Optimization over Structured Subsets of Positive Semidefinite Matrices via Column Generation

Amir Ali Ahmadi* Princeton, ORFE a_a_a@princeton.edu

Sanjeeb Dash IBM Research sanjeebd@us.ibm.com

Georgina Hall* Princeton, ORFE gh4@princeton.edu

December 16, 2015

Abstract

We develop algorithms for inner approximating the cone of positive semidefinite matrices via linear programming and second order cone programming. Starting with an initial linear algebraic approximation suggested recently by Ahmadi and Majumdar, we describe an iterative process through which our approximation is improved at every step. This is done using ideas from column generation in large-scale linear and integer programming. We then apply these techniques to approximate the sum of squares cone in a nonconvex polynomial optimization setting, and the copositive cone for a discrete optimization problem.

1 Introduction

Semidefinite programming is a powerful tool in optimization that is used in many different contexts, perhaps most notably to obtain strong bounds on discrete optimization problems or nonconvex polynomial programs. One difficulty in applying semidefinite programming is that state-of-the-art general-purpose solvers often cannot solve very large instances reliably and in a reasonable amount of time. As a result, at relatively large scales, one has to resort either to specialized solution techniques and algorithms that employ problem structure, or to easier optimization problems that lead to weaker bounds. We will focus on the latter approach in this paper.

At a high level, our goal is to not solve semidefinite programs (SDPs) to optimality, but rather replace them with cheaper conic relaxations—*linear and second order cone relaxations* to be precise—that return useful bounds quickly. Throughout the paper, we will aim to find lower bounds (for minimization problems); i.e., bounds that certify the distance of a candidate solution to optimality. Fast, good-quality lower bounds are especially important in the context of branch-and-bound schemes, where one needs to strike a delicate balance between the time spent on bounding and the time spent on branching, in order to keep the overall solution time low. Currently, in commercial integer programming solvers, almost all lower bounding approaches using branch-and-bound schemes exclusively produce linear inequalities. Even though semidefinite cuts are known to be stronger, they are often too expensive to be used even at the root node of a

*Amir Ali Ahmadi and Georgina Hall are partially supported by the Young Investigator Program award of the AFSOR.

branch-and-bound tree. Because of this, many high-performance solvers, e.g., IBM ILOG CPLEX [15] and Gurobi [1], do not even provide an SDP solver and instead solely work with LP and SOCP relaxations. Our goal in this paper is to offer some tools that exploit the power of SDP-based cuts, while staying entirely in the realm of LP and SOCP. We apply these tools to classical problems in both nonconvex polynomial optimization and discrete optimization.

Techniques that provide lower bounds on minimization problems are precisely those that certify nonnegativity of objective functions on feasible sets. To see this, note that a scalar γ is a lower bound on the minimum value of a function $f : \mathbb{R}^n \to \mathbb{R}$ on a set $K \subseteq \mathbb{R}^n$, if and only if $f(x) - \gamma \geq 0$ for all $x \in K$. As most discrete optimization problems (including those in the complexity class NP) can be written as polynomial optimization problems, the problem of certifying nonnegativity of polynomial functions, either globally or on basic semialgebraic sets, is a fundamental one. A polynomial $p(x) := p(x_1, \ldots, x_n)$ is said to be *nonnegative*, if $p(x) \geq 0$ for all $x \in \mathbb{R}^n$. Unfortunately, even in this unconstrained setting, the problem of testing nonnegativity of a polynomial p is NP-hard even when its degree equals four. This is an immediate corollary of the fact that checking if a symmetric matrix M is copositive—i.e., if $x^T Mx \geq 0, \forall x \geq 0$—is NP-hard.[1] Indeed, M is copositive if and only if the homogeneous quartic polynomial $p(x) = \sum_{i,j} M_{i,j} x_i^2 x_j^2$ is nonnegative.

Despite this computational complexity barrier, there has been great success in using sum of squares (SOS) programming [32], [22], [30] to obtain certificates of nonnegativity of polynomials in practical settings. It is known from Artin's solution [7] to Hilbert's 17th problem that a polynomial $p(x)$ is nonnegative if and only if

$$p(x) = \frac{\sum_{i=1}^t q_i^2(x)}{\sum_{i=1}^r g_i^2(x)} \Leftrightarrow (\sum_{i=1}^r g_i^2(x))p(x) = \sum_{i=1}^t q_i^2(x) \tag{1}$$

for some polynomials $q_1, \ldots, q_t, g_1, \ldots, g_r$. When p is a quadratic polynomial, then the polynomials g_i are not needed and the polynomials q_i can be assumed to be linear functions. In this case, by writing $p(x)$ as

$$p(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}^T Q \begin{pmatrix} 1 \\ x \end{pmatrix},$$

where Q is an $(n+1) \times (n+1)$ symmetric matrix, checking nonnegativity of $p(x)$ reduces to checking the nonnegativity of the eigenvalues of Q; i.e., checking if Q is positive semidefinite.

More generally, if the degrees of q_i and g_i are fixed in (1), then checking for a representation of p of the form in (1) reduces to solving an SDP, whose size depends on the dimension of x, and the degrees of p, q_i and g_i [32]. This insight has led to significant progress in certifying nonnegativity of polynomials arising in many areas. In practice, the "first level" of the SOS hierarchy is often the one used, where the polynomials g_i are left out and one simply checks if p is a sum of squares of other polynomials. In this case already, because of the numerical difficulty of solving large SDPs, the polynomials that can be certified to be nonnegative usually do not have very high degrees or very many variables. For example, finding a sum of squares certificate that a given quartic polynomial over n variables is nonnegative requires solving an SDP involving roughly $O(n^4)$ constraints and a positive semidefinite matrix variable of size $O(n^2) \times O(n^2)$. Even for a handful of or a dozen variables, the underlying semidefinite constraints prove to be expensive. Indeed, in the absence of additional structure, most examples in the literature have less than 10 variables.

[1] Weak NP-hardness of testing matrix copositivity is originally proven by Murty and Kabadi [29]; its strong NP-hardness is apparent from the work of de Klerk and Pasechnik [17].

Recently other systematic approaches to certifying nonnegativity of polynomials have been proposed which lead to less expensive optimization problems than semidefinite programming problems. In particular, Ahmadi and Majumdar [4], [3] introduce "DSOS and SDSOS" optimization techniques, which replace semidefinite programs arising in the nonnegativity certification problem by linear programs and second-order cone programs. Instead of optimizing over the cone of sum of squares polynomials, the authors optimize over two subsets which they call "diagonally dominant sum of squares" and "scaled diagonally dominant sum of squares" polynomials (see Section 2.1 for formal definitions). In the language of semidefinite programming, this translates to solving optimization problems over the cone of diagonally dominant matrices and scaled diagonally dominant matrices. These can be done by LP and SOCP respectively. The authors have had notable success with these techniques in different applications. For instance, they are able to run these relaxations for polynomial optimization problems of degree 4 in 70 variables in the order of a few minutes. They have also used their techniques to push the size limits of some SOS problems in controls; examples include stabilizing a model of a humanoid robot with 30 state variables and 14 control inputs [26], or exploring the real-time applications of SOS techniques in problems such as collision-free autonomous motion planning [5].

Motivated by these results, our goal in this paper is to start with DSOS and SDSOS techniques and improve on them. By exploiting ideas from column generation in integer/linear programming, and by appropriately interpreting the DSOS and SDSOS constraints, we produce several iterative LP and SOCP-based algorithms that improve the quality of the bounds obtained from the DSOS and SDSOS relaxations. Geometrically, this amounts to optimizing over structured subsets of sum of squares polynomials that are larger than the sets of diagonally dominant or scaled diagonally dominant sum of squares polynomials. For semidefinite programming, this is equivalent to optimizing over structured subsets of the cone of positive semidefinite matrices. An important distinction to make between the DSOS/SDSOS/SOS approaches and our approach, is that our approximations iteratively get larger in the direction of the given objective function, unlike the DSOS, SDSOS, and SOS approaches which all try to inner approximate the set of nonnegative polynomials *irrespective* of any particular direction.

The organization of the rest of the paper is as follows. In the next section, we review relevant notation, and discuss the prior literature on DSOS and SDSOS programming. In Section 3, we give a high-level overview of our column generation approaches in the context of a general SDP. In Section 4, we describe an application of our ideas to nonconvex polynomial optimization and present computational experiments with certain column generation implementations. In Section 5, we apply our column generation approach to a specific discrete optimization application, namely the stable set problem. All the work in these sections can be viewed as providing techniques to optimize over subsets of positive semidefinite matrices. We then conclude in Section 6 with some future directions, and discuss ideas for column generation which allow one to go beyond subsets of positive semidefinite matrices in the case of polynomial optimization and the stable set problem.

2 Preliminaries

Let us first introduce some notation on matrices. We denote the set of real symmetric $n \times n$ matrices by S_n. Given two matrices A and B in S_n, we denote their matrix inner product by $A \cdot B := \sum_{i,j} A_{ij}B_{ij} = \text{Trace}(AB)$. The set of symmetric matrices with nonnegative entries is denoted by N_n. A symmetric matrix A is *positive semidefinite* (psd) if $x^T Ax \geq 0$ for all $x \in \mathbb{R}^n$; this will be denoted by the standard notation

$A \succeq 0$, and our notation for the set of $n \times n$ psd matrices is P_n. A matrix A is *copositive* if $x^T Ax \geq 0$ for all $x \geq 0$. The set of copositive matrices is denoted by C_n. All three sets N_n, P_n, C_n are convex cones and we have the obvious inclusion $N_n + P_n \subseteq C_n$. This inclusion is strict if $n \geq 5$ [13], [12]. For a cone of matrices in S_n, we define its dual cone $\mathcal{K}^*$ as $\{Y \in \mathcal{S}_n : Y \cdot X \geq 0, \ \forall X \in \mathcal{K}\}$.

