				PURE_OA_SOC	2.06	0	0
25	5	25	300	PAJARITO_OA	677.95	0	0
				PAJARITO_TREE	101.79	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	1800	0	3
				PURE_OA_SOC	4.69	0	0
30	3	30	435	PAJARITO_OA	630.63	0	0
				PAJARITO_TREE	60.52	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	1800	0	3
				PURE_OA_SOC	3.65	0	0
30	5	30	435	PAJARITO_OA	1800	0	3
				PAJARITO_TREE	1740.35	0	1
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	1800	0	3

				PURE_OA_SOC	8.85	0	0
35	3	35	595	PAJARITO_OA	674.56	0	3
				PAJARITO_TREE	121.10	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	1800	0	3
				PURE_OA_SOC	6.65	0	0
35	5	35	595	PAJARITO_OA	1800	0	3
				PAJARITO_TREE	1800	0	3
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	1800	0	3
				PURE_OA_SOC	84.80	0	0

Table 2.2: Performance of the different algorithms on the cardinality-constrained binary least squares problem with binary entries.

n	k	m_{b}	m_{c}	Method	Time	#Abort	#Limit
20	8	20	190	PAJARITO_OA	0.05	0	0
				PAJARITO_TREE	0.08	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	56.14	0	0
				PURE_OA_SOC	0.16	0	0
20	12	20	190	PAJARITO_OA	0.05	0	0
				PAJARITO_TREE	0.05	0	0
				KOB_1	1800	0	3
				KOB_2	-	3	0
				LAZY_SOC	201.14	0	0
				PURE_OA_SOC	0.79	0	0
30	8	30	435	PAJARITO_OA	198.92	0	0
				PAJARITO_TREE	27.24	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3

				LAZY_SOC	220.11	0	3
				PURE_OA_SOC	13.80	0	0
30	12	30	435	PAJARITO_OA	27.27	0	3
				PAJARITO_TREE	15.79	0	1
				KOB_1	1800	0	3
				KOB_2	-	3	0
				LAZY_SOC	-	3	0
				PURE_OA_SOC	11.57	0	0
40	8	40	780	PAJARITO_OA	90.77	0	0
				PAJARITO_TREE	29.03	0	0
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	110.53	0	3
				PURE_OA_SOC	14.71	0	0
40	12	40	780	PAJARITO_OA	186.97	0	3
				PAJARITO_TREE	42.45	0	3
				KOB_1	1800	0	3
				KOB_2	1800	0	3
				LAZY_SOC	-	3	0
				PURE_OA_SOC	15.50	0	0

Quadratic knapsack problem

We generate random instances of the quadratic knapsack problem following [131], who specify instances that have become the standard to computationally test this optimization problem. That is, we first set a *density* value $\Delta = 0.8$, which corresponds to the percentage of nonzero elements of the matrix C. Each weight $w_j,\ j \in [n]$ is uniformly randomly distributed in $[1, 50]$. The ij entry of C equals the ji entry and is nonzero with probability Δ, in which case it is uniformly distributed in $[1, 100],\ i, j \in [n]$. The capacity c of the knapsack is fixed at $\frac{1}{2}\sum_{j=1}^{n} w_j$. We present our results in Table 2.3, again with time averaged over three instances.

Table 2.3: Performance of the different algorithms on the quadratic knapsack problem.

n	m_b	m_c	Method	Time	#Abort	#Limit

10	10	45	PAJARITO_OA	11.40	0	0
			PAJARITO_TREE	0.83	0	0
			KOB_1	142.62	0	0
			KOB_2	11.73	0	0
			LAZY_SOC	15.31	0	0
			PURE_OA_SOC	0.73	0	0
20	20	190	PAJARITO_OA	1800	0	3
			PAJARITO_TREE	1800	0	3
			KOB_1	1800	0	3
			KOB_2	-	3	-
			LAZY_SOC	-	3	-
			PURE_OA_SOC	11.56	0	0
30	30	370	PAJARITO_OA	1800	0	3
			PAJARITO_TREE	1800	0	3
			KOB_1	1800	0	3
			KOB_2	-	3	-
			LAZY_SOC	-	3	-
			PURE_OA_SOC	174.38	0	0
40	40	780	PAJARITO_OA	1800	0	3
			PAJARITO_TREE	1800	0	3
			KOB_1	1800	0	3
			KOB_2	-	3	-
			LAZY_SOC	-	3	-
			PURE_OA_SOC	1800	0	3

2.6 Conclusion and future work

In this work we have proposed two algorithms based on spectral decomposition to improve outer approximation algorithms for integer semidefinite programs whenever these are derived from binary QCQPs. The experiments evaluating our approach are promising, and seem to indicate that integer semidefinite programming is a strong candidate to solve binary QCQPs, competing with state-of-the-art global solvers.

In a sense, these algorithms have been implemented in their most basic form, and we believe they can be improved using the Branch-and-Bound ideas of [38]. Our strengthenings were based on adding cuts derived from eigenvectors of a certain matrix that simultaneously diagonalizes the matrix determining the objective of an ISDP and an aggregation of the constraint matrices. We believe more progress can be made in this direction, and that different alternatives to this approach could be found.

Copositive and completely positive programming are also alternatives to solve BQCQPs. Although some algorithms that inner and outer approximate these cones have been proposed, such as the algorithm of Bundfuss and Dur [29] and the recent work of Gouveia et al. [66], the size of problems that can be solved remains limited. It would be interesting to find families of copositive and completely positive problems that can be cast as integer QCQPs and therefore could be tackled with integer semidefinite optimization problems, providing a new way to approach these hard conic problems.

Chapter 3

VECTOR CLOCK OPTIMIZATION VIA LATENCY LENGTHS

3.1 Introduction

Rumor spreading, and more generally information flow on graphs has been active research topic with important contributions in sociology [67], software engineering [140], and mathematics [69]. These problems typically involve the minimization of total number of messages, the total number of transmissions or the completion time under certain restrictions on how information can be shared [137]. In general, these problem address the case where a set messages -given upfront- are to be shared through the network.

In the same spirit, one may study problems where nodes are perpetually generating information over time, and restrictions on how nodes communicate results in an information latency at each node with respect to the latest available information at all other nodes. This idea of latency of information and temporal distance was introduced and studied by Kossinets, Kleinberg and Watts in [88] in the context of social networks, inspired by vector clocks arising in the study of distributed computing systems [92, 108]. Fundamentally, they find that such temporal measures provide structural insights that are not apparent from analyses of the pure social network topology. Combining the point of view of rapid information sharing and vector clocks, Chen, An, Niaparast, Ravi & Rudenko [34] introduce "vector clock problems" in which nodes are perpetually generating information which must be transmitted over the network while keeping the maximum or average latency low.

The study of this family of problems is motivated by the need to develop efficient communication protocols to keep information fresh under stringent communication constraints and where information is created perpetually. Indeed, very recently the telecommunications community has been interested in the development such algorithms [24, 119, 142] as large efforts stemming from academia, government and the private sector are aimed at developing a sixth generation (6G) mobile communication system [36, 84, 162]. This system is envisioned to make use of the synergy of terrestrial networks and satellite constellations to realize ubiquitous global communications. Low Earth Orbit satellite constellations (LEOs) are an essential component, by enabling low-cost and high-throughput global communication services [36], especially on sparsely populated areas and areas where physical infrastructure is hard to deploy. Satellites can communicate with each other directly via the inter-satellite link [35], which consist in a satellite sending a laser-beam to a recipient and thus requires the satellites to align. This results in discrete time frames in which satellites can communicate with each other, and naturally low latency of information in the network is paramount for this application [91]. We refer the reader to [148] for a simple exposition of these considerations.

In [34], the authors develop tools to solve broadcasting problems and their extension to the vector clock case - which we will introduce shortly - on rooted graphs, by combining combinatorial ideas with linear programming and rounding. In the rooted version of the problem, a node, the root, wants to spread a message as fast as possible, or keep the latency low respect to every other vertex in the vector clock case. Although their results are encouraging, the tools used seem hard to generalize to the *multi-commodity case* which more appropriately captures the problem of low latency in satellite-terrestrial communications. In this generalized version of the problem, we are given pairs of sources and terminals $(s_i, t_i),\ i \in I$ where I is some index set, and our objective is to send, as fast as possible, the message from each s_i to its corresponding t_i, or alternatively to keep the latency low between them. As far as we are aware, no poly-logarithmic approximation algorithm that runs in poly time is known for this generalized version of the problem: The best known approximation is super poly-logarithmic – see [122]. The purpose of this work is to further shed light on approaches to solve it.

In [34], the authors provide a constant-factor approximation algorithm to the rooted vector clock problem on trees. Interestingly, this algorithm provides *local* periodic communications (or *schedules*, a term we will introduce soon). Informally, this means that each edge is activated periodically, generating a “locally periodic" communication pattern over time. This schedule is then repeated to obtain a recurring infinite “global" schedule. Hence, as another aspect of this pattern, we have the *frequency* of activation of an edge for communication: how many times an edge is used for communication in a given time interval.

The purpose of this chapter is to study if the phenomenon of local periodicity - and hence edge frequencies- generalizes and controls the quality of schedules in both the (s, t) version of the problem and in general graphs. *Our main result is to show that indeed, there are locally periodic schedules that are near-optimal (within a poly-logarithmic factor) for multi-commodity rumor spreading problems.*

Multi-commodity Vector clock

To formally introduce the problems, we use the notation and ideas developed for vector clocks [34, 88, 92, 108]. Suppose we are given an undirected graph $G = (V, E)$. We consider a synchronous model where at each time step, each node generates fresh information, and is interested in the latest information generated at the other nodes. We consider information refreshing protocols where information exchange happens according to the **telephone model** [71] in synchronous rounds. In this model, each node at any time step t can communicate with at most one of its neighbors. Thus, at time t, a *matching* is selected and all information between both ends of the matched edges is shared (with no bandwidth limitations). We refer to a sequence of matchings indexed by time as a *schedule*. As we wish to understand the long-run behaviour of information spread in a graph, we will consider infinite schedules, i.e with an infinite time horizon.