For a vector variable $x \in \mathbb{R}^n$ and a vector $q \in \mathbb{Z}_+^n$, let a monomial in x be denoted as $x^q := \Pi_{i=1}^n x_i^{q_i}$, and let its degree be $\sum_{i=1}^n q_i$. A polynomial is said to be *homogeneous* or a *form* if all of its monomials have the same degree. A form $p(x)$ in n variables is nonnegative if $p(x) \geq 0$ for all $x \in \mathbb{R}^n$, or equivalently for all x on the unit sphere in $\mathbb{R}^n$. The set of nonnegative (or positive semidefinite) forms in n variables and degree d is denoted by $PSD_{n,d}$. A form $p(x)$ is a *sum of squares* (sos) if it can be written as $p(x) = \sum_{i=1}^r q_i^2(x)$ for some forms $q_1, \ldots, q_r$. The set of sos forms in n variables and degree d is a cone denoted by $SOS_{n,d}$. We have the obvious inclusion $SOS_{n,d} \subseteq PSD_{n,d}$, which is strict unless $d = 2$, or $n = 2$, or $(n, d) = (3, 4)$ [20]. Let $z(x, d)$ be the vector of all monomials of degree exactly d; it is well known that a form p of degree $2d$ is sos if and only if it can be written as $p(x) = z^T(x, d)Qz(x, d)$, for some psd matrix Q [32], [31]. The size of the matrix Q, which is often called the *Gram matrix*, is $\binom{n+d-1}{d} \times \binom{n+d-1}{d}$. At the price of imposing a semidefinite constraint of this size, one obtains the very useful ability to search and optimize over the convex cone of sos forms via semidefinite programming.

2.1 DSOS and SDSOS optimization

In order to alleviate the problem of scalability posed by the SDPs arising from sum of squares programs, Ahmadi and Majumdar [4], [3][2] recently introduced similar-purpose LP and SOCP-based optimization problems that they refer to as *DSOS and SDSOS programs*. Since we will be building on these concepts, we briefly review their relevant aspects to make our paper self-contained.

The idea in [4], [3] is to replace the condition that the Gram matrix Q be positive semidefinite with stronger but cheaper conditions in the hope of obtaining more efficient inner approximations to the cone $SOS_{n,d}$. Two such conditions come from the concepts of *diagonally dominant* and *scaled diagonally dominant* matrices in linear algebra. We recall these definitions below.

Definition 2.1. *A symmetric matrix $A = (a_{ij})$ is* diagonally dominant *(dd) if $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$ for all i. We say that A is* scaled diagonally dominant *(sdd) if there exists a diagonal matrix D, with positive diagonal entries, such that DAD is diagonally dominant.*

We refer to the set of $n \times n$ dd (resp. sdd) matrices as DD_n (resp. SDD_n). The following inclusions are a consequence of Gershgorin's circle theorem:

$$DD_n \subseteq SDD_n \subseteq P_n.$$

We now use these matrices to introduce the cones of "dsos" and "sdsos" forms and some of their generalizations, which all constitute special subsets of the cone of nonnegative forms. We remark that in the interest of brevity, we do not give the original definitions of dsos and sdsos polynomials as they appear in [4] (as sos polynomials of a particular structure), but rather an equivalent characterization of them that is more useful for our purposes. The equivalence is proven in [4].

[2]The work in [4] is currently in preparation for submission; the one in [3] is a shorter conference version of [4] which has already appeared. The presentation of the current paper is meant to be self-contained.

Definition 2.2 ([3, 4]). *Recall that $z(x, d)$ denotes the vector of all monomials of degree exactly d. A form $p(x)$ of degree $2d$ is said to be*

- diagonally-dominant-sum-of-squares *(dsos) if it admits a representation as $p(x) = z^T(x, d)Qz(x, d)$, where Q is a dd matrix.*
- scaled-diagonally-dominant-sum-of-squares *(sdsos) if it admits a representation as $p(x) = z^T(x, d)Qz(x, d)$, where Q is an sdd matrix.*
- r-diagonally-dominant-sum-of-squares *(r-dsos) if there exists a positive integer r such that $p(x)(\sum_{i=1}^n x_i^2)^r$ is dsos.*
- r-scaled diagonally-dominant-sum-of-squares *(r-sdsos) if there exists a positive integer r such that $p(x)(\sum_{i=1}^n x_i^2)^r$ is sdsos.*

We denote the cone of forms in n variables and degree d that are dsos, sdsos, r-dsos, and r-sdsos by $DSOS_{n,d}$, $SDSOS_{n,d}$, $rDSOS_{n,d}$, and $rSDSOS_{n,d}$ respectively. The following inclusion relations are straightforward:

$$DSOS_{n,d} \subseteq SDSOS_{n,d} \subseteq SOS_{n,d} \subseteq PSD_{n,d},$$
$$rDSOS_{n,d} \subseteq rSDSOS_{n,d} \subseteq PSD_{n,d}, \forall r.$$

The multiplier $(\sum_{i=1}^n x_i^2)^r$ should be thought of as a special denominator in the Artin-type representation in (1). By appealing to some theorems of real algebraic geometry, it is shown in [4] that under some conditions, as the power r increases, the sets $rDSOS_{n,d}$ (and hence $rSDSOS_{n,d}$) fill up the entire cone $PSD_{n,d}$. We will mostly be concerned with the cones $DSOS_{n,d}$ and $SDSOS_{n,d}$, which correspond to the case where $r = 0$. From the point of view of optimization, our interest in all of these algebraic notions stems from the following theorem.

Theorem 2.3 ([3, 4]). *For any integer $r \geq 0$, the cone $rDSOS_{n,d}$ is polyhedral and the cone $rSDSOS_{n,d}$ has a second order cone representation. Moreover, for any fixed d and r, one can optimize a linear function over $rDSOS_{n,d}$ (resp. $rSDSOS_{n,d}$) by solving a linear program (resp. second order cone program) of size polynomial in n.*

The "LP part" of this theorem is not hard to see. The equality $p(x)(\sum_{i=1}^n x_i^2)^r = z^T(x, d)Qz(x, d)$ gives rise to linear equality constraints between the coefficients of p and the entries of the matrix Q (whose size is polynomial in n for fixed d and r). The requirement of diagonal dominance on the matrix Q can also be described by linear inequality constraints on Q. The "SOCP part" of the statement comes from the fact, shown in [4], that a matrix A is sdd if and only if it can be expressed as

$$A = \sum_{i<j} M_{2\times 2}^{ij},$$

where each $M_{2\times 2}^{ij}$ is an $n \times n$ symmetric matrix with zeros everywhere except for four entries $M_{ii}, M_{ij}, M_{ji}, M_{jj}$, which must make the 2×2 matrix $\begin{bmatrix} M_{ii} & M_{ij} \\ M_{ji} & M_{jj} \end{bmatrix}$ symmetric and positive semidefinite. These constraints are *rotated quadratic cone* constraints and can be imposed using SOCP [6], [24]:

$$M_{ii} \geq 0, \ \left|\left| \begin{pmatrix} 2M_{ij} \\ M_{ii} - M_{jj} \end{pmatrix} \right|\right| \leq M_{ii} + M_{jj}.$$

We refer to optimization problems with a linear objective posed over the convex cones $DSOS_{n,d}$, $SDSOS_{n,d}$, and $SOS_{n,d}$ as DSOS programs, SDSOS programs, and SOS programs respectively. In general, quality of approximation decreases, while scalability increases, as we go from SOS to SDSOS to DSOS programs. Depending on the size of the application at hand, one may choose one approach over the other.

3 Column generation for inner approximation of positive semidefinite cones

In this section, we describe a natural approach to apply techniques from the theory of column generation [9], [18] in large-scale integer programming to the problem of optimizing over nonnegative polynomials. Here is the rough idea: We can think of all SOS/SDSOS/DSOS approaches as ways of proving that a polynomial is nonnegative by writing it as a nonnegative linear combination of certain "atom" polynomials that are already known to be nonnegative. For SOS, these atoms are all the squares (there are infinitely many). For DSOS, there is actually a finite number of atoms corresponding to the extreme rays of the cone of diagonally dominant matrices (see Theorem 3.1 below). For SDSOS, once again we have infinitely many atoms, but with a specific structure which is amenable to an SOCP representation. Now the column generation idea is to start with a certain "cheap" subset of atoms (columns) and only add new ones—one or a limited number in each iteration—if they improve our desired objective function. This results in a sequence of monotonically improving bounds; we stop the column generation procedure when we are happy with the quality of the bound, or when we have consumed a predetermined budget on time.

In the LP case, after the addition of one or a few new atoms, one can obtain the new optimal solution from the previous solution in much less time than required to solve the new problem from scratch. However, as we show with some examples in this paper, even if one were to resolve the problems from scratch after each iteration (as we do for all of our SOCPs and some of our LPs), the overall procedure is still relatively fast. This is because in each iteration, with the introduction of a constant number k of new atoms, the problem size essentially increases only by k new variables and/or k new constraints. This is in contrast to other types of hierarchies—such as the rDSOS and rSDSOS hierarchies of Definition 2.2—that blow up in size by a factor that depends on the dimension in each iteration.

In the next two subsections we make this general idea more precise. While our focus in this section is on column generation for general SDPs, the next two sections show how the techniques are useful for approximation of SOS programs for polynomial optimization (Section 4), and copositive programs for discrete optimization (Section 5).

3.1 LP-based column generation

Consider a general SDP

$$\begin{aligned} \max_{y \in \mathbb{R}^m} \quad & b^T y \\ \text{s.t.} \quad & C - \sum_{i=1}^{m} y_i A_i \succeq 0, \end{aligned} \tag{2}$$

with $b \in \mathbb{R}^m, C, A_i \in S_n$ as input, and its dual

$$\begin{array}{ll} \min\limits_{X \in S_n} & C \cdot X \\ \text{s.t.} & A_i \cdot X = b_i,\ i = 1, \ldots, m, \\ & X \succeq 0. \end{array} \tag{3}$$

Our goal is to inner approximate the feasible set of (2) by increasingly larger polyhedral sets. We consider LPs of the form

$$\begin{array}{ll} \max\limits_{y, \alpha} & b^T y \\ \text{s.t.} & C - \sum\limits_{i=1}^{m} y_i A_i = \sum\limits_{i=1}^{t} \alpha_i B_i, \\ & \alpha_i \geq 0,\ i = 1, \ldots, t. \end{array} \tag{4}$$

Here, the matrices $B_1, \ldots, B_t \in P_n$ are some fixed set of positive semidefinite matrices (our psd "atoms"). To expand our inner approximation, we will continually add to this list of matrices. This is done by considering the dual LP

$$\begin{array}{ll} \min\limits_{X \in S_n} & C \cdot X \\ \text{s.t.} & A_i \cdot X = b_i,\ i = 1, \ldots, m, \\ & X \cdot B_i \geq 0,\ i = 1, \ldots, t, \end{array} \tag{5}$$

which in fact gives a polyhedral outer approximation (i.e., relaxation) of the spectrahedral feasible set of the SDP in (3). If the optimal solution X^* of the LP in (5) is already psd, then we are done and have found the optimal value of our SDP. If not, we can use the violation of positive semidefiniteness to extract one (or more) new psd atoms B_j. Adding such atoms to (4) is called *column generation*, and the problem of finding such atoms is called the *pricing subproblem*. (On the other hand, if one starts off with an LP of the form (5) as an approximation of (3), then the approach of adding inequalities to the LP iteratively that are violated by the current solution is called a *cutting plane* approach, and the associated problem of finding violated constraints is called the *separation subproblem*.) The simplest idea for pricing is to look at the eigenvectors v_j of X^* that correspond to negative eigenvalues. From each of them, one can generate a rank-one psd atom $B_j = v_j v_j^T$, which can be added with a new variable ("column") α_j to the primal LP in (4), and as a new constraint ("cut") to the dual LP in (5). The subproblem can then be defined as getting the most negative eigenvector, which is equivalent to minimizing the quadratic form $x^T X^* x$ over the unit sphere $\{x \mid ||x|| = 1\}$. Other possible strategies are discussed later in the paper.

This LP-based column generation idea is rather straightforward, but what does it have to do with DSOS optimization? The connection comes from the extreme-ray description of the cone of diagonally dominant matrices, which allows us to interpret a DSOS program as a particular and effective way of obtaining n^2 initial psd atoms.

Let $\mathcal{U}_{n,k}$ denote the set of vectors in $\mathbb{R}^n$ which have at most k nonzero components, each equal to ± 1, and define $U_{n,k} \subset S_n$ to be the set of matrices

$$U_{n,k} := \{uu^T : u \in \mathcal{U}_{n,k}\}.$$

For a finite set of matrices $T = \{T_1, \ldots, T_t\}$, let

$$\text{cone}(T) := \{\sum_{i=1}^{t} \alpha_i T_i : \alpha_1, \ldots, \alpha_t \geq 0\}.$$

Theorem 3.1 (Barker and Carlson [8]). $DD_n = \text{cone}(U_{n,2})$.

This theorem tells us that DD_n has exactly n^2 extreme rays. It also leads to a convenient representation of the dual cone:

$$DD_n^* = \{X \in S_n : v_i^T X v_i \geq 0, \text{ for all vectors } v_i \text{ with at most 2 nonzero components, each equal to } \pm 1\}.$$

Throughout the paper, we will be initializing our LPs with the DSOS bound; i.e., our initial set of psd atoms B_i will be the n^2 rank-one matrices $u_i u_i^T$ in $U_{n,2}$. This is because this bound is often cheap and effective. Moreover, it guarantees feasibility of our initial LPs (see Theorems 4.1 and 5.1), which is crucial for starting column generation. One also readily sees that the DSOS bound can be improved if we were to instead optimize over the cone $U_{n,3}$, which has n^3 atoms. However, in settings that we are interested in, we cannot afford to include all these atoms; instead, we will have pricing subproblems that try to pick a useful subset (see Section 4).

3.2 SOCP-based column generation

In a similar vein, we present an SOCP-based column generation algorithm that in our experience often does much better than the LP-based approach. The idea is once again to optimize over structured subsets of the positive semidefinite cone that are SOCP representable and that are larger than the set SDD_n of scaled diagonally dominant matrices. This will be achieved by working with the following SOCP

$$\begin{aligned} \max_{y \in \mathbb{R}^m, \Lambda_i \in S_2} \quad & b^T y \\ \text{s.t.} \quad & C - \sum_{i=1}^{m} y_i A_i = \sum_{i=1}^{t} V_i \Lambda_i V_i^T, \\ & \Lambda_i \succeq 0, \ i = 1, \ldots, t. \end{aligned} \tag{6}$$

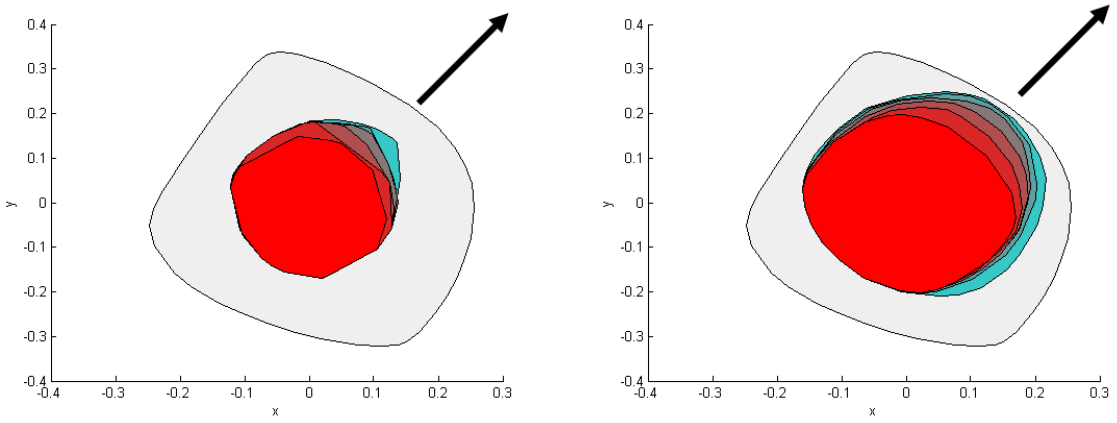
Here, the decision matrices Λ_i are 2×2 and the positive semidefiniteness constraints on them can be imposed via rotated quadratic cone constraints as explained in Section 2.1. The $n \times 2$ matrices V_i are fixed for all $i = 1, \ldots, t$. Note that this is a direct generalization of the LP in (4), in the case where the atoms B_i are rank-one. To generate a new SOCP atom, we work with the dual of (6):

$$\begin{aligned} \min_{X \in S_n} \quad & C \cdot X \\ \text{s.t.} \quad & A_i \cdot X = b_i, \ i = 1, \ldots, m, \\ & V_i^T X V_i \succeq 0, \ i = 1, \ldots, t. \end{aligned} \tag{7}$$

Once again, if the optimal solution X^* is psd, we have solved our SDP exactly; if not, we can use X^* to produce new SOCP-based cuts. For example, by placing the two eigenvectors of X^* corresponding to its two most negative eigenvalues as the columns of an $n \times 2$ matrix V_{t+1}, we have produced a new useful

atom. (Of course, we can also choose to add more pairs of eigenvectors and add multiple atoms.) As in the LP case, by construction, our bound can only improve in every iteration.

We will always be initializing our SOCP iterations with the SDSOS bound. It is not hard to see that this corresponds to the case where we have $\binom{n}{2}$ initial $n \times 2$ atoms V_i, which have zeros everywhere, except for a 1 in the first column in position j and a 1 in the second column in position $k \neq j$. We denote the set of all such $n \times 2$ matrices by $\mathcal{V}_{n,2}$



(a) LP starting with DSOS and adding 5 atoms.

(b) SOCP starting with SDSOS and adding 5 atoms.

Figure 1: LP and SOCP-based column generation for inner approximation of a spectrahedron.

Figure 1 shows an example of both the LP and SOCP column generation procedures. We produced two 10×10 random symmetric matrices E and F. The outer most set is the feasible set of an SDP with the constraint $I + xE + yF \succeq 0$. (Here, I is the 10×10 identity matrix.) The SDP wishes to maximize $x + y$ over this set. The innermost set in Figure 1(a) is the polyhedral set where $I + xE + yF$ is dd. The innermost set in Figure 1(b) is the SOCP-representable set where $I + xE + yF$ is sdd. In both cases, we do 5 iterations of column generation that expand these sets by introducing one new atom at a time. These atoms come from the most negative eigenvector (resp. the two most negative eigenvectors) of the dual optimal solution as explained above. Note that in both cases, we are growing our approximation of the positive semidefinite cone in the direction that we care about (the northeast). This is in contrast to algebraic hierarchies based on "positive multipliers" (see the rDSOS and rSDSOS hierarchies in Definition 2.2 for example), which completely ignore the objective function.