To formalize these ideas, fix a schedule M and denote by $\phi_v^t(u)$ the *view* that v has of node u at time t, representing the latest information that v has of u at time t. More concretely, $\phi_v^t(u)$ is the largest $t' \leq t$ for which the information of u at time t' could be transmitted through the sequence of communications in our fixed schedule M and arrives at v by time t. We assume that everyone starts off with the freshest information and thus $\phi_v^0(u) = 0$ for all pairs of nodes v and u. Since every node is constantly generating new information, $\phi_v^t(v) = t$ for all $v \in V$. The *information latency* of v with respect to u at time t, denoted by $l_v^t(u)$ is given by $t - \phi_v^t(u)$, and is a measure of the freshness of information that v has of u. One may interpret this quantity as follows: suppose $l_v^t(u) = k$. Then, in the most recent k steps, there is a path of matched edges along which communication was scheduled in monotonic order (i.e., from u to v) which was able to convey the information at u from k steps earlier to v by the current time t. The objectives of the vector clock problem is to build schedules which minimize the sum or maximum of the latencies $l_s^t(t)$ where $(s, t) \in I$, the set of pairs of interest.

Problem 3.1 (Multi-commodity Max Vector Clock (**MCMVC**))**.** *Given a graph G and source-sink pairs* $(s_i, t_i), i \in I$*, find a schedule M that minimizes* $\max_{(s,t)\in I} \max_{\tau\geq 0} l_s^\tau(t)$.

Problem 3.2 (Multi-commodity Average Vector Clock (**MCAVC**))**.** *Given a graph G and source-sink pairs* $(s_i, t_i), i \in I$*, find a schedule M that minimizes* $\max_{\tau\geq 0} \sum_{(s,t)\in I} l_s^\tau(t)$.

The special rooted versions of these problems, where the s, t pairs are given by (r, v) where r is a fixed root in G and v ranges over the vertices of G have been studied in [34] by Chen et al. We present their results in the next section. They call these problems **MAXRVC** and **AVGRVC**, respectively.

Multi-commodity broadcast problems

One of the most studied problems in the setup of rumor spreading is the **minimum broadcast problem** [50, 51, 79, 122, 137]. We can state its objective in the previously introduced language, by relating the latency objectives defined above for infinite schedules to a 'static' one-phase version of delay. Here, the notion of latency is replaced by the notion of **delay**, the first time at which the information of interest is received. Given $(s_i, t_i), i \in I$ pairs, the goal is to pick a finite vector-clock schedule where all sinks t_i start with $l^0_{t_i}(s_i) = -\infty$. Define the first time at which the latency of any sink with respect to its corresponding source becomes finite as the delay in reaching this node. This is the first time at which the sink hears from the source since the start of the schedule.

Problem 3.3 (Multi-commodity Minimum Broadcast-Time Problem (**MCMBT**))**.** *Given a graph G and source-sink pairs (s_i, t_i) $i \in I$, find a finite schedule (under the telephone model) that minimizes the maximum delay of any sink t_i from its corresponding source s_i over all $i \in I$.*

We can also consider the average version.

Problem 3.4 (Multi-commodity Average Broadcast-Time Problem **MCABT**)**.** *Given a graph G and source-sink pairs (s_i, t_i) $i \in I$, find a finite schedule (under the telephone model) that minimizes the average delay at sinks t_i of heading from their corresponding sources s_i where the average is taken over all $i \in I$.*

In it most classical form, where the pairs are given by (r, v) with v ranging over the vertices of G, problem **MCMBT** is known as the *rooted minimum broadcast time problem (**MBT**)*. The analogous rooted version of **MCABT** is known as *rooted average broadcast time problem (**ABT**)*.

For the broadcast problems, in [137], Ravi introduced the first poly-logarithmic approximation algorithm for the Minimum Broadcast-Time problem in any arbitrary network. This work related such schemes to finding spanning trees that simultaneously have small maximum degree and diameter (the so-called poise of the graph). In [51], Elkin and Kortsarz give the best known $O(\frac{\log k}{\log\log k})$- approximation factor algorithm, where k is the number of terminals the information should be delivered to in the multicast version. On the hardness side, [50, 113] showed that the Minimum Broadcast Time Problem is 3-inapproximable unless $P = NP$ by reducing the problem to set cover.

In [34], the authors studied the rooted version of vector clock problems and related them to the corresponding broadcast variants within a logarithmic factor in both directions. Using these relations, they prove the following results.

Theorem 1.2 **of [34].** *Let G be a graph on n nodes and r be a vertex of G. Then, ABT has a $O(\frac{\log^2(n)}{\log\log n})$-approximation algorithm.*

Theorem 1.3 **of [34].** *Let G be a graph on n nodes and r be a vertex of G. Then, $AvgRVC$ has a $O(\frac{\log^3(n)}{\log\log n})-$approximation algorithm.*

They provide the only constant-factor approximation algorithm in their paper for $AvgRVC$ in the case when G is a tree.

Theorem 1.4 **of [34].** *There exists a* 40*-approximation for AvgRVC on trees.*

In particular, there are no known poly-logarithmic approximations for the multi-commodity versions (**MCMVC** and **MCMBT**, **MCAVC** and **MCABT**) of these problems. The best known guarantee for **MCMBT** is super-poly-logarithmic but sub polynomial due to Nikzad and Ravi [122], giving an approximation ratio of $2^{O(\log\log k\sqrt{logk})}$ where $k = |I|$, the number of source-sink pairs.

In this state of affairs, the constant factor approximation of Theorem 4 of [34] represents a surprising advance. This theorem works by constructing a schedule that is locally periodic. More specifically, we say a schedule is *global periodic* if there exists $P \in \mathbb{N}$ such that the schedule repeats every P time-steps. In other words, the matching used at time t is the same as $t + P$ for all $t \in \mathbb{N}$. We say a schedule is *local periodic* if for every edge e, there exists a period p_e such that the edge appears periodically in every p_e time-steps. More formally, for every edge e, there exists p_e, t_e such that the edge e appears only in time-steps $t_e + k \cdot p_e$ where $k \in \mathbb{N}$. Note that since the graph is finite, any local periodic schedule is also a global periodic (where the global period P is the lowest common multiple of all the local periods p_e).

Having a local periodic property provides much more structure to the schedule. For example, one can now attempt to relate the latency of a message along a path to the sum of the periods on the edges along the path. If such relationships exists, then instead of finding an infinitely long schedule, one can focus on finding suitable periods for each edge of the graph. In Theorem 4 of [34], the authors exploited the structure of a tree, where all uv paths are unique, and found the correct relationship between periods and latency. This allowed them to use dynamic programming and linear programming to find suitable periods and thus a constant approximation to the original problem.

Overview and outline

Our main result is to prove the existence of locally periodic schedules for the **MCMVC** and **MCAVC** problems that are only poly-logarithmic worse than optimal.

Theorem 3.1. *There exists local periodic schedules for **MCMVC** and **MCAVC** that are within a poly-logarithmic factor of optimal.*

The theorem is proved by showing that there exists a second order cone, integer program whose objective value approximates, up to polylogarithmic factors, the optimal value of **MCMVC** or **MCAVC** whenever G is an arbitrary graph (Theorem 3.3). The above theorem follows from the fact that the program can be randomly rounded to construct a *local* periodic schedule for **MCMVC** or **MCAVC**. The proof of Theorem 3.3 has two parts: First, we define the math program and show that a solution can be rounded within a poly-logarithmic factor to a local periodic schedule (Proposition 3.1); Second, we show that starting with an optimal infinite schedule for these problems, via a series of transformations, we can get a feasible solution to the math program that is only a poly-logarithmic factor away (Proposition 3.2).

We emphasize there that this theorem only proves the existence of provable good locally periodic schedules. While its proof involves constructing a local periodic schedule from the solution to the math program, we do not know how to solve (even approximately) these second order cone integer programs in polynomial time.

The utility of the result is two-fold: One, we have essentially reduced the poly-logarithmic approximability of these problems to that of approximating this second order cone integer program. Two, the definition of the program itself uses the *frequency* viewpoint of local periodic schedules as the underlying variables, and a length function on the edges defined as the reciprocal of the edge frequencies. Shortest paths under these length functions between multi-commodity pairs accurately estimate the delay of transmitting messages between them in this formulation. Thus, an alternate interpretation of our result is that the math program boils down to assigning frequencies to the edges of the undirected graph: The sum of edge frequencies around any node should be no more than one (the telephone model constraint on average). The objective to minimize is (the max or the average of) the shortest distances between pairs according to a latency function that is the shortest path function according to the reciprocal of the edge frequency. We believe both of these offer fresh approaches to approximating the multi-commodity rumor problems.

We also show that if G is a tree, then there is an algorithm that runs in polynomial time that approximates **MCMBT** (and as a consequence, also **MCMVC**) up to a constant factor. This was a problem left open in [34]. The best known approximation factor before our work was $O(\frac{\log k}{\log \log k})$ for k source-sink pairs that follows from the relationships derived earlier in [80].

Theorem 3.2. *Let G be a tree, and $(s_i, t_i), i \in I$ be source-sink pairs. Then,* ***MCMBT*** *has a 6-approximation algorithm.*

The rest of this chapter is organized as follows:

(a) In Section 3.2 we provide an algorithm for multi-commodity broadcast problem (**MCMBT**) on trees and prove that the algorithm provides a factor of 6 approximation. We then proceed to prove Theorem 3.2.

(b) In Section 3.3 We prove Theorem 3.1.

(c) In Section 3.4 we conclude with some remarks.

3.2 The multi-commodity vector clock and minimum time broadcast problems on trees

In this section, we provide an algorithm for multi-commodity broadcast problem (**MCMBT**) on trees and prove that the algorithm provides a factor of 6 approximation to the problem, and prove Theorem 3.2. Using the ideas in Theorem 2.2 of [34], this also gives an approximation algorithm for **MCMVC** in trees.