4 Nonconvex polynomial optimization

In this section, we apply the ideas described in the previous section to sum of squares algorithms for nonconvex polynomial optimization. In particular, we consider the NP-hard problem of minimizing a form (of degree ≥ 4) on the sphere. Recall that $z(x, d)$ is the vector of all monomials in n variables with degree d. Let $p(x)$ be a form with n variables and even degree $2d$, and let coef(p) be the vector of its coefficients with the monomial ordering given by $z(x, 2d)$. Thus $p(x)$ can be viewed as coef$(p)^T z(x, 2d)$. Let $s(x) := (\sum_{i=1}^n x_i^2)^d$. With this notation, the problem of minimizing a form p on the unit sphere can be written as

$$\begin{aligned} \max_{\lambda} \quad & \lambda \\ \text{s.t.} \quad & p(x) - \lambda s(x) \geq 0, \forall x \in \mathbb{R}^n. \end{aligned} \tag{8}$$

With the SOS programming approach, the following SDP is solved to get the largest scalar λ and an SOS certificate proving that $p(x) - \lambda s(x)$ is nonnegative:

$$\begin{aligned} \max_{\lambda, Y} \quad & \lambda \\ \text{s.t.} \quad & p(x) - \lambda s(x) = z^T(x,d) Y z(x,d) \\ & Y \succeq 0. \end{aligned} \tag{9}$$

The sum of squares certificate is directly read from an eigenvalue decomposition of the solution Y to the SDP above and has the form

$$p(x) - \lambda s(x) \geq \sum_i (z^T(x,d) u_i)^2,$$

where $Y = \sum_i u_i u_i^T$. Since all sos polynomials are nonnegative, the optimal value of the SDP in (9) is a lower bound to the optimal value of the optimization problem in (8). Unfortunately, before solving the SDP, we do not have access to the vectors u_i in the decomposition of the optimal matrix Y. However, the fact that such vectors exist hints at how we should go about replacing P_n by a polyhedral restriction in (9): If the constraint $Y \succeq 0$ is changed to

$$Y = \sum_{u \in \mathcal{U}} \alpha_u u u^T, \alpha_u \geq 0, \tag{10}$$

where $\mathcal{U}$ is a finite set, then (9) becomes an LP. This is one interpretation of Ahmadi and Majumdar's work in [3, 4] where they replace P_n by DD_n. Indeed, this is equivalent to taking $\mathcal{U} = \mathcal{U}_{n,2}$ in (10), as shown in Theorem 3.1. We are interested in extending their results by replacing P_n by larger restrictions than DD_n. A natural candidate for example would be obtained by changing $\mathcal{U}_{n,2}$ to $\mathcal{U}_{n,3}$. However, although $\mathcal{U}_{n,3}$ is finite, it contains a very large set of vectors even for small values of n and d. For instance, when $n = 30$ and $d = 4$, $\mathcal{U}_{n,3}$ has over 66 million elements. Therefore we use column generation ideas to iteratively expand $\mathcal{U}$ in a manageable fashion. To initialize our procedure, we need to start with good enough atoms to have a feasible LP. The following result guarantees that replacing $Y \succeq 0$ with $Y \in DD_n$ always yields an initial feasible LP in the setting that we are interested in.

Theorem 4.1. *For any form p of degree $2d$, there exists $\lambda \in \mathbb{R}$ such that $p(x) - \lambda(\sum_{i=1}^n x_i^2)^d$ is dsos.*

Proof. As before, let $s(x) = (\sum_{i=1}^n x_i^2)^d$. We observe that the form $s(x)$ is strictly in the interior of $DSOS_{n,2d}$. Indeed, by expanding out the expression we see that we can write $s(x)$ as $z^T(x,d) Q z(x,d)$, where Q is a diagonal matrix with all diagonal entries positive. So Q is in the interior of $DD_{\binom{n+d-1}{d}}$, and hence $s(x)$ is in the interior of $DSOS_{n,2d}$. This entails that for $\alpha > 0$ small enough, the form

$$(1-\alpha)s(x) + \alpha p(x)$$

will be dsos. Since $DSOS_{n,2d}$ is a cone, the form

$$\frac{(1-\alpha)}{\alpha} s(x) + p(x)$$

will also be dsos. By taking λ to be smaller than $-\frac{1-\alpha}{\alpha}$, the claim is established. □

As $DD_n \subseteq SDD_n$, the theorem above implies that replacing $Y \succeq 0$ with $Y \in SDD_n$ also yields an initial feasible SOCP. Motivated in part by this theorem, we will always start our LP-based iterative process with the restriction that $Y \in DD_n$. Let us now explain how we improve on this approximation via column generation.

Suppose we have a set $\mathcal{U}$ of vectors in $\mathbb{R}^n$, whose outerproducts form all of the rank-one psd atoms that we want to consider. This set could be finite but very large, or even infinite. For our purposes $\mathcal{U}$ always includes $\mathcal{U}_{n,2}$, as we initialize our algorithm with the dsos relaxation. Let us consider first the case where $\mathcal{U}$ is finite: $\mathcal{U} = \{u_1, \ldots, u_t\}$. Then the problem that we are interested in solving is

$$\begin{aligned} \max_{\lambda, \alpha_j} \quad & \lambda \\ \text{s.t.} \quad & p(x) - \lambda s(x) = z^T(x,d) Y z(x,d) \\ & Y = \sum_{j=1}^{t} \alpha_j u_j u_j^T, \ \alpha_j \geq 0 \text{ for } j = 1, \ldots, t. \end{aligned}$$

Suppose $z(x, 2d)$ has m monomials and let the ith monomial in $p(x)$ have coefficient b_i, i.e., coef$(p) = (b_1, \ldots, b_m)^T$. Also let s_i be the ith entry in coef$(s(x))$. We rewrite the previous problem as

$$\begin{aligned} \max_{\lambda, \alpha_j} \quad & \lambda \\ \text{s.t.} \quad & A_i \cdot Y + \lambda s_i = b_i \text{ for } i = 1, \ldots, m \\ & Y = \sum_{j=1}^{t} \alpha_j u_j u_j^T, \ \alpha_j \geq 0 \text{ for } j = 1, \ldots, t. \end{aligned}$$

where A_i is a matrix that collects entries of Y that contribute to a coefficient corresponding to monomial i in $z(x, 2d)$, when $z^T(x,d) Y z(x,d)$ is expanded out. The above is equivalent to

$$\begin{aligned} \max_{\lambda, \alpha_j} \quad & \lambda \\ \text{s.t.} \quad & \sum_j \alpha_j (A_i \cdot u_j u_j^T) + \lambda s_i = b_i \ \text{ for } i = 1, \ldots, m. \\ & \alpha_j \geq 0 \text{ for } j = 1, \ldots, t. \end{aligned} \tag{11}$$

The dual problem is

$$\begin{aligned} \min_{\mu} \quad & \sum_{i=1}^{m} \mu_i b_i \\ \text{s.t.} \quad & (\sum_{i=1}^{m} \mu_i A_i) \cdot u_j u_j^T \geq 0, \ j = 1, \ldots, t \\ & \sum_{i=1}^{m} \mu_i s_i = 1. \end{aligned}$$

In the column generation framework, suppose we consider only a subset of the primal LP variables corresponding to the matrices $u_1 u_1^T, \ldots, u_k u_k^T$ for some $k < t$ (call this the reduced primal problem). Let

$(\bar{\alpha}_1, \ldots, \bar{\alpha}_k)$ stand for an optimal solution of the reduced primal problem and let $\bar{\mu} = (\bar{\mu}_1, \ldots, \bar{\mu}_m)$ stand for an optimal dual solution. If we have

$$(\sum_{i=1}^{m} \bar{\mu}_i A_i) \cdot u_j u_j^T \geq 0 \text{ for } j = k+1, \ldots, t, \tag{12}$$

then $\bar{\mu}$ is an optimal dual solution for the original larger primal problem with columns $1, \ldots, t$. In other words, if we simply set $\alpha_{k+1} = \cdots = \alpha_t = 0$, then the solution of the reduced primal problem becomes a solution of the original primal problem. On the other hand, if (12) is not true, then suppose the condition is violated for some $u_l u_l^T$. We can augment the reduced primal problem by adding the variable α_l, and repeat this process.

Let $B = \sum_{i=1}^{m} \bar{\mu}_i A_i$. We can test if (12) is false by solving the *pricing subproblem*:

$$\min_{u \in \mathcal{U}} u^T B u. \tag{13}$$

If $u^T B u < 0$, then there is an element u in $\mathcal{U}$ such that the matrix uu^T violates the dual constraint written in (12). Problem (13) may or may not be easy to solve depending on the set $\mathcal{U}$. For example, an ambitious column generation strategy to improve on dsos (i.e., $\mathcal{U} = \mathcal{U}_{n,2}$), would be to take $\mathcal{U} = \mathcal{U}_{n,n}$; i.e., the set all vectors in $\mathbb{R}^n$ consisting of zeros, ones, and minus ones. In this case, the pricing problem (13) becomes

$$\min_{u \in \{0, \pm 1\}^n} u^T B u.$$

Unfortunately, the above problem generalizes the quadratic unconstrained boolean optimization problem (QUBO) and is NP-hard. Nevertheless, there are good heuristics for this problem (see e.g., [11],[16]) that can be used to find near optimal solutions very fast. While we did not pursue this pricing subproblem, we did consider optimizing over $\mathcal{U}_{n,3}$. We refer to the vectors in $\mathcal{U}_{n,3}$ as "triples" for obvious reasons and generally refer to the process of adding atoms drawn from $U_{n,3}$ as optimizing over "triples".

Even though one can theoretically solve (13) with $\mathcal{U} = \mathcal{U}_{n,3}$ in polynomial time by simple enumeration of n^3 elements, this is very impractical. Our simple implementation is a partial enumeration and is implemented as follows. We iterate through the triples (in a fixed order), and test to see whether the condition $u^T B u \geq 0$ is violated by a given triple u, and collect such violating triples in a list. We terminate the iteration when we collect a fixed number of violating triples (say t_1). We then sort the violating triples by increasing values of $u^T B u$ (remember, these values are all negative for the violating triples) and select the t_2 most violated triples (or fewer if less than t_2 are violated overall) and add them to our current set of atoms. In a subsequent iteration we start off enumerating triples right after the last triple enumerated in the current iteration so that we do not repeatedly scan only the same subset of triples. Although our implementation is somewhat straightforward and can be obviously improved, we are able to demonstrate that optimizing over triples improves over the best bounds obtained by Ahmadi and Majumdar in a similar amount of time (see Section 4.2).