Corollary 3.1. *Let G be a tree, and $(s_i, t_i), i \in I$ be source-sink pairs. Then,* ***MCMVC*** *has an $O(1)$-approximation algorithm.*

This complements and nearly matches the 2-approximation derived in [34] for the corresponding rooted analogue, MaxRVC in trees.

Let G be a tree rooted at r. Let $\mathcal{S} = \{(s_i, t_i), i \in I\}$ be the set of source-sink pairs. Given an s-t pair, we define $P(s, t)$ to be the unique path that joins them. We define the intersection graph G' as in the proof of 3.2. To simplify the presentation of our algorithm, we assume that G' has just one connected component. Otherwise, we simply run the algorithm in parallel in each component. We further assume that each vertex of G belongs to some $P(s,t)$. Otherwise, we simply remove that vertex from G. Finally, we assume without loss of generality that the root r of G is the source for the pair (s_1, t_1). In other words, $s_1 = r$. Our algorithm is as follows:

1. Build the intersection graph G' of $\mathcal{S}$.

2. Using breadth-first-search in the graph G' starting at vertex (s_1, t_1), label vertices with the layer number corresponding to its distance to the root (e.g. the first layer are the neighbors of (s_1, t_1), and the second layer are neighbors of the first layer that is not yet labelled, and so on). Denote the layers L_1 up to L_p.

3. For each layer L_k, define R_k to be the set of vertices v of G that satisfy the following conditions:

   - v belongs to the intersection of paths joining s_i to t_i and s_j to t_j for some $(s_i, t_i) \in L_k$ and $(s_j, t_j) \in L_{k+1}$.
   - v is the farthest away from the root among the vertices in the intersection of $P(s_i, t_i)$ and $P(s_j, t_j)$.

4. For all i and each v in R_i, let $T_G(v)$ be the subtree of G rooted at v. Define $T(v)$ to be the intersection of $T_G(V)$ and the union of the paths of the (st) pairs in L_{i+1} Notice that this is a subtree of G rooted at v, and that for each $v \in R_i$ these trees are disjoint, by the second condition in the definition of v.

5. Let $T_G(s_1) = P(s_1, t_1)$

6. For $v = s_1$ and for each $v \in R_i$ where i is odd, do in parallel: find the optimal MBT schedule[1] in the subtree $T(v)$ rooted at v. Run this schedule backwards to transmit the message of every node in the tree to v. Let $\mathcal{T}_{odd}$ be this schedule.

7. Repeat the above for all $v \in R_i$ where i is even. Let $\mathcal{T}_{even}$ be this schedule.

8. Run the schedules $\mathcal{T}_{odd}$ again.

9. In sequence, run $\mathcal{T}'_{odd}, \mathcal{T}'_{even}, \mathcal{T}'_{odd}$ where $\mathcal{T}'_{odd}, \mathcal{T}'_{even}$ are the schedules $\mathcal{T}_{odd}, \mathcal{T}_{even}$ respectively but backwards.

[1]The optimal rooted MBT schedule in a tree can be computed in polynomial time by dynamic programming.

We now proceed to prove Theorem 3.2.

Proof. Let $\mathcal{M}^*$ be the optimal schedule for the **MCMBT** problem 3.3 in G. Let OPT be the optimal transmission time, i.e. the time by which $\mathcal{M}^*$ is able to send the message from each s_i to t_i. We first show that our algorithm will successfully send every message by the end of its execution.

Consider a source-sink pair (s, t). If $s = s_1$, then its message is sent in the first time we run Step 5. Now suppose s belongs to layer L_{k+1}. By construction, there exists $r_s \in R_k$ that is an ancestor of s. Let v_s be the lowest common ancestor of s and t; in other words, let v_s be the vertex on $P(s, t)$ with the shortest distance to the root s_1. We claim that after Step 6, 7, and 8, s can successfully send a message to v_s.

First assume $k + 1$ is odd. Then Step 6 ensures a message is sent from s to r_s. Note that both v_s and r_s by definition are both ancestors of s. This implies both r_s and v_s are on the path from s to the root. If s sees v_s on its way to r_s, then our claim is true. Thus, we may assume v_s is an ancestor of r_s.

Let $r' \in R_k$ such that $r_s \in T(r')$. Note that r' by definition belongs in a source-sink path with label $k - 1$ in G'. Since r' is also an ancestor of s, r' must be closer to the root than v_s, otherwise, the path from s to v_s contains r' and (s, t) should get label k instead, a contradiction. Then, v_s is on the path from r_s to r', implying when Step 7 would allow a message to be sent from r_s to v_s, proving our claim. If $k + 1$ is even, then the same analysis can be done to show a message is sent from s to v_s in Steps 7 and 8.

Since Step 9 is running Step 6, 7 and 8 backwards, it is easy to check that in this step a message can be sent from v_s to t. Thus it remains to show that the final schedule has length at most $6OPT$. Note that schedule $\mathcal{T}$ is the optimal schedule to collect all messages in the subtree $T(r_s)$ and $\mathcal{M}^*$ is an schedule that also accomplishes this. Thus the length of $\mathcal{T}$ is at most OPT. The same argument holds for $\mathcal{T}'$. Therefore, it follows that the final combined schedule has length at most $6OPT$.

□

Proof of Corollary 3.1 . The proof follows immediately by applying Theorem 6 of [34], which states the following: If there exists an α-approximation for the Minimum Broadcast-Time Problem, then there exists a 2α-approximation for the maximum rooted vector clock problem. We observe that although the proof is given for the rooted case, it goes through verbatim for the multi-commodity case. □

3.3 The multi-commodity vector clock problem on general graphs

We begin by introducing a mathematical program designed to model optimal schedules to the **MCMVC** problem. By changing the objective of this program we can also capture the **MCAVC** problem. Therefore, we will only discuss the program for the former problem in this extended abstract.

Our first observation is that schedules for the **MCMVC** are, by definition, infinite, whereas a mathematical program can only characterize finite schedules. Fortunately, Lemma 3.1 (Theorem 2.1) from [34], implies that it suffices to consider only finite schedules and then repeat it ad infinitum. Recall that an infinite schedule M is (globally) *periodic* with period P if $M_i = M_{i+P}$ for all $i \in \mathbb{N}$.

Lemma 3.1. *[34] Let M be an infinite schedule and $L^t(M)$ be the corresponding matrix of latencies (i.e., $L^t(v,u) = l^t_v(u)$). Let $opt \in \mathbb{N}$ be such that $\|L^t(M)\|_1 \leq opt$ for all $t \in \mathbb{N}$. Then, there exists a periodic, infinite schedule S with period $opt + 1$ and latency matrices $L^t(S)$ such that $\|L^t(S)\|_1 \leq 2opt$ for all $t \in \mathbb{N}$.*

The result of Lemma 3.1 holds regardless of the matrix norm we choose. Hence we can use it to only account for source-sink pairs of interest and also either their maximum or their average (sum).

The key to formulate our mathematical program is the following observation: Under a periodic schedule of period P, each edge has a certain frequency, the number of times the edge is used for communication divided by P. Our idea is to use a mathematical program to obtain these fractional frequencies as variables and use it to build a schedule. Ideally, we want the frequencies to satisfy the following:

1. Since a node can communicate with at most one neighbor at a given time t under the telephone model, the sum of the frequencies of edges incident to a node cannot exceed 1. Formally, if p_e is the frequency of an edge, and $\partial(u)$ is the set of incident edges to u, we require $\sum_{e\in\partial(u)} p_e \leq 1$.

2. Consider a pair of adjacent nodes u and v linked by edge e. If the communication over e happens s times in the time window $[1, P]$ in a periodic schedule, then our estimate of its frequency is $\frac{s}{P}$. Then, on average, the latency of communication between these two adjacent nodes is $\frac{P}{s} = \frac{1}{p_e}$ which we will call its latency length.

3. Suppose u and v are non-adjacent nodes. Let $e_1, \ldots, e_k$ be a simple path between them. By the previous observation, we expect that the average latency for

communication between them via this path is $\sum_{i=1}^{k} \frac{1}{p_{e_i}}$ where p_{e_i} is the frequency of edge e_i. Thus, if we think of the value $\frac{1}{p_e}$ as edge (latency) lengths, then the communication time between the nodes is estimated by the shortest path given by these lengths.

These lead us to our mathematical formulation for the **MCMVC** problem in a graph G. Given a set of (s_i, t_i) pairs with $i \in \mathcal{I}$ we consider the following program.

$$\begin{aligned} &\min \max_i d_{s_i t_i} \\ &s.t: \sum_{e \in \partial(v)} p_e \leq \frac{1}{2} \; \forall \, v \; \in V, \\ &l_e = \frac{1}{p_e} \; \forall e \in E, \\ &p_e > 0 \end{aligned} \tag{3.1}$$

where $d_{s_i t_i}$ is the shortest-distance between s_i and t_i according to edge lengths l_e.

Note that the first constraint follows from our first observation about frequencies, but from reasons that will become clearer later, we have made the more stringent requirement that the sum of the p_e over the edges incident to a node is at most $\frac{1}{2}$. The second constraint is motivated by our second observation on how to relate frequencies and its delays on an edge. The objective function then follows from our third observation where the latency between any pair should correspond to the shortest path under these length functions. The last positive constraint simply ensures that all variables are well-defined. By using the bounds on the objective function for the problem, we can take care of the open constraint on p_e by just requiring each of them to be at least $\frac{1}{n^2}$.

Notice that this MP cannot be solved as stated, as we need some way to specify in the program the fact that the variables $d_{s_i t_i}$ denote the value of the shortest path between u and v under edge lengths $\frac{1}{p_e}$. To do this, we use a standard min-cost unit flow formulation of shortest paths. We call this the *flow formulation* of $MCMVC$ which we define formally in Section 3. We also point out that by changing the objective function, we can easily accommodate other l_p-norms and their rooted variants.