We can also have pricing subproblems where the set $\mathcal{U}$ is infinite. Consider e.g. the case $\mathcal{U} = \mathbb{R}^n$ in (13). In this case, if there is a feasible solution with a negative objective value, then the problem is clearly unbounded below. Hence, we look for a solution with the smallest value of "violation" of the dual constraint divided by the norm of the violating matrix. In other words, we want the expression $u^T B u / \text{norm}(uu^T)$ to be as small as possible, where norm is the Euclidean norm of the vector consisting of all entries of uu^T. This is the same as minimizing $u^T B u / ||u||^2$. The eigenvector corresponding to the smallest eigenvalue yields

such a minimizing solution. This is the motivation behind the strategy described in the previous section for our LP column generation scheme. In this case, we can use a similar strategy for our SOCP column generation scheme. We replace $Y \succeq 0$ by $Y \in SDD_n$ in (9) and iteratively expand SDD_n by using the "two most negative eigenvector technique" described in Section 3.2.

4.1 Experiments with a 10-variable quartic

We illustrate the behaviour of these different strategies on an example. Let $p(x)$ be a degree-four form defined on 10 variables, where the components of coef(p) are drawn independently at random from the normal distribution $\mathcal{N}(0,1)$. Thus $d = 2$ and $n = 10$, and the form $p(x)$ is 'fully dense' in the sense that coef(p) has essentially all nonzero components. In Figure 2, we show how the lower bound on the optimal value of $p(x)$ over the unit sphere changes per iteration for different methods. The x-axis shows the number of iterations of the column generation algorithm, i.e., the number of times columns are added and the LP (or SOCP) is resolved. The y-axis shows the lower bound obtained from each LP or SOCP. Each curve represents one way of adding columns. The three horizontal lines (from top to bottom) represent, respectively, the SDP bound, the 1SDSOS bound and the 1DSOS bound. The curve DSOS$_k$ gives the bound obtained by solving LPs, where the first LP has $Y \in DD_n$ and subsequent columns are generated from a single eigenvector corresponding to the most negative eigenvalue of the dual optimal solution as described in Section 3.1. The LP triples curve also corresponds to an LP sequence, but this time the columns that are added are taken from $U_{n,3}$ and are more than one in each iteration (see the next subsection). This bound saturates when constraints coming from all elements of $U_{n,3}$ are satisfied. Finally, the curve SDSOS$_k$ gives the bound obtained by SOCP-based column generation as explained just above.

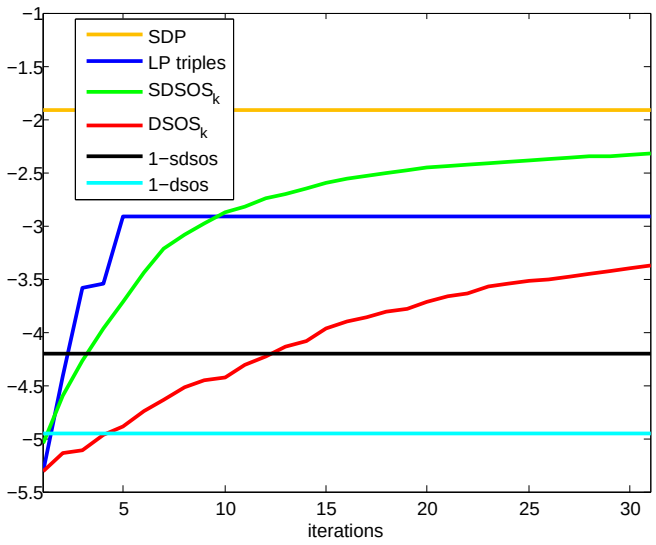


Figure 2: Lower bounds for a polynomial of degree 4 in 10 variables obtained via LP and SOCP based column generation

4.2 Larger computational experiments

In this section, we consider larger problem instances ranging from 15 variables to 40 variables, and we only apply our "triples" column generation strategy. Our problem instances are again fully dense and generated in exactly the same way as the $n = 10$ example of the previous subsection.

To solve the triples pricing subproblem with our partial enumeration strategy, we set t_1 to 300,000 and t_2 to 5000. Thus in each iteration, we find up to 300,000 violated triples, and add up to 5000 of them. In other words, we augment our LP by up to 5000 columns in each iteration. This is somewhat unusual as in practice at most a few dozen columns are added in each iteration. The logic for this is that primal simplex is very fast in reoptimizing an LP when a small number of additional columns are added to an LP whose optimal basis is known. However, in our context, we observed that the associated LPs are very hard for the simplex routines inside our LP solver (CPLEX 12.4) and take much more time than CPLEX's interior point solver. We therefore use CPLEX's interior point ("barrier") solver not only for the initial LP but for subsequent LPs after adding columns. Because interior point solvers do not benefit significantly from warm starts, each LP takes a similar amount of time to solve as the initial LP, and therefore it makes sense to add a large number of columns in each iteration to amortize the time for each expensive solve over many columns.

Table 1 is taken from the work of Ahmadi and Majumdar [4], where they report lower bounds on the minimum value of fourth-degree forms on the unit sphere obtained using different methods, and the respective computing times (in seconds).

	n=15		n=20		n=25		n=30		n=40	
	bd	t(s)	bd	t(s)	bd	t(s)	bd	t(s)	bd	t(s)
DSOS	-10.96	0.38	-18.012	0.74	-26.45	15.51	-36.85	7.88	-62.30	10.68
SDSOS	-10.43	0.53	-17.33	1.06	-25.79	8.72	-36.04	5.65	-61.25	18.66
1-DSOS	-9.22	6.26	-15.72	37.98	-23.58	369.08	NA	NA	NA	NA
1-SDSOS	-8.97	14.39	-15.29	82.30	-23.14	538.54	NA	NA	NA	NA
SOS	-3.26	5.60	-3.58	82.22	-3.71	1068.66	NA	NA	NA	NA

Table 1: Comparison of optimal values in [4] for lower bounding a quartic form on the sphere for varying dimension, along with run times (in seconds). These results are obtained on a 3.4 GHz Windows computer with 16 GB of memory.

In Table 2, we give our bounds for the exact same problem instances. We report two bounds, obtained at two different times (if applicable). In the first case (columns labeled C1), the time taken by 1-SDSOS in Table 1 is taken as a limit, and we report the bound from the last column generation iteration occuring before this limit; the 1-SDSOS bound is the best non-SDP bound reported in the experiments of Ahmadi and Majumdar. In the columns labeled as C2, we take 600 seconds as a limit and report the last bound obtained before this limit. In a couple of instances ($n = 15$ and $n = 20$), our column generation algorithm terminates before the 600 second limit, and we report the termination time in this case.

	n=15		n=20		n=25		n=30		n=40	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
time (sec)	10.96	31.19	70.70	471.39	508.63	600	N/A	600	N/A	600
triples bound	-6.20	-5.57	-12.38	-9.02	-20.08	-20.08	N/A	-32.38	N/A	-35.14

Table 2: Lower bounds on the optimal value of a form on the sphere for varying degrees of polynomials using Triples on a 2.33 GHz Linux machine with 32 GB of memory.

We observe that in the same amount of time (and even on a slightly slower machine), we are able to consistently beat the 1SDSOS bound, which is the strongest non-SDP bound produced in [4].

5 Inner approximations of copositive programs and the maximum stable set problem

Semidefinite programming has been used extensively for approximation of NP-hard combinatorial optimization problems. One such example is finding the *stability number* of a graph. A stable set (or independent set) of a graph $G = (V, E)$ is a set of nodes of G, no two of which are adjacent. The size of the largest stable set of a graph G is called the stability number (or independent set number) of G and is denoted by $\alpha(G)$. Throughout, $G(V, E)$ is taken to be an undirected, unweighted graph on n nodes. It is known that the problem of testing if $\alpha(G)$ is greater than a given integer k is NP-hard [21]. Furthermore, the stability number cannot be approximated within a factor $n^{1-\epsilon}$ for any $\epsilon > 0$ unless P=NP [19]. The natural integer programming formulation of this problem is given by

$$\begin{aligned} \alpha(G) = & \max_{x_i} \sum_{i=1}^{n} x_i \\ & \text{s.t. } x_i + x_j \leq 1, \text{ if} (i,j) \in E \\ & x_i \in \{0,1\}. \end{aligned} \tag{14}$$

Although this optimization problem is intractable, there are several computationally-tractable relaxations that provide upper bounds on the stability number of a graph. For example, the obvious LP relaxation of (14) can be obtained by relaxing the constraint $x_i \in \{0, 1\}$ to $x_i \in [0, 1]$:

$$\begin{aligned} LP(G) = & \max_{x_i} \sum_{i} x_i \\ & \text{s.t. } x_i + x_j \leq 1, \text{ if} (i,j) \in E \\ & x_i \in [0,1]. \end{aligned} \tag{15}$$

This bound can be improved upon by adding the so-called *clique inequalities* to the LP. For instance, one can add 3-clique inequalities to (15) by requiring that $x_i + x_j + x_k \leq 1$ for all (i, j, k) forming a triangle. This set of constraints is denoted by C_3. More generally, one can define C_k to be the set of k-clique inequalities given by $x_{i_1} + x_{i_2} + \ldots + x_{i_k} \leq 1$ when $(i_1, i_2, \ldots, i_k)$ form a clique. This leads to a hierarchy of LP

relaxations:

$$\begin{aligned} LP_k(G) = &\max \sum_i x_i \\ & x_i \in [0,1] \\ & C_2, \ldots, C_k \text{ are satisfied.} \end{aligned} \tag{16}$$

Notice that for $k = 2$, this simply corresponds to (15), in other words, $LP_2(G) = LP(G)$.