Let G be a graph on n nodes and m edges. Define the *path polytope* $\mathcal{P}_{st}$ to be the set of vectors $f^{st} = (f^{st}(e)) \in \mathbb{R}^m$ such that the following equations hold:

$$\sum_{w \in \partial s} \left(f^{st}(sw) - f^{st}(ws)\right) = 1$$
$$\sum_{w \in \partial t} \left(f^{st}(tw) - f^{st}(wt)\right) = -1$$
$$\sum_{w \in \partial a} \left(f^{st}(aw) - f^{st}(wa)\right) = 0, \ a \notin \{s, t\}$$
$$f^{st}(e) \in [0, 1] \ \forall e \in E.$$

f^{st} represents one unit of flow going from s to t. To simplify the notation, we write f^i instead of $f^{s_i t_i}$ for $i \in I$ and P_i instead of $P_{s_i t_i}$. We can therefore re-formulate program (3.1) by replacing the objective function with

$$\min \ \max_i \sum_e \frac{1}{p_e} f^i(e),$$

and insist that $f^i \in P_i, \forall i \in I$. An important fact about this problem is that at optimality, the f variables are integral along shortest paths. (The proof of this and other unproven claims are in the Appendix).

Observation 3.1. *If there exists an optimal solution for the flow-reformulated version of program* (3.1)*, then there exists an optimal solution for which the variables $f^i(e)$ are binary, for all $e \in E$ and all $i \in I$.*

Proof. Suppose that we are given an optimal solution for (3.1) (p_e, f^i) with optimal value opt and where the f^i are not binary. Assign to the graph G the edge lengths $\frac{1}{p_e}$ and solve the shortest path problem on G using the path polytope. This returns flow variables $\hat{f}^i$ for every i which are binary since the path polytope is integral [39]. Notice that $(p_e, \hat{f}^i)$ is feasible for (3.1) and so the objective at that point, i.e $\sum_e \frac{1}{p_e} \hat{f}^i(e) \geq opt$. Moreover, notice that $(f^i(e)), e \in E$ is a feasible solution for the shortest path problem on the graph G with edge lengths $\frac{1}{p_e}$. It follows that $opt = \sum_e \frac{1}{p_e} f^i(e)$ is minimized by $\sum_e \frac{1}{p_e} \hat{f}^i(e)$, as this last quantity is the optimal value for the shortest path problem on G with edge lengths $\frac{1}{p_e}$. Thus, as desired, $\sum_e \frac{1}{p_e} \hat{f}^i(e) = opt$. □

We now turn to prove the main theorem of this section. Namely, that program 3.1 provides an $O(\log^5 n)$ approximation factor for $MCMVC$.

Theorem 3.3. *For any undirected graph, let opt_{mp} denote the optimal value of program 3.1 and opt_{MCMVC} the maximum $s - t$ latency of the optimal schedule $\mathcal{M}$. Then for* *some constants c, c',*

$$c \log^2 n \cdot opt_{mp} \geq opt_{MCMVC} \geq \frac{opt_{mp}}{c' \log^3 n}.$$

Furthermore, the proof of the first inequality constructs a locally periodic schedule with the corresponding guarantee.

We prove the two inequalities in Propositions 3.1 and 3.2 below respectively.

To prove the first inequality, we will use the so-called *pinwheel scheduling* [77]: Suppose that we are given a set of integers $A = \{a_1, a_2, \ldots a_n\}$. The pinwheel problem consists in finding an infinite sequence S of integers $j_1, j_2, \ldots$ over $\{1, 2, \ldots, n\}$ such that any sub sequence of $a_i (1 \leq i \leq n)$ consecutive entries contains at least one of the i. We call such a sequence *correct* for A. For example, the sequence $1, 2, 1, 2, \ldots$ is a correct sequence for $A = \{a_1, a_2\} = \{2, 3\}$[2]. The main result we will be using is Corollary 3.2 from [77]:

Corollary 3.2 **of [77].** *Suppose $A = \{a_1, \ldots, a_n\}$ where $\sum_{i=1}^{n} \frac{1}{a_i} \leq \frac{1}{2}$. Then, we can find an correct sequence for A using a greedy algorithm.*

We refer the reader to the paper for the proof of this corollary and for the greedy algorithm used to find the sequence. Further, let us point out that the sequence provided by the greedy algorithm is periodic. This will ensure that the schedule that we provide in the next lemma is *locally periodic*.

Proposition 3.1. *Given an optimal solution p^*, f^* to (3.1) with objective value $opt_{mp} = Z$, there exists a locally periodic, infinite schedule of maximum latency L at most $c \cdot \log^2 n \cdot Z$ for some constant c.*

Proof. We will build an infinite schedule using Corollary 3.2 from [77]. We will then prove that this schedule has the following property: for each (s, t) pair, the latency between s and t is at most $c \cdot \log n \cdot Z$ at any arbitrary time T. For each vertex $v \in V$, let $p^*(v)$ be the set of corresponding p^* values of edges in $\partial(v)$. Define $A_v := \{\lfloor \frac{1}{p} \rfloor, p \in p^*(v)\}$. By Corollary 3.2 from [77] and the feasibility of p^* for program 3.1, there exist an infinite schedule $\mathcal{M}_v$ such that each edge e in $\partial(v)$ is activated at least once in each $\frac{1}{p_e}$ time intervals. This procedure provides us with

[2]We observe that the vocabulary used in [77] is slightly different. In this paper, the authors call a correct sequence a “successful schedule". We have changed the vocabulary as we have given a different meaning to the word *schedule*: a sequence of matchings in G.

$|V| = n$ vertex schedules, or a table of vertex schedules. A column of this table corresponds to a time T, and each vertex wants to active one of its adjacent edges at this time. This may lead to *conflicts*: two edges adjacent to the same vertex wanting to be activated at the same time (e.g., from their other endpoints). For now, let us suppose there are no conflicts. we will show how to deal with this afterwards.

Consider an s, t pair and an arbitrary time T. By integrality of the flow f^*, there is a unique path between s and t determined by the f^i that are 1. Call this path $e_1, \dots, e_q$. By optimality of p^*, f^* we have $\sum_{j=1}^{q} p^*_{e_j} \leq Z$. Now, consider time $T' = T - \sum_{j=1}^{q} \frac{1}{p^*_{e_j}}$. By definition of the pinwheel schedule, by time $T' + \frac{1}{p^*_{e_1}}$, edge e_1 has been activated. After additional time $\frac{1}{p^*_{e_2}}$, edge e_2 is activated, which shows that by time $T' + \frac{1}{p^*_{e_1}} + \frac{1}{p^*_{e_2}}$ edges e_1 and e_2 are activated, in that order. Continuing this way, we see that after time $\sum_{j=1}^{q} \frac{1}{p^*_{e_i}}$ edges $e_1, \dots e_q$ have been activated, with e_j being activated exactly after e_{j-1} for all $1 \leq j \leq q$. Therefore, at time T, the latency between s and t is at most $\sum_{j=1}^{q} \frac{1}{p^*_{e_i}}$ which in turn is at most Z.

Let's now discuss how to solve conflicts. The idea is simply to start the vertex schedules at different random times, and diluting time by a factor of $O(\log n)$ so that with high probability there are no conflicts. However, we will need to do this carefully so as to not lose the local periodicity of edges incident to a vertex. For each v, start the schedule of v at time t_v where t_v is drawn uniformly at random from $1, \dots, N_v$ where N_v is the period of pinwheel schedule of the edges incident on v. Now, fix a time T and let e be incident to v. Next, we assume that each p is the inverse of a power of 2. We can do this by rounding each p_i to the closest power of 2 i.e. if $\frac{1}{2^{j+1}} \leq p \leq \frac{1}{2^j}$ for some j, we set $p = \frac{1}{2^{j+1}}$. This results in losing a factor of 2 in objective. Since we insisted $p \geq \frac{1}{n^2}$, there are at most $\log n^2 \leq 2 \log n$ different distinct values for the edge frequencies in the graph. Furthermore, from the property of the greedy algorithm used to provide the pinwheel schedule, the probability of edge e appearing at time T in the schedule of v is p_e. This process generates at time T a random graph G_T where an edge e is selected at random with probability p_e. Since $\sum_{e \in \partial(v)} p_e \leq \frac{1}{2}$, we can use Hoeffding's inequality and a simple union bound to argue that with probability at least $1 - \frac{1}{n}$, the maximum degree of G_T is no larger than $4 \log n + 1$.

By Vizing's theorem [46], there exists an edge coloring of G_T with at most $4 \log n + 2$ colors, where each color class is a matching. In conclusion, with high probability, at time T there are no more than $4 \log n + 2$ conflicts. We can solve these conflicts by simply scheduling these matchings in any order by diluting time by a factor of $4 \log n + 2$ but this would destroy local periodicity. We use the fact there are only

$2\log n$ distinct frequencies and dilute time by a further factor of $2\log n$ to retain local periodicity. Since all frequencies are inverses of powers of two, we bucket together the different p's according to their corresponding power of 2: for $j \in 1, \ldots, 2\log n$ we set $B_j = \{p_e = \frac{1}{2^j}\}$. The crucial observation here is that if two edges in the same bucket B_j are in conflict at time T, then they will be in conflict in time $T + 2^j$. This periodicity of the conflicts will allow us to schedule the edges without losing local periodicity. First, we dilute time T by a $2\log n$ factor, obtaining time slots $T_1, \ldots T_{2\log n}$. Since we have at most $2\log n$ frequency buckets, we assign edges in this time T in bucket j to time T_j. We further dilute each time T_j by an additional $(4\log n + 2)$ factor (the maximum degree of G_T), obtaining times T_{j_k} where k ranges from 1 to $4\log n + 2$. Since there are overall at most $2\log n$ frequency buckets in time T and there are at most $4\log n + 2$ conflicts of edges (over all buckets and hence) in bucket B_j, we appropriately assign each edge in bucket j to one of the slots T_{j_k} with k ranging from 1 to $\log n + 2$. By the periodicity of conflicts, we can use this same assignment of slots in times $T + l \cdot 2^j$ with $l \in \mathbb{N}$. It is easy to check that the resulting schedule has no conflicts.

To conclude, observe that an edge p_e is periodic from the perspective of both of its endpoints. However, it might happen that both endpoints want to active the edge in an out of sync manner. To remedy this, we simply assign each edge to one of its end points, say by using a depth-first algorithm to order the nodes of the graph and assigning the edge to the endpoint with lower value in the ordering. If a node v has an edge not assigned to it, we don't consider it when constructing schedule $\mathcal{M}_v$ with pinwheel. Thus, the schedule we obtain is locally periodic. This concludes the proof of Proposition 3.1. □

We proceed to prove the second inequality in Theorem 3.3.

Proposition 3.2. *There exists a feasible solution to program 3.1 with objective value* $O(\log^3 n) opt_{MCMVC}$.

Proof. The proof requires various steps, which we list next for clarity. For simplicity we will write opt for the value of opt_{MCMVC}.

1. Using the optimal **MCMVC** schedule, we find a schedule of **MCMBT** with the same objective value opt. This schedule specifies (s, t) paths in the graph.
2. We build the intersection graph of these paths and using a network decomposition idea [9], we decompose it into a set of $O(\log n)$ subgraphs, each of which is made of disjoint low-diameter subgraphs (where the diameters are also $O(\log n \cdot opt)$).

3. We convert each low-diameter subgraph into a rooted tree spanning its nodes such that the time to perform rooted broadcast in the tree is $O(\log n \cdot opt)$.

4. In each tree, we use the this rooted broadcast schedule for **MBT** to derive a solution for (3.1), of value at most $O(\log^2 n \cdot opt)$, thus losing an additional logarithmic factor in this step.

5. Finally, we combine these solutions for the $O(\log n)$ different forests that a node may participate in to get a final solution to (3.1) of value $O(\log^3 n \cdot opt)$.

Let G be an arbitrary graph and (s_i, t_i) $i \in I$ be source sink pair. To simplify our proof, we will assume that G is connected, and that every node in G belongs to some $s - t$ pairs. Otherwise we can simply repeat the argument in the connected components of G removing vertices of G which are not in any $s - t$ path.

We begin with the first bullet. Let $\mathcal{M}^*$ be an optimal schedule for problem **MCMVC** with minimum maximum latency opt. Selecting an arbitrary time T greater than opt and looking backwards opt steps, i.e from time $T - opt + 1$ to T, gives a schedule $\mathcal{M}'$ for the **MCMBT** problem with maximum delay at most opt as clearly in this time interval every $s - t$ pair is guaranteed to have transmitted a message from s to t.

Next we detail how to accomplish the second bullet. The idea is to use a low-diameter decomposition of the *intersection* graph given by the $s - t$ pairs. Given an s-t pair, we define $P(s, t)$ to be the unique path that $\mathcal{M}'$ uses to send the message from s to t. We define the intersection graph G' whose vertex set is the set of s, t pairs, and there is an edge between (s_i, t_i) and (s_j, t_j) if and only if $P(s_i, t_i)$ and $P(s_j, t_j)$ intersect. Using a well-known (see, e.g. Section 2.1 of [17]) result from [9], we can decompose G' into $O(\log n)$ forests, where for any tree in any of the forest, the depth of the tree is at most $O(\log n)$.

We then turn to the third bullet. We denote by C_i^j the j^{th} tree of forest i, we let Q_i^j be the subgraph consisting of all the edges (and vertices) of G contained in the $s - t$ paths of G' and let r_{ij} be an arbitrary source node in Q_i^j.

We focus on converting a single subgraph Q_i^j rooted at r_{ij} to a tree so we will drop the index notation and simply write Q and r. First, assume without loss of generality that $\mathcal{M}'$ is a schedule for the problem **MCMBT** in Q. This is well defined by our construction of Q. Denote the reverse schedule of M' by $\mathcal{M}'_{-}$. Denote by $\hat{\mathcal{M}}$ the schedule that alternates between $\mathcal{M}'$ and $\mathcal{M}'_{-}$. This schedule has delay $2opt$, in the sense that after $2opt$ steps, every t has heard from their corresponding s and every s has heard from their corresponding t.

Let v be an arbitrary vertex in Q. We claim that by repeating schedule $\hat{\mathcal{M}}$ $O(\log n)$ times r has heard from v. This follows by observing that any node in a path at distance l from the path containing the root r in G' can reach another node in a path at distance $(l-1)$ by running $\hat{\mathcal{M}}$ once. Since the depth of Q is $O(\log n)$, the claim follows.

Observe that by running this same repeated schedule backwards, we are guarantee to transmit a message from r to every node v. Hence, it gives a solution to the rooted broadcast problem with delay $2opt \cdot \log(n)$. By using the first edge via which every node in Q hears the broadcast message from r as its incoming edge (as in [137]), we can define a directed spanning tree rooted at r which provides a solution to the rooted MBT problem within time $O(\log n \cdot opt)$ as desired.

The fourth bullet is the content of Lemma 3.2. By using the distance between any $s-t$ pair as being at most the distance via the root in the tree that they both occur in, we get a bound of $O(\log n \cdot opt)$ as an upper bound on the delay between the pair. Using the lemma, we conclude that the feasible solution it gives has objective value an additional logarithmic factor worse.

Lemma 3.2. *Given a tree Q with root r, let $\mathcal{M}$ be any schedule to the rooted MBT problem in Q. Then from $\mathcal{M}$ we can derive, for each edge e a frequency p_e that is feasible for (3.1) such that the sum of inverse of the frequencies over any path from a vertex to the root is at most $O(\log n)$ times the delay at that node.*

Proof. First, we can assign to each edge in Q an index l_e^Q such that the sum over the l_e^Q over the path rv equals the delay from r to v. Let u be a node in Q, and $v_1 \ldots v_k$ its children. suppose the message from r arrives to u at time k. Then, u will send the message to its children in some order according to $\mathcal{M}$. If v is such a child, we let $l_{uv} = i$ if v is the i^{th} children of v to talk to u after u received the message from r. Notice that by definition this implies that v has delay $k+i$. It is not hard to see that these lengths satisfy the required property.

We set

$$p_e^Q = \frac{1}{2l_e^Q(\log n + 1)}. \tag{3.2}$$

For any vertex v with k children in the tree, the sum of their assigned frequencies is

$$\sum_{e \text{ child of } v} p_e^Q \leq \sum_{i=1}^{k} \frac{1}{2i(\log n + 1)} \leq \frac{\log n}{2(\log n + 1)}.$$

The edge from v to its parent has p_e at most $\frac{1}{2(\log n+1)}$ leading to $\sum_{e\in\partial(v)} p_e^Q \leq \frac{1}{2}$ showing that it is feasible for (3.1).

Next, assign to each edge in G lengths given by $\frac{1}{p_e}$. The distance between r and v under this length is then simply the sum over the edges in the r-v path of $\frac{1}{p_e} = 2l_e^Q(\log n + 1)$. This is only $O(\log n)$ times the delay from r to v.

□

To finish the proof, we prove the last bullet. For each edge, recall that it may be part of $O(\log n)$ different forests and hence in up to that many different spanning trees in steps 2 and 3. If its frequencies in these trees are $p_e^1, \ldots, p_e^l$ for $l = O(\log n)$, we reset its final frequency to be $p_e = \frac{1}{l}\max_i\{p_e^i\}$. It is not hard to see that this also stays feasible but weakens the objective by a further logarithmic factor, giving the final result.

□

Theorem 3.1 can also be proved for problem **MCAVC**. The proof is similar to the proofs for **MCMVC** and the theorem follows immediately from proving Proposition 3.1 and 3.2 for **MCAVC**. The versions below imply that there is a locally periodic schedule for **MCAVC** that is within $O(\log^6 n)$ of optimal.

Proof of Proposition 3.1 for MCAVC. The proof is identical to the original version for **MCMVC**. Given a solution to the math program under the new objective that sums over all distances between (s_i, t_i) pairs insterad of the max, one can follow the same procedure to obtain a local periodic schedule. This schedule guarantees that every edge e appears $O(\log^2 n) * l_e$ where l_e is the edge-length from the math program. Then, any (s_i, t_i) pair also has latency at most $O(\log^2 n) * d_i$ at any time t where d_i is the shortest distance between s_i and t_i under the edge lengths. Thus the average latency for all pairs is also within $O(\log^2 n)$ factor of the math program objective, proving our proposition. □

*Proof of Proposition 3.2 for **MCAVC**.* The idea is to reduce the problem into $\log n$ instances of **MCMVC** and then stitch these solutions back together. This will add another $O(\log n)$ factor to the approximation. First, given a schedule for **MCAVC** with average latency of OPT_{MCAVC}. Fix a particular time and separate the (s_i, t_i) pairs into $O(\log n)$ buckets of powers of 2 based on their latency at that time. For each bucket k, we ensure that pairs within the bucket can receive a message between time 2^k and 2^{k+1}.

Then applying Proposition 3.2 from Step 2, for each bucket k, one can obtain a solution to the math program such that the pairs within each bucket k has a graph distance of at most $O(\log^3 n) * 2^{k+1}$. To combine all the buckets, we simply blow up the edge-lengths by another factor of $B = O(\log n)$ where B is the number of buckets. Furthermore, if an edge appears in multiple buckets, and thus may be assigned multiple edge-lengths, simply take the shortest one. This ensures that the shortest distance of any (s_i, t_i) pair in bucket k is at most $O(\log^4 n) * 2^{k+1}$. Therefore, the average of all (s_i, t_i) pairs is at most $O(\log^4 n) * OPT_{MCAVC}$.

Lastly, it remains to show that combing the k buckets does not violate any of the constraints. Mainly, we need to check that sum of the periods p_e around any vertex remains at most $1/2$. Note that within each bucket, this constraint was held. If we do not adjust these periods, if we naively sum across all buckets, then we can guarantee that the sum is at most $B/2$. Since we blew up the length by a factor of B, thus reducing the periods by a factor of B it follows the sum is now at most $1/2$. Note that even though for any edge that appears in multiple buckets we kept the shortest distance, and hence the largest period, the sum still remains below $1/2$. This completes our proof.