There are also semidefinite programming (SDP) relaxations that provide upper bounds to the stability number. The most famous one perhaps is the Lovász theta number $\vartheta(G)$ [25], which is defined as the optimal value of the following SDP:

$$\begin{aligned} \vartheta(G) := &\max_X J \cdot X \\ & \text{s.t. } I \cdot X = 1, \\ & X_{i,j} = 0, \forall (i,j) \in E \\ & X \in P_n. \end{aligned} \tag{17}$$

Here J is the all-ones matrix and I is the identity matrix of size n. The Lovász theta number is known to always give at least as good of an upper bound as the LP in (15), even with the addition of clique inequalities of all sizes (there are exponentially many); see, e.g., [23, Section 6.5.2] for a proof. In other words,

$$\vartheta(G) \leq LP_k(G), \forall k.$$

An alternative SDP relaxation for stable set is due to de Klerk and Pasechnik. In [17], they show that the stability number can be obtained through a conic linear program over the set of copositive matrices. Namely,

$$\begin{aligned} \alpha(G) = &\min_\lambda \lambda \\ & \text{s.t. } \lambda(I + A) - J \in C_n, \end{aligned} \tag{18}$$

where A is the adjacency matrix of G. Replacing C_n by the restriction $P_n + N_n$, we obtain the aforementioned relaxation through the following SDP

$$\begin{aligned} SDP(G) := &\min_{\lambda, X} \lambda \\ & \text{s.t. } \lambda(I + A) - J \geq X \\ & X \in P_n. \end{aligned} \tag{19}$$

This latter SDP is more expensive to solve than the Lovász SDP (17), but the bound that it obtains is always at least as good (and sometimes strictly better). A proof of this statement is given in [17, Lemma 5.2], where it is shown that (19) is an equivalent formulation of an SDP of Schrijver [36], which produces stronger upper bounds than (17).

Another reason for the interest in the copositive approach is that it allows for well-known SDP and LP hierarchies—developed respectively by Parrilo [31, Section 5] and de Klerk and Pasechnik [17]—that produce a sequence of improving bounds on the stability number. In fact, by appealing to Positivstellensatz results of Pólya [33], and Powers and Reznick [34], de Klerk and Pasechnik show that their LP hierarchy produces the exact stability number in $\alpha^2(G)$ number of steps [17, Theorem 4.1]. This immediately implies

the same result for stronger hierarchies, such as the SDP hierarchy of Parrilo [31], or the rDSOS and rSDSOS hierarchies of Ahmadi and Majumdar [4].

One notable difficulty with the use of copositivity-based SDP relaxations such as (19) in applications is scalibility. For example, it takes more than 5 hours to solve (19) when the input is a randomly generated Erdós-Renyi graph with 300 nodes and edge probability $p = 0.8$. [3] Hence, instead of using (19), we will solve a sequence of LPs/SOCPs generated in an iterative fashion. These easier optimization problems will provide upper bounds on the stability number in a more reasonable amount of time, though they will be weaker than the ones obtained via (19).

We will derive both our LP and SOCP sequences from formulation (18) of the stability number. To obtain the first LP in the sequence, we replace C_n by $DD_n + N_n$ (instead of replacing C_n by $P_n + N_n$ as was done in (19)) and get

$$\begin{aligned} DSOS_1(G) := \min_{\lambda, X} \ & \lambda \\ & \text{s.t. } \lambda(I + A) - J \geq X \\ & X \in DD_n. \end{aligned} \tag{20}$$

This is an LP whose optimal value is a valid upperbound on the stability number as $DD_n \subseteq P_n$.

Theorem 5.1. *The LP in (20) is always feasible.*

Proof. We need to show that for any $n \times n$ adjacency matrix A, there exists a diagonally dominant matrix D, a nonnegative matrix N, and a scalar λ such that

$$\lambda(I + A) - J = D + N. \tag{21}$$

Notice first that $\lambda(I + A) - J$ is a matrix with $\lambda - 1$ on the diagonal and at entry (i, j), if (i, j) is an edge in the graph, and with -1 at entry (i, j) if (i, j) is not an edge in the graph. If we denote by d_i the degree of node i, then let us take $\lambda = n - \min_i d_i + 1$ and D a matrix with diagonal entries $\lambda - 1$ and off-diagonal entries equal to 0 if there is an edge, and -1 if not. This matrix is diagonally dominant as there are at most $n - \min_i d_i$ minus ones on each row. Furthermore, if we take N to be a matrix with $\lambda - 1$ at the entries (i, j) where (i, j) is an edge in the graph, then (21) is satisfied and $N \geq 0$. □

Feasibility of this LP is important for us as it allows us to initiate column generation. By contrast, if we were to replace the diagonal dominance constraint by a diagonal constraint for example, the LP could fail to be feasible. This fact has been observed by de Klerk and Pasechnik in [17] and Bomze and de Klerk in [10].

To generate the next LP in the sequence via column generation, we think of the extreme-ray description of the set of diagonally dominant matrices as explained in Section 3. Theorem 3.1 tells us that these are given by the matrices in $U_{n,2}$ and so we can rewrite (20) as

$$\begin{aligned} DSOS_1(G) := \min_{\lambda, \alpha_i} \ & \lambda \\ & \text{s.t. } \lambda(I + A) - J \geq X \\ & X = \sum_{u_i u_i^T \in U_{n,2}} \alpha_i u_i u_i^T, \\ & \alpha_i \geq 0, \ i = 1, \ldots, n^2. \end{aligned} \tag{22}$$

[3] The solver in this case is MOSEK [2] and the machine used has 3.4GHz speed and 16GB RAM; see Table 4 for more results. The solution time with the popular SDP solver SeDuMi [37] e.g. would be several times larger.

The column generation procedure aims to add new matrix atoms to the existing set $U_{n,2}$ in such a way that the current bound $DSOS_1$ improves. There are numerous ways of choosing these atoms. We focus first on the cutting plane approach based on eigenvectors. The dual of (22) is the LP

$$\begin{aligned} DSOS_1(G) := \max_X \; & J \cdot X \\ & \text{s.t. } (A+I) \cdot X = 1 \\ & X \geq 0 \\ & (u_i u_i^T) \cdot X \geq 0, \forall u_i u_i^T \in U_{n,2}. \end{aligned} \tag{23}$$

If our optimal solution X^* to (23) is positive semidefinite, then we are obtaining the best bound we can possibly produce, which is the SDP bound of (19). If this is not the case however, we pick our atom matrix to be the outer product uu^T of the eigenvector u corresponding to the most negative eigenvalue of X^*. The optimal value of the LP

$$\begin{aligned} DSOS_2(G) := \max_X \; & J \cdot X \\ & \text{s.t. } (A+I) \cdot X = 1 \\ & X \geq 0 \\ & (u_i u_i^T) \cdot X \geq 0, \forall u_i u_i^T \in U_{n,2} \\ & (uu^T) \cdot X \geq 0 \end{aligned} \tag{24}$$

that we derive is guaranteed to be no worse than $DSOS_1$ as the feasible set of (24) is smaller than the feasible set of (23). Under mild nondegeneracy assumptions (satisfied, e.g., by uniqueness of the optimal solution to (23)), the new bound will be strictly better. By reiterating the same process, we create a sequence of LPs whose optimal values $DSOS_1, DSOS_2, \ldots$ are a nonincreasing sequence of upper bounds on the stability number.

Generating the sequence of SOCPs is done in an analogous way. Instead of replacing the constraint $X \in P_n$ in (19) by $X \in DD_n$, we replace it by $X \in SDD_n$ and get

$$\begin{aligned} SDSOS_1(G) := \min_{\lambda, X} \; & \lambda \\ & \text{s.t. } \lambda(I+A) - J \geq X \\ & X \in SDD_n. \end{aligned} \tag{25}$$

Once again, we need to reformulate the problem in such a way that the set of scaled diagonally dominant matrices is described as some combination of psd "atom" matrices. In this case, we can write any matrix $X \in SDD_n$ as

$$X = \sum_{V_i \in \mathcal{V}_{n,2}} V_i \begin{pmatrix} a_i^1 & a_i^2 \\ a_i^2 & a_i^3 \end{pmatrix} V_i^T,$$

where a_i^1, a_i^2, a_i^3 are variables making the 2×2 matrix psd, and the V_i's are our atoms. Recall from Section 3 that the set $\mathcal{V}_{n,2}$ consists of all $n \times 2$ matrices which have zeros everywhere, except for a 1 in the first column in position j and a 1 in the second column in position $k \neq j$. This gives rise to an equivalent formulation of

(25):

$$\begin{aligned} SDSOS_1(G) := \min_{\lambda, a_i^j} \ & \lambda \\ & \text{s.t. } \lambda(I+A) - J \geq X \\ & X = \sum_{V_i \in \mathcal{V}_{n,2}} V_i \begin{pmatrix} a_i^1 & a_i^2 \\ a_i^2 & a_i^3 \end{pmatrix} V_i^T \\ & \begin{pmatrix} a_i^1 & a_i^2 \\ a_i^2 & a_i^3 \end{pmatrix} \succeq 0, \ i = 1, \ldots, \binom{n}{2}. \end{aligned} \tag{26}$$

Just like the LP case, we now want to generate one (or more) $n \times 2$ matrix V to add to the set $\{V_i\}_i$ so that the bound $SDSOS_1$ improves. We do this again by using a cutting plane approach originating from the dual of (26):

$$\begin{aligned} SDSOS_1(G) := \max_{X} \ & J \cdot X \\ & \text{s.t. } (A+I) \cdot X = 1 \\ & X \geq 0 \\ & V_i^T \cdot X V_i \succeq 0, \ i = 1, \ldots, \binom{n}{2}. \end{aligned} \tag{27}$$

Note that strong duality holds between this primal-dual pair as it is easy to check that both problems are strictly feasible. We then take our new atom to be

$$V = (w_1 \ w_2),$$

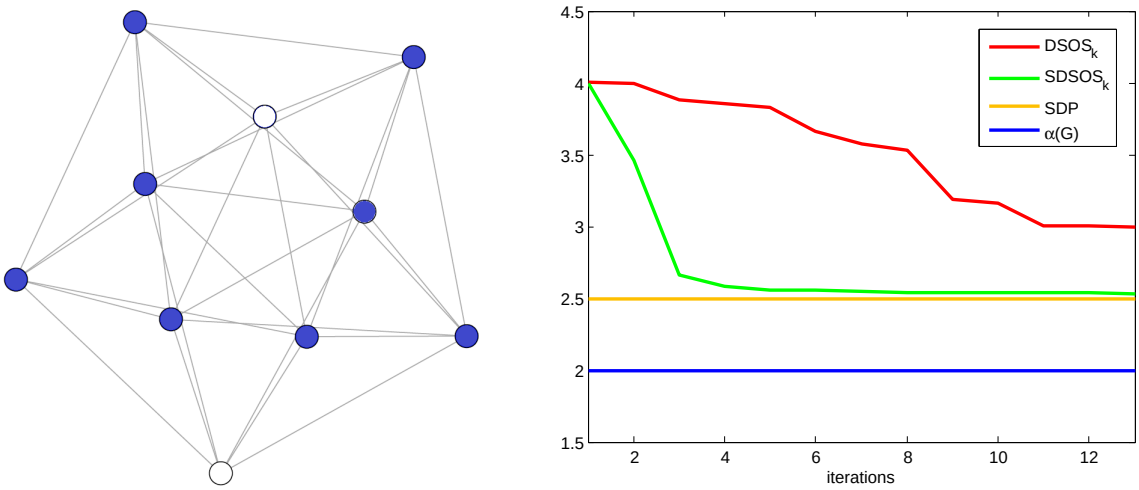
where w_1 and w_2 are two eigenvectors corresponding to the two most negative eigenvalues of X^*, the optimal solution of (27). If X^* only has one negative eigenvalue, we add a linear constraint to our problem; if $X^* \succeq 0$, then the bound obtained is identical to the one obtained through SDP (19) and we cannot hope to improve. Our next iterate is therefore

$$\begin{aligned} SDSOS_2(G) := \max_{X} \ & J \cdot X \\ & \text{s.t. } (A+I) \cdot X = 1 \\ & X \geq 0 \\ & V_i^T \cdot X V_i \succeq 0, \ i = 1, \ldots, \binom{n}{2} \\ & V^T \cdot X V \succeq 0. \end{aligned} \tag{28}$$

Note that the optimization problems generated iteratively in this fashion always remain SOCPs and their optimal values form a nonincreasing sequence of upper bounds on the stability number.

To illustrate the column generation method for both LPs and SOCPs, we consider the complement of the Petersen graph as shown in Figure 3(a) as an example. The stability number of this graph is 2 and one of its maximum stable sets is designated by the two white nodes. In Figure 3(b), we compare the upper bound obtained via (19) and the bounds obtained using the iterative LPs and SOCPs as described in (24) and (28).

Note that it takes 3 iterations for the SOCP sequence to produce an upperbound strictly within one unit of the actual stable set number (which would immediately tell us the value of α), whereas it takes 13 iterations for the LP sequence to do the same. It is also interesting to compare the sequence of LPs/SOCPs



(a) The complement of Petersen Graph

(b) Upper bounds on the stable set number $\alpha(G)$

Figure 3: Bounds obtained through SDP (19) and iterative SOCPs and LPs for the complement of the Petersen graph.

obtained through column generation to the sequence that one could obtain using the concept of r-dsos/r-sdsos polynomials. Indeed, LP (20) (resp. SOCP (25)) can be written in polynomial form as

$$\begin{aligned} DSOS_1(G) \text{ (resp. } SDSOS_1(G)) = \min_{\lambda} \ & \lambda \\ \text{s.t.} \ & \begin{pmatrix} x_1^2 \\ \vdots \\ x_n^2 \end{pmatrix}^T (\lambda(I+A) - J) \begin{pmatrix} x_1^2 \\ \vdots \\ x_n^2 \end{pmatrix} \text{ is dsos (resp. sdsos).} \end{aligned} \tag{29}$$

Iteration k in the sequence of LPs/SOCPs would then correspond to requiring that this polynomial be k-dsos or k-sdsos. For this particular example, we give the 1-dsos, 2-dsos, 1-sdsos and 2-sdsos bounds in Table 3.

Iteration	r-dsos bounds	r-sdsos bounds
$r = 0$	4.00	4.00
$r = 1$	2.71	2.52
$r = 2$	2.50	2.50

Table 3: Bounds obtained through rDSOS and rSDSOS hierarchies.

Though this sequence of LPs/SOCPs gives strong upper bounds, each iteration is more expensive than the iterations done in the column generation approach. Indeed, in each of the column generation iterations, only one constraint is added to our problem, whereas in the rDSOS/rSDSOS hierarchies, the number of constraints is roughly multiplied by n^2 at each iteration.

Finally, we investigate how these techniques perform on graphs with a large number of nodes, where the SDP bound cannot be found in a reasonable amount of time. The graphs we test these techniques on are Erdös-Rényi graphs $ER(n, p)$; i.e. graphs on n nodes where an edge is added between each pair of nodes independently and with probability p. In our case, we take n to be between 150 and 300, and p to be either

0.3 or 0.8 so as to experiment with both medium and high density graphs.[4]

In Table 4, we present the results of the iterative SOCP procedure and contrast them with the SDP bounds. The third column of the table contains the SOCP upper bound obtained through (27); the solver time needed to obtain this bound is given in the fourth column. The fifth and sixth columns correspond respectively to the SOCP iterative bounds obtained after 5 mins solving time and 10 mins solving time. Finally, the last two columns chart the SDP bound obtained from (19) and the time in seconds needed to solve the SDP. All SOCP and SDP experiments were done using Matlab, the solver MOSEK [2], the SPOTLESS toolbox [27], and a computer with 3.4 GHz speed and 16 GB RAM.

n	p	$SDSOS_1$	time (s)	$SDSOS_k$ (5 mins)	$SDSOS_k$ (10 mins)	$SDP(G)$	time (s)
150	0.3	105.70	1.05	39.93	37.00	20.43	221.13
150	0.8	31.78	1.07	9.96	9.43	6.02	206.28
200	0.3	140.47	1.84	70.15	56.37	23.73	1,086.42
200	0.8	40.92	2.07	12.29	11.60	6.45	896.84
250	0.3	176.25	3.51	111.63	92.93	26.78	4,284.01
250	0.8	51.87	3.90	17.25	15.39	7.18	3,503.79
300	0.3	210.32	5.69	151.71	134.14	29.13	32,300.60
300	0.8	60.97	5.73	19.53	17.24	7.65	20,586.02

Table 4: SDP bounds and iterative SOCP bounds obtained on ER(n,p) graphs.

From the table, we note that it is better to run the SDP rather than the SOCPs for small n, as the bounds obtained are better and the times taken to do so are comparable. However, as n gets bigger, the SOCPs become valuable as they provide good upper bounds in reasonable amounts of time. For example, for $n = 300$ and $p = 0.8$, the SOCP obtains a bound that is only twice as big as the SDP bound, but it does so 30 times faster. The sparser graphs don't do as well, a trend that we will also observe in Table 5. Finally, notice that the improvement in the first 5 mins is significantly better than the improvement in the last 5 mins. This is partly due to the fact that the SOCPs generated at the beginning are sparser, and hence faster to solve.

In Table 5, we present the results of the iterative LP procedure used on the same instances. All LP results were obtained using a computer with 2.3 GHz speed and 32GB RAM and the solver CPLEX 12.4 [15]. The third and fourth columns in the table contain the LP bound obtained with (23) and the solver time taken to do so. Columns 5 and 6 correspond to the LP iterative bounds obtained after 5 mins solving time and 10 mins solving time using the eigenvector-based column generation technique (see discussion around (24)). The seventh and eighth columns are the standard LP bounds obtained using (16) and the time taken to obtain the bound. Finally, the last column gives bounds obtained by column generation using "triples", as described in Section 4.2. In this case, we take $t_1 = 300,000$ and $t_2 = 500$.

[4] All instances used for these tests are available online at `http://aaa.princeton.edu/software`.

n	p	$DSOS_1$	time (s)	$DSOS_k$ (5m)	$DSOS_k$ (10m)	LP_2	time (s)	$LP_{triples}$ (10m)
150	0.3	117	< 1	110.64	110.26	75	< 1	89.00
150	0.8	46	< 1	24.65	19.13	75	< 1	23.64
200	0.3	157	< 1	147.12	146.71	100	< 1	129.82
200	0.8	54	< 1	39.27	36.01	100	< 1	30.43
250	0.3	194	< 1	184.89	184.31	125	< 1	168.00
250	0.8	68	< 1	55.01	53.18	125	< 1	40.19
300	0.3	230	< 1	222.43	221.56	150	< 1	205.00
300	0.8	78	< 1	65.77	64.84	150	< 1	60.00

Table 5: LP bounds obtained on the same $ER(n, p)$ graphs.

We note that in this case the upper bound with triples via column generation does better for this range of n than eigenvector-based column generation in the same amount of time. Furthermore, the iterative LP scheme seems to perform better in the dense regime. In particular, the first iteration does significantly better than the standard LP for $p = 0.8$, even though both LPs are of similar size. This would remain true even if the 3-clique inequalities were added as in (16), since the optimal value of LP_3 is always at least $n/3$. This is because the vector $(\frac{1}{3}, \ldots, \frac{1}{3})$ is feasible to the LP in (16) with $k = 3$. Note that this LP would have order n^3 constraints, which is more expensive than our LP. On the contrary, for sparse regimes, the standard LP gives better bounds than ours.