□

3.4 Summary and future work

In this chapter, studied multi-commodity version of previously studied minimum broadcast, average broadcast, minimum vector clock and average vector clock. We provided an algorithm to solve up to constant factors the multi-commodity minimum broadcast problem on trees, and introduced a second order cone, integer program to solve the problem on general graphs. The main theoretical question that remains open is whether we can solve, or at least approximate the provided program in polynomial time. We note that a positive answer would subsume all previous results, as this is the most general version of the vector clock/broadcast problem. It could be however that the more constrained version of the problems have better approximation ratios. A different way to try to solve this problem is to provide an alternate formulation for specifying the shortest paths between vertices that does not rely on the flow polytope. Providing a constant factor approximation algorithm for MCABT in trees is also open.

CONCLUSIONS AND FUTURE WORK

In this thesis, we revisited the question of approximating semidefinite programs with LPs. Care was taken to ensure the LPs were not arbitrary, but flexible enough to produce strong relaxations. In Chapter 1, we presented results showing that strong relaxation can be obtained in four families of problems. In a sense, these linear programs can be viewed as programs that incorporate spectral information of the problem while remaining linear. We believe the questions considered open up different avenues of research. Although we focused on spectral information, it is possible that many other approaches can be taken in the formulation of instance-dependent relaxations. Perhaps the most interesting question left open is whether the linear programs proposed can approximate the maximum cut problem to a factor better than 2 for any graph. The experiments performed suggest that this is the case.

In the second chapter, we explored the consequences of incorporating spectral information into outer approximation algorithms to solve binary semidefinite programs, with varying degrees of success. We believe there is still much work to be done in this area. For example, we posit that strong integral cuts can be derived for the outer approximation integer second order programs proposed. Advances in this direction would have a direct impact on the quality of the outer approximation algorithms, allowing one to tackle larger binary quadratically constrained quadratic problems. For the time being, it seems that using binary semidefinite programming to solve BQCQPs is still behind alternative approaches such as linearization and spatial branch and bound. Nonetheless, we point out that our investigation was very preliminary, and probably substantial improvements can be made.

In the last chapter, we consider an infinite horizon vector clock problem, and derived a structural result for optimal solutions by showing that there are near-optimal locally periodic schedules. Although we were not able to solve the mathematical program presented to find the periods, we believe that this result could be exploited to find a polynomial time algorithm to approximate the multi-commodity vector clock problem. For now, this problem remains open.

BIBLIOGRAPHY

[1] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.4 (2010), pp. 433–459.

[2] Tobias Achterberg. "SCIP: solving constraint integer programs". In: *Mathematical Programming Computation* 1 (2009), pp. 1–41.

[3] Amir Ali Ahmadi. "Algebraic relaxations and hardness results in polynomial optimization and Lyapunov analysis". In: *arXiv preprint arXiv:1201.2892* (2012).

[4] Amir Ali Ahmadi and Georgina Hall. "Sum of squares basis pursuit with linear and second order cone programming". In: *Algebraic and geometric methods in discrete mathematics* 685 (2017), pp. 27–53.

[5] Amir Ali Ahmadi and Anirudha Majumdar. "DSOS and SDSOS optimization: more tractable alternatives to sum of squares and semidefinite optimization". In: *SIAM Journal on Applied Algebra and Geometry* 3.2 (2019), pp. 193–230.

[6] Farid Alizadeh. "Interior point methods in semidefinite programming with applications to combinatorial optimization". In: *SIAM Journal on Optimization* 5.1 (1995), pp. 13–51.

[7] Noga Alon and Benny Sudakov. "Bipartite subgraphs and the smallest eigenvalue". In: *Combinatorics Probability and Computing* 9.1 (2000), pp. 1–12.

[8] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 10.0.* 2022. URL: `http://docs.mosek.com/9.0/toolbox/index.html`.

[9] Baruch Awerbuch and David Peleg. "Routing with polynomial communication-space trade-off". In: *SIAM Journal on Discrete Mathematics* 5.2 (1992), pp. 151–162.

[10] Egon Balas. "A note on the branch-and-bound principle". In: *Operations Research* 16.2 (1968), pp. 442–445.

[11] Radu Baltean-Lugojan et al. "Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks". In: *preprint: http://www. optimization-online. org/DB_ HTML/2018/11/6943. html* (2019).

[12] Xiaowei Bao, Nikolaos V Sahinidis, and Mohit Tawarmalani. "Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons". In: *Mathematical Programming* 129 (2011), pp. 129–157.

[13] Francisco Barahona and Ali Ridha Mahjoub. "On the cut polytope". In: *Mathematical Programming* 36.2 (1986), pp. 157–173.

[14] John E Beasley. *Heuristic algorithms for the unconstrained binary quadratic programming problem.* Tech. rep. Working Paper, The Management School, Imperial College, London, England, 1998.

[15] Aharon Ben-Tal and Dick Den Hertog. "Hidden conic quadratic representation of some nonconvex quadratic optimization problems". In: *Mathematical Programming* 143 (2014), pp. 1–29.

[16] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[17] Bonnie Berger and Lenore Cowen. "Fast deterministic constructions of low-diameter network decompositions". In: *LCS Technical Memos* (1991).

[18] Dimitris Bertsimas and Ryan Cory-Wright. "On polyhedral and second-order cone decompositions of semidefinite optimization problems". In: *Operations Research Letters* 48.1 (2020), pp. 78–85.

[19] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671.

[20] Avinash Bhardwaj, Philipp Rostalski, and Raman Sanyal. "Deciding polyhedrality of spectrahedra". In: *SIAM Journal on Optimization* 25.3 (2015), pp. 1873–1884.

[21] Lorenz T Biegler and Ignacio E Grossmann. "Retrospective on optimization". In: *Computers & Chemical Engineering* 28.8 (2004), pp. 1169–1192.

[22] Taras Bodnar, Nestor Parolya, and Wolfgang Schmid. "On the equivalence of quadratic optimization problems commonly used in portfolio theory". In: *European Journal of Operational Research* 229.3 (2013), pp. 637–644.

[23] Endre Boros and Peter L Hammer. "The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds". In: *Annals of Operations Research* 33.3 (1991), pp. 151–180.

[24] Alexandros-Apostolos A Boulogeorgos, Josep Miquel Jornet, and Angeliki Alexiou. "Directional terahertz communication systems for 6G: Fact check". In: *IEEE Vehicular Technology Magazine* 16.4 (2021), pp. 68–77.

[25] Stephen Boyd and Jacob Mattingley. "Branch and bound methods". In: *Notes for EE364b, Stanford University* 2006 (2007), p. 07.

[26] Gábor Braun et al. "Approximation limits of linear programs (beyond hierarchies)". In: *Mathematics of Operations Research* 40.3 (2015), pp. 756–772.

[27] Andries E. Brouwer, Cohen, and Arnold Neumaier. *Distance-Regular Graphs*. Springer-Verlag, 1989.

[28] Robin Brown et al. "A copositive framework for analysis of hybrid ising-classical algorithms". In: *arXiv preprint arXiv:2207.13630* (2022).

[29] Stefan Bundfuss and Mirjam Dür. "An adaptive linear approximation algorithm for copositive programs". In: *SIAM Journal on Optimization* 20.1 (2009), pp. 30–53.

[30] Jorge Cadima and Ian T Jolliffe. "Loading and correlations in the interpretation of principle compenents". In: *Journal of Applied Statistics* 22.2 (1995), pp. 203–214.

[31] J Camacho et al. "All sparse PCA models are wrong, but some are useful. Part I: computation of scores, residuals and explained variance". In: *Chemometrics and Intelligent Laboratory Systems* 196 (2020), p. 103907.

[32] Siu On Chan et al. "Approximate constraint satisfaction requires large LP relaxations". In: *Journal of the ACM (JACM)* 63.4 (2016), pp. 1–22.

[33] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. "Integrality gaps for Sherali-Adams relaxations". In: *Proceedings of the forty-first annual ACM Symposium on Theory of Computing*. 2009, pp. 283–292.

[34] Da Qi Chen et al. "Timeliness Through Telephones: Approximating Information Freshness in Vector Clock Models". In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2023, pp. 2411–2428.

[35] Quan Chen et al. "Analysis of inter-satellite link paths for LEO mega-constellation networks". In: *IEEE Transactions on Vehicular Technology* 70.3 (2021), pp. 2743–2755.

[36] Shanzhi Chen et al. "Vision, requirements, and technology trend of 6G: How to tackle the challenges of system coverage, capacity, user data-rate and movement speed". In: *IEEE Wireless Communications* 27.2 (2020), pp. 218–228.

[37] Fan Chung and Mary Radcliffe. "On the spectra of general random graphs". In: *the Electronic Journal of Combinatorics* (2011), P215–P215.

[38] Chris Coey, Miles Lubin, and Juan Pablo Vielma. "Outer approximation with conic certificates for mixed-integer convex problems". In: *Mathematical Programming Computation* 12.2 (2020), pp. 249–293.

[39] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Vol. 271. Springer, 2014.

[40] Keith Conrad. "Simultaneous commutativity of operators". In: *University of Connecticut* (1992).

[41] Alexandre d'Aspremont et al. "A direct formulation for sparse PCA using semidefinite programming". In: *Advances in Neural Information Processing systems* 17 (2004).

[42] Charles Delorme and Svatopluk Poljak. "Laplacian eigenvalues and the maximum cut problem". In: *Mathematical Programming* 62.1-3 (1993), pp. 557–574.

[43] Zhibin Deng et al. "A branch-and-cut approach to portfolio selection with marginal risk control in a linear conic programming framework". In: *Journal of Systems Science and Systems Engineering* 22.4 (2013), pp. 385–400.

[44] Santanu S Dey et al. "Cutting plane generation through sparse principal component analysis". In: *URL http://www. optimization-online. org/DB_HTML/2021/02/8259. html* (2021).