Overall, the high-level conclusion is that running the SDP is worthwhile for small sizes of the graph. As the number of nodes increases, column generation becomes valuable, providing upper bounds in a reasonable amount of time. Contrasting Tables 4 and 5, our initial experiments seem to show that the iterative SOCP bounds are better than the ones obtained using the iterative LPs. It may be valuable to experiment with different approaches to column generation however, as the technique used to generate the new atoms seems to impact the bounds obtained.

6 Conclusions and future research

For many problems of discrete and polynomial optimization, there are hierarchies of SDP-based sum of squares algorithms that produce provably optimal bounds in the limit [32], [22]. However, these hierarchies can often be expensive computationally. In this paper, we were interested in problem sizes where even the first level of the hierarchy is too expensive, and hence we resorted to algorithms that replace the underlying SDPs with LPs or SOCPs. We built on the recent work of Ahmadi and Majumdar on DSOS and SDSOS optimization [4], [3], which serves exactly this purpose. We showed that by using ideas from the theory of column generation, the performance of their algorithms is improvable. We did this by iteratively optimizing over increasingly larger structured subsets of the cone of positive semidefinite matrices, without resorting to the more expensive rDSOS and rSDSOS hierarchies.

There is certainly a lot of room to improve our column generation algorithms. In particular, we only experimented with a few types of pricing subproblems and particular strategies for solving them. As in the theory of large-scale integer programming, the success of column generation comes from good "engineering", which fine-tunes the algorithms to the problem at hand. Developing warm-start strategies for our iterative SOCPs for example, would be a very useful problem to work on in the future.

Here is another interesting research direction, which for illustrative purposes we outline for the problem studied in Section 4; i.e., minimizing a form on the sphere. Recall that given a form p of degree $2d$, we are trying to find the largest λ such that $p(x) - \lambda(\sum_{i=1}^{n} x_i^2)^d$ is a sum of squares. Instead of solving this sum of squares program, we looked for the largest λ for which we could write $p(x) - \lambda$ as a conic combination of a certain set of nonnegative polynomials. These polynomials for us were always either a single square or a sum of squares of polynomials. There are polynomials, however, that are nonnegative but not representable as a sum of squares. Two classic examples [28], [14] are the Motzkin polynomial

$$M(x, y, z) = x^6 + y^4z^2 + y^2z^4 - 3x^2y^2z^2,$$

and the Choi-Lam polynomial

$$CL(w, x, y, z) = w^4 + x^2y^2 + y^2z^2 + x^2z^2 - 4wxyz.$$

Either of these polynomials can be shown to be nonnegative using the arithmetic mean-geometric mean (am-gm) inequality, which states that if $x_1, \ldots, x_k \in \mathbb{R}$, then

$$x_1, \ldots, x_k \geq 0 \Rightarrow (\sum_{i=1}^{k} x_i)/k \geq (\Pi_{i=1}^{k} x_i)^{\frac{1}{k}}.$$

For example, in the case of the Motzkin polynomial, it is clear that the monomials x^6, y^4z^2 and y^2z^4 are nonnegative for all $x, y, z \in \mathbb{R}$, and letting x_1, x_2, x_3 stand for these monomials respectively, the am-gm inequality implies that

$$x^6 + y^4z^2 + y^2z^4 \geq 3x^2y^2z^2 \text{ for all } x, y, z \in \mathbb{R}.$$

These polynomials are known to be extreme in the cone of nonnegative polynomials and they cannot be written as a sum of squares (sos) [35].

It would be interesting to study the separation problems associated with using such non-sos polynomials in column generation. We briefly present one separation algorithm for a family of polynomials whose nonnegativity is provable through the am-gm inequality and includes the Motzkin and Choi-Lam polynomials. This will be a relatively easy-to-solve integer program in itself, whose goal is to find a polynomial q amongst this family which is to be added as our new "nonnegative atom".

The family of n-variate polynomials under consideration consists of polynomials with only $k+1$ nonzero coefficients, with k of them equal to one, and one equal to $-k$. (Notice that the Motzkin and the Choi-Lam polynomials are of this form with k equal to three and four respectively.) Let m be the number of monomials in p. Given a dual vector μ of (11) of dimension m, one can check if there exists a nonnegative degree $2d$ polynomial $q(x)$ in our family such that $\mu \cdot \text{coef}(q(x)) < 0$. This can be done by solving the following

integer program (we assume that $p(x) = \sum_{i=1}^m x^{\alpha_i}$):

$$\begin{aligned} \min_{c,y} \quad & \sum_{i=1}^m \mu_i c_i - \sum_{i=1}^m k\mu_i y_i \\ \text{s.t.} \quad & \sum_{i:\alpha_i \text{ is even}} \alpha_i c_i = k \sum_{i=1}^m \alpha_i y_i \\ & \sum_{i=1}^m c_i = k \\ & \sum_{i=1}^m y_i = 1 \\ & c_i \in \{0,1\}, y_i \in \{0,1\}, i = 1, \ldots, m, c_i = 0 \text{ if } \alpha_i \text{ is not even.} \end{aligned} \tag{30}$$

Here, we have $\alpha_i \in \mathbb{N}^n$ and the variables c_i, y_i form the coefficients of the polynomial q. The above integer program has $2m$ variables, but only $n+2$ constraints (not counting the integer constraints). If a polynomial $q(x)$ with a negative objective value is found, then one can add it as a new atom for column generation. In our specific randomly generated polynomial optimization examples, such polynomials did not seem to help in our preliminary experiments. Nevertheless, it would be interesting to consider other instances and problem structures.

Similarly, in the column generation approach to obtaining inner approximations of the copositive cone, one need not stick to positive demidefinite matrices. It is known that the 5×5 "Horn matrix" [13] for example is extreme in the copositive cone but cannot be written as the sum of a nonnegative and a positive semidefinite matrix. One could define a separation problem for a family of Horn-like matrices and add them in a column generation approach. Exploring such strategies is left for future research.

7 Acknowledgments

We are grateful to Anirudha Majumdar for insightful discussions and for his help with some of the numerical experiments in this paper.

References

[1] Gurobi optimizer reference manual. *URL: http://www. gurobi. com*, 2012.

[2] *MOSEK reference manual*, 2013. Version 7. Latest version available at `http://www.mosek.com/`.

[3] A. A. Ahmadi and A. Majumdar. DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization. In *Proceedings of the 48th Annual Conference on Information Sciences and Systems*. Princeton University, 2014.

[4] A. A. Ahmadi and A. Majumdar. DSOS and SDSOS: more tractable alternatives to sum of squares and semidefinite optimization. In preparation, 2015.

[5] A. A. Ahmadi and A. Majumdar. Some applications of polynomial optimization in operations research and real-time decision making. Prepint available at `http://arxiv.org/abs/1504.06002`. To appear in Optimization Letters, 2015.

[6] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

[7] E. Artin. Über die Zerlegung definiter Funktionen in Quadrate. In *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*, volume 5, pages 100–115. Springer, 1927.

[8] G. Barker and D. Carlson. Cones of diagonally dominant matrices. *Pacific Journal of Mathematics*, 57(1):15–32, 1975.

[9] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[10] I. M. Bomze and E. de Klerk. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *Journal of Global Optimization*, 24(2):163–185, 2002.

[11] E. Boros, P. Hammer, and G. Tavares. Local search heuristics for quadratic unconstrained binary optimization (qubo). *Journal of Heuristics*, 13(2):99–132, 2007.

[12] S. Burer. Copositive programming. In *Handbook on semidefinite, conic and polynomial optimization*, pages 201–218. Springer, 2012.

[13] S. Burer, K. M. Anstreicher, and M. Dür. The difference between $5\times$ 5 doubly nonnegative and completely positive matrices. *Linear Algebra and its Applications*, 431(9):1539–1552, 2009.

[14] M. D. Choi and T. Y. Lam. Extremal positive semidefinite forms. *Math. Ann.*, 231:1–18, 1977.

[15] CPLEX. V12. 4: Users manual for CPLEX. *International Business Machines Corporation*, 46(53):157.

[16] S. Dash. A note on QUBO instances defined on Chimera graphs. *arXiv preprint arXiv:1306.1202*, 2013.

[17] E. de Klerk and D. Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM Journal on Optimization*, 12(4):875–892, 2002.

[18] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[19] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*.

[20] D. Hilbert. Über die Darstellung Definiter Formen als Summe von Formenquadraten. *Math. Ann.*, 32, 1888.

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

[22] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[23] M. Laurent and F. Vallentin. *Lecture Notes on Semidefinite Optimization*. 2012.

[24] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1):193–228, 1998.

[25] L. Lovász. On the Shannon capacity of a graph. *Information Theory, IEEE Transactions on*, 25(1):1–7, 1979.

[26] A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control and verification of high-dimensional systems via DSOS and SDSOS optimization. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014.

[27] A. Megretski. SPOT: systems polynomial optimization tools. 2013.

[28] T. S. Motzkin. The arithmetic-geometric inequality. In *Inequalities*, pages 205–224. Academic Press, New York, 1967.

[29] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

[30] Y. Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, volume 33 of *Appl. Optim.*, pages 405–440. Kluwer Acad. Publ., Dordrecht, 2000.

[31] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, May 2000.

[32] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2, Ser. B):293–320, 2003.

[33] G. Pólya. Über positive Darstellung von Polynomen. *Vierteljschr. Naturforsch. Ges. Zürich*, 73:141–145, 1928.

[34] V. Powers and B. Reznick. A new bound for Pólya's theorem with applications to polynomials positive on polyhedra. *Journal of Pure and Applied Algebra*, 164(1):221–229, 2001.

[35] B. Reznick. Some concrete aspects of Hilbert's 17th problem. In *Contemporary Mathematics*, volume 253, pages 251–272. American Mathematical Society, 2000.

[36] A. Schrijver. A comparison of the Delsarte and Lovász bounds. *Information Theory, IEEE Transactions on*, 25(4):425–429, 1979.

[37] J. Sturm. *SeDuMi version 1.05*, Oct. 2001. Latest version available at `http://sedumi.ie.lehigh.edu/`.