[45] Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*. Vol. 15. Springer, 2009.

[46] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2005.

[47] Krzysztof Domino et al. "Quadratic and higher-order unconstrained binary optimization of railway rescheduling for quantum computing". In: *Quantum Information Processing* 21.9 (2022), p. 337.

[48] Mirjam Dür and Franz Rendl. "Conic optimization: a survey with special focus on copositive optimization and binary quadratic problems". In: *EURO Journal on Computational Optimization* 9 (2021), p. 100021.

[49] Marco A Duran and Ignacio E Grossmann. "An outer-approximation algorithm for a class of mixed-integer nonlinear programs". In: *Mathematical Programming* 36 (1986), pp. 307–339.

[50] Michael Elkin and Guy Kortsarz. "Combinatorial logarithmic approximation algorithm for directed telephone broadcast problem". In: *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*. 2002, pp. 438–447.

[51] Michael Elkin and Guy Kortsarz. "Sublogarithmic approximation for telephone multicast: path out of jungle." In: *SODA*. Vol. 3. 2003, pp. 76–85.

[52] Bruno Escoffier and Peter L Hammer. "Approximation of the quadratic set covering problem". In: *Discrete Optimization* 4.3-4 (2007), pp. 378–386.

[53] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". In: *arXiv preprint arXiv:1411.4028* (2014).

[54] Hamza Fawzi. "On polyhedral approximations of the positive semidefinite cone". In: *Mathematics of Operations Research* 46.4 (2021), pp. 1479–1489.

[55] Uriel Feige and Eran Ofek. "Spectral techniques applied to sparse random graphs". In: *Random Structures & Algorithms* 27.2 (2005), pp. 251–275.

[56] Roger Fletcher and Sven Leyffer. "Solving mixed integer nonlinear programs by outer approximation". In: *Mathematical Programming* 66 (1994), pp. 327–349.

[57] Santo Fortunato. "Community detection in graphs". In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[58] Joel Friedman, Jeff Kahn, and Endre Szemeredi. "On the second eigenvalue of random regular graphs". In: *Proceedings of the twenty-first annual ACM Symposium on Theory of Computing*. 1989, pp. 587–598.

[59] Alan Frieze and Michał Karoński. *Introduction to Random Graphs*. Cambridge University Press, 2016.

[60] Fabio Furini et al. "QPLIB: a library of quadratic programming instances". In: *Mathematical Programming Computation* 11 (2019), pp. 237–265.

[61] Tristan Gally and MARC E Pfetsch. "Computing restricted isometry constants via mixed-integer semidefinite programming". In: *preprint, submitted* (2016).

[62] Tristan Gally, Marc E Pfetsch, and Stefan Ulbrich. "A framework for solving mixed-integer semidefinite programs". In: *Optimization Methods and Software* 33.3 (2018), pp. 594–632.

[63] Michael Garstka, Mark Cannon, and Paul Goulart. "COSMO: A conic operator splitting method for convex conic problems". In: *Journal of Optimization Theory and Applications* 190.3 (2021), pp. 779–810.

[64] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[65] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[66] João Gouveia, Ting Kei Pong, and Mina Saee. "Inner approximating the completely positive cone via the cone of scaled diagonally dominant matrices". In: *Journal of Global Optimization* 76 (2020), pp. 383–405.

[67] Mark S Granovetter. "The strength of weak ties". In: *American Journal of Sociology* 78.6 (1973), pp. 1360–1380.

[68] Gian Giacomo Guerreschi. "Solving Quadratic Unconstrained Binary Optimization with divide-and-conquer and quantum algorithms". In: *arXiv preprint arXiv:2101.07813* (2021).

[69] András Hajnal, Eric C Milner, and Endre Szemerédi. "A cure for the telephone disease". In: *Canadian Mathematical Bulletin* 15.3 (1972), pp. 447–450.

[70] Johan Hastad. "Clique is hard to approximate within $n^{1-\epsilon}$". In: *Proceedings of 37th Conference on Foundations of Computer Science*. IEEE. 1996, pp. 627–636.

[71] Sandra M Hedetniemi, Stephen T Hedetniemi, and Arthur L Liestman. "A survey of gossiping and broadcasting in communication networks". In: *Networks* 18.4 (1988), pp. 319–349.

[72] Christoph Helmberg and Franz Rendl. "A spectral bundle method for semidefinite programming". In: *SIAM Journal on Optimization* 10.3 (2000), pp. 673–696.

[73] Christoph Helmberg, Franz Rendl, and Robert Weismantel. "A semidefinite programming approach to the quadratic knapsack problem". In: *Journal of Combinatorial Optimization* 4 (2000), pp. 197–215.

[74] Didier Henrion and Andrea Garulli. *Positive polynomials in control*. Vol. 312. Springer Science & Business Media, 2005.

[75] Hassan Hijazi, Guanglei Wang, and Carleton Coffrin. “Gravity: A mathematical modeling language for optimization and machine learning”. In: *OpenReview* (2018).

[76] Christopher Hojny and Marc E Pfetsch. “Handling symmetries in mixed-integer semidefinite programs”. In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer. 2023, pp. 69–78.

[77] Robert Holte et al. “The pinwheel: A real-time scheduling problem”. In: *Proceedings of the 22nd Hawaii International Conference of System Science*. 1989, pp. 693–702.

[78] Roger A Horn and Charles R Johnson. *Matrix Analysis*. Cambridge university press, 2012.

[79] Jennifer Iglesias et al. “Plane gossip: Approximating rumor spread in planar graphs”. In: *Latin American Symposium on Theoretical Informatics*. Springer. 2018, pp. 611–624.

[80] Jennifer Iglesias et al. “Rumors across radio, wireless, telephone”. In: *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2015.

[81] IBM ILOG. “CPLEX User’s Manual, Version 12 Release 6”. In: *International Business Machines Corporation* (2014).

[82] John NR Jeffers. “Two case studies in the application of principal component analysis”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 16.3 (1967), pp. 225–236.

[83] Jehn-Ruey Jiang and Chun-Wei Chu. “Classifying and benchmarking quantum annealing algorithms based on quadratic unconstrained binary optimization for solving NP-hard problems”. In: *IEEE Access* (2023).

[84] Wei Jiang et al. “The road towards 6G: A comprehensive survey”. In: *IEEE Open Journal of the Communications Society* 2 (2021), pp. 334–366.

[85] James E Kelley Jr. “The cutting-plane method for solving convex programs”. In: *Journal of the society for Industrial and Applied Mathematics* 8.4 (1960), pp. 703–712.

[86] Sunyoung Kim, Masakazu Kojima, and Makoto Yamashita. “Second order cone programming relaxation of a positive semidefinite constraint”. In: *Optimization Methods and Software* 18.5 (2003), pp. 535–541.

[87] Ken Kobayashi and Yuich Takano. “A branch-and-cut algorithm for solving mixed-integer semidefinite optimization problems”. In: *Computational Optimization and Applications* 75 (2020), pp. 493–513.

[88] Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. “The structure of information pathways in a social communication network”. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 435–443.

[89] Pravesh K Kothari, Raghu Meka, and Prasad Raghavendra. “Approximating rectangles by juntas and weakly-exponential lower bounds for LP relaxations of CSPs”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 590–603.

[90] Kartik Krishnan and John E Mitchell. “A unifying framework for several cutting plane methods for semidefinite programming”. In: *Optimization methods and software* 21.1 (2006), pp. 57–74.

[91] Gregory Kuperman et al. “Space-based adaptive communications node (space-bacn)”. In: *Free-Space Laser Communications XXXIV*. Vol. 11993. 2022, pp. 195–199.

[92] Leslie Lamport. “Time, clocks, and the ordering of events in a distributed system”. In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 179–196.

[93] Gert RG Lanckriet et al. “Learning the kernel matrix with semidefinite programming”. In: *Journal of Machine Learning Research* 5 (2004), pp. 27–72.

[94] Jean B Lasserre. “Global optimization with polynomials and the problem of moments”. In: *SIAM Journal on Optimization* 11.3 (2001), pp. 796–817.

[95] Eugene L Lawler and David E Wood. “Branch-and-bound methods: A survey”. In: *Operations Research* 14.4 (1966), pp. 699–719.

[96] Sven Leyffer. “Deterministic methods for mixed integer nonlinear programming”. PhD thesis. University of Dundee, Dundee, 1993.

[97] Yongchun Li and Weijun Xie. “Exact and approximation algorithms for sparse PCA”. In: *arXiv preprint arXiv:2008.12438* (2020).

[98] Marco Locatelli. “Exactness conditions for an SDP relaxation of the extended trust region problem”. In: *Optimization Letters* 10.6 (2016), pp. 1141–1151.

[99] Johan Lofberg. “YALMIP: A toolbox for modeling and optimization in MATLAB”. In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No. 04CH37508)*. IEEE, pp. 284–289.

[100] Eliane Maria Loiola et al. “A survey for the quadratic assignment problem”. In: *European Journal of Operational Research* 176.2 (2007), pp. 657–690.

[101] László Lovász. “On the Shannon capacity of a graph”. In: *IEEE Transactions on Information theory* 25.1 (1979), pp. 1–7.

[102] Miles Lubin et al. “Extended formulations in mixed-integer convex programming”. In: *Integer Programming and Combinatorial Optimization: 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings 18*. Springer. 2016, pp. 102–113.

[103] Miles Lubin et al. “Polyhedral approximation in mixed-integer convex optimization”. In: *Mathematical Programming* 172 (2018), pp. 139–168.

[104] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. “Ramanujan graphs”. In: *Combinatorica* 8.3 (1988), pp. 261–277.

[105] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. “A survey of recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics”. In: *arXiv preprint arXiv:1908.05209* (2019).

[106] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. “Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), pp. 331–360.

[107] Jérôme Malick and Frédéric Roupin. “Solving-cluster problems to optimality with semidefinite programming”. In: *Mathematical Programming* 136.2 (2012), pp. 279–300.

[108] Friedemann Mattern et al. *Virtual time and global states of distributed systems*. Univ., Department of Computer Science, 1988.

[109] Arul Mazumder and Sridhar Tayur. “Five Starter Problems: Solving Quadratic Unconstrained Binary Optimization Models on Quantum Computers”. In: *arXiv preprint arXiv:2401.08989* (2024).

[110] Frank de Meijer and Renata Sotirov. “On integrality in semidefinite programming for discrete optimization”. In: *arXiv preprint arXiv:2306.09865* (2023).

[111] Frank de Meijer and Renata Sotirov. “The Chvátal-Gomory Procedure for Integer SDPs with Applications in Combinatorial Optimization”. In: *arXiv preprint arXiv:2201.10224* (2022).

[112] Peter Merz and Kengo Katayama. “Memetic algorithms for the unconstrained binary quadratic programming problem”. In: *BioSystems* 78.1-3 (2004), pp. 99–118.

[113] Martin Middendorf. “Minimum broadcast time is NP-complete for 3-regular planar graphs and deadline 2”. In: *Information Processing Letters* 46.6 (1993), pp. 281–287.

[114] Renee Mirka and David P Williamson. “An Experimental Evaluation of Semidefinite Programming and Spectral Algorithms for Max Cut”. In: *20th International Symposium on Experimental Algorithms (SEA 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2022.

[115] Ruth Misener and Christodoulos A Floudas. "GloMIQO: Global mixed-integer quadratic optimizer". In: *Journal of Global Optimization* 57.1 (2013), pp. 3–50.

[116] John E Mitchell. "Branch-and-cut algorithms for combinatorial optimization problems". In: *Handbook of Applied Optimization* 1.1 (2002), pp. 65–77.

[117] Bojan Mohar and Svatopluk Poljak. "Eigenvalues and the max-cut problem". In: *Czechoslovak Mathematical Journal* 40.2 (1990), pp. 343–352.

[118] Ramon E Moore. "Global optimization to prescribed accuracy". In: *Computers & Mathematics with Applications* 21.6-7 (1991), pp. 25–39.

[119] Daniel Morales and Josep M Jornet. "Adapt: An adaptive directional antenna protocol for medium access control in terahertz communication networks". In: *Ad Hoc Networks* 119 (2021), p. 102540.

[120] Arkadi Nemirovski. "Interior point polynomial time methods in convex programming". In: *Lecture notes* (2004), pp. 3215–3224.

[121] Yurii Nesterov et al. *Semidefinite relaxation and nonconvex quadratic optimization*. Tech. rep. Université catholique de Louvain, Center for Operations Research and Econometrics, 1997.

[122] Afshin Nikzad and R Ravi. "Sending secrets swiftly: Approximation algorithms for generalized multicast problems". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2014, pp. 568–607.

[123] Ryan O'Donnell and Tselil Schramm. "Sherali–Adams Strikes Back". In: *CCC '19: Proceedings of the 34th Computational Complexity Conference* (2019).

[124] Brendan O'donoghue et al. "Conic optimization via operator splitting and homogeneous self-dual embedding". In: *Journal of Optimization Theory and Applications* 169 (2016), pp. 1042–1068.

[125] Gurobi Optimization et al. *Gurobi optimizer reference manual*. 2020.

[126] Manfred Padberg and Giovanni Rinaldi. "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems". In: *SIAM Review* 33.1 (1991), pp. 60–100.

[127] Jaehyun Park and Stephen Boyd. "General heuristics for nonconvex quadratically constrained quadratic programming". In: *arXiv preprint arXiv:1703.07870* (2017).

[128] Pablo A Parrilo. "Semidefinite programming relaxations for semialgebraic problems". In: *Mathematical Programming* 96.2 (2003), pp. 293–320.

[129] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

[130] Alberto Del Pia, Santanu S Dey, and Marco Molinaro. "Mixed-integer quadratic programming is in NP". In: *Mathematical Programming* 162 (2017), pp. 225–240.

[131] David Pisinger. “The quadratic knapsack problem—a survey”. In: *Discrete Applied Mathematics* 155.5 (2007), pp. 623–648.

[132] Svatopluk Poljak and Franz Rendl. “Nonpolyhedral relaxations of graph-bisection problems”. In: *SIAM Journal on Optimization* 5.3 (1995), pp. 467–487.

[133] Svatopluk Poljak and Zsolt Tuza. “The expected relative error of the polyhedral approximation of the max-cut problem”. In: *Operations Research Letters* 16.4 (1994), pp. 191–198.

[134] Abraham P Punnen. *The quadratic unconstrained binary optimization problem*. Springer, 2022.

[135] Andrea Qualizza, Pietro Belotti, and François Margot. “Linear programming relaxations of quadratically constrained quadratic programs”. In: *Mixed Integer Nonlinear Programming*. Springer, 2012, pp. 407–426.

[136] Motakuri V Ramana. “Polyhedra, spectrahedra, and semidefinite programming”. In: *Topics in semidefinite and interior-point methods, Fields Institute Communications* 18 (1997), pp. 27–38.

[137] Ramamoorthi Ravi. “Rapid rumor ramification: Approximating the minimum broadcast time”. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE. (1994), pp. 202–213.

[138] Gerhard Reinelt. “TSPLIB—A traveling salesman problem library”. In: *ORSA journal on computing* 3.4 (1991), pp. 376–384.

[139] Luis Miguel Rios and Nikolaos V Sahinidis. “Portfolio optimization for wealth-dependent risk preferences”. In: *Annals of Operations Research* 177.1 (2010), pp. 63–90.

[140] Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis. “System level concurrency control for distributed database systems”. In: *ACM Transactions on Database Systems (TODS)* 3.2 (1978), pp. 178–198.

[141] Ryan A. Rossi and Nesreen K. Ahmed. “The Network Data Repository with Interactive Graph Analytics and Visualization”. In: *AAAI*. 2015. URL: `http://networkrepository.com`.

[142] Walid Saad, Mehdi Bennis, and Mingzhe Chen. “A vision of 6G wireless systems: Applications, trends, technologies, and open research problems”. In: *IEEE network* 34.3 (2019), pp. 134–142.

[143] Nikolaos V Sahinidis. “BARON 14.3. 1: global optimization of mixed-integer nonlinear programs, user’s manual”. In: *The Optimization Firm, LLC, Pittsburgh* (2014).

[144] Hanif D Sherali and Barbara MP Fraticelli. “Enhancing RLT relaxations via a new class of semidefinite cuts”. In: *Journal of Global Optimization* 22.1 (2002), pp. 233–261.

[145] Naum Zuselevich Shor. “Dual quadratic estimates in polynomial and Boolean programming”. In: *Annals of Operations Research* 25.1 (1990), pp. 163–168.

[146] Kartik Krishnan Sivaramakrishnan. *Linear programming approaches to semidefinite programming problems*. Rensselaer Polytechnic Institute, 2002.

[147] Daniel Spielman. “Spectral graph theory”. In: *Combinatorial scientific computing* 18 (2012).

[148] Science & technology. “DARPA, lasers and an internet in orbit”. In: *The Economist* (Feb. 8, 2023). URL: `https://www.economist.com/science-and-technology/2023/02/08/darpa-lasers-and-an-internet-in-orbit` (visited on 02/08/2023).

[149] Konstantin Tikhomirov and Pierre Youssef. “The spectral gap of dense random regular graphs”. In: *The Annals of Probability* 47.1 (2019), pp. 362–419.

[150] Luca Trevisan. “Max cut and the smallest eigenvalue”. In: *SIAM Journal on Computing* 41.6 (2012), pp. 1769–1786.

[151] Lieven Vandenberghe and Stephen Boyd. “Semidefinite programming”. In: *SIAM review* 38.1 (1996), pp. 49–95.

[152] Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. “Linear programming relaxations of maxcut”. In: *Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms*. 2007, pp. 53–61.

[153] Cynthia Vinzant. “What is... a spectrahedron”. In: *Notices Amer. Math. Soc* 61.5 (2014), pp. 492–494.

[154] Andreas Wächter and Lorenz T Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106 (2006), pp. 25–57.

[155] Alex L Wang and Fatma Kılınç-Karzan. “On the tightness of SDP relaxations of QCQPs”. In: *Mathematical Programming* 193.1 (2022), pp. 33–73.

[156] Yifei Wang et al. “A decomposition augmented lagrangian method for low-rank semidefinite programming”. In: *SIAM Journal on Optimization* 33.3 (2023), pp. 1361–1390.

[157] Yuzhu Wang, Akihiro Tanaka, and Akiko Yoshise. “Polyhedral approximations of the semidefinite cone and their application”. In: *Computational Optimization and Applications* 78.3 (2021), pp. 893–913.

[158] Zaiwen Wen and Wotao Yin. “A feasible method for optimization with orthogonality constraints”. In: *Mathematical Programming* 142.1-2 (2013), pp. 397–434.

[159] Heng Yang et al. “An inexact projected gradient method with rounding and lifting by nonlinear programming for solving rank-one semidefinite relaxation of polynomial optimization”. In: *Mathematical Programming* 201.1-2 (2023), pp. 409–472.

[160] Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. “SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints”. In: *Mathematical Programming Computation* 7.3 (2015), pp. 331–366.

[161] Kazuo Yonekura and Yoshihiro Kanno. “Global optimization of robust truss topology via mixed integer semidefinite programming”. In: *Optimization and Engineering* 11 (2010), pp. 355–379.

[162] Xiaohu You et al. “Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts”. In: *Science China Information Sciences* 64 (2021), pp. 1–74.

[163] Xin-Yuan Zhao, Defeng Sun, and Kim-Chuan Toh. “A Newton-CG augmented Lagrangian method for semidefinite programming”. In: *SIAM Journal on Optimization* 20.4 (2010), pp. 1737–1765.

[164] Hui Zou, Trevor Hastie, and Robert Tibshirani. “Sparse principal component analysis”. In: *Journal of Computational and Graphical Statistics* 15.2 (2006), pp. 265–286.