Noname manuscript No.
(will be inserted by the editor)

Inner approximating the completely positive cone via the cone of scaled diagonally dominant matrices

João Gouveia · Ting Kei Pong · Mina Saee

the date of receipt and acceptance should be inserted later

Abstract Motivated by the expressive power of completely positive programming to encode hard optimization problems, many approximation schemes for the completely positive cone have been proposed and successfully used. Most schemes are based on outer approximations, with the only inner approximations available being a linear programming based method proposed by Bundfuss and Dür [9] and also Yıldırım [25], and a semidefinite programming based method proposed by Lasserre [17]. In this paper, we propose the use of the cone of nonnegative scaled diagonally dominant matrices as a natural inner approximation to the completely positive cone. Using projections of this cone we derive new graph-based second-order cone approximation schemes for completely positive programming, leading to both uniform and problem-dependent hierarchies. This offers a compromise between the expressive power of semidefinite programming and the speed of linear programming based approaches. Numerical results on random problems, standard quadratic programs and the stable set problem are presented to illustrate the effectiveness of our approach.

1 Introduction

Copositive programming and its dual counterpart of completely positive programming are classes of convex optimization problems that have in the past decades developed as a particularly expressive tool to encode optimization problems, especially for many problems arising from combinatorial or quadratic optimization. A classical example of that can be found in [10], which shows that general quadratic programs with a mix of binary and continuous variables can be expressed as copositive programs. A large body of work has been developed in the area and there is a series of survey papers that can be consulted for further information. We refer the readers to [8, 11, 13] and references therein for more details.

The first author's research was partially supported by FCT under grants UID/MAT/00324/2019 through CMUC, and P2020 SAICTPAC/0011/2015. The second author's research was supported partly by a research grant (G-UADF) from The Hong Kong Polytechnic University and the third author's research was supported by a PhD scholarship from FCT, grant PD/BD/128060/2016.

J. Gouveia
CMUC, Department of Mathematics, University of Coimbra, Portugal
E-mail: jgouveia@mat.uc.pt

T. K. Pong
Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong, People's Republic of China
E-mail: tk.pong@polyu.edu.hk

M. Saee
CMUC, Department of Mathematics, University of Coimbra, Portugal
E-mail: minasaee@mat.uc.pt

In this paper we will focus on general completely positive programs which are linear optimization problems of the form (see Section 1.1 below for notation)

$$\begin{aligned} v_p := \min\ & \mathrm{tr}(CX) \\ \text{s.t. } & \mathrm{tr}(A_i X) = b_i,\ i = 1, \dots, m, \\ & X \in \mathcal{CP}^n, \end{aligned} \tag{1.1}$$

where C and A_i, $i = 1, \dots, m$ are symmetric matrices, and $\mathcal{CP}^n$ is the closed cone of $n \times n$ completely positive matrices defined as

$$\mathcal{CP}^n := \{X \in \mathcal{S}^n : \exists B \geq 0, \quad X = B^T B\}. \tag{1.2}$$

We also consider the dual problem of (1.1), which is the following copositive programming problem

$$\begin{aligned} v_d := \max\ & b^T y \\ \text{s.t. } & C - \textstyle\sum_{i=1}^m y_i A_i \in \mathcal{COP}^n, \end{aligned} \tag{1.3}$$

where $\mathcal{COP}^n$ is the closed cone of $n \times n$ copositive matrices and is defined as

$$\mathcal{COP}^n := \{X \in \mathcal{S}^n : v^T X v \geq 0, \quad \forall v \geq 0\}. \tag{1.4}$$

It is well known that completely positive programming problems (1.1) are NP-hard in general. Several approximation schemes have been proposed and successfully used in the literature, based on approximations to $\mathcal{CP}^n$. The simplest one is to replace $\mathcal{CP}^n$ by the cone of nonnegative positive semidefinite matrices, which is strictly larger than $\mathcal{CP}^n$ when $n \geq 5$, hence leading to a lower bound to v_p. Other popular lower bounds are those relying on semidefinite programming sums of squares techniques as introduced in [19]. For upper bounds based on inner approximations to $\mathcal{CP}^n$, the literature is somewhat sparser.

One way of constructing inner approximations to $\mathcal{CP}^n$ is to make use of the fact that the extreme rays of $\mathcal{CP}^n$ are matrices of the form vv^T with $v \in \mathbb{R}^n_+ \backslash \{0\}$; see [1]. Thus, one can pick uniformly spaced $v \in \Delta^n = \{x \in \mathbb{R}^n_+ : \sum x_i = 1\}$, and approximate $\mathcal{CP}^n$ by the cone the matrices vv^T generate (see [9, 25]). This leads to linear programming (LP) approximations to (1.1). Another inner approximation to $\mathcal{CP}^n$ is that proposed in [17], based on the theory of moments, leading to semidefinite programming (SDP) approximations to (1.1). In both cases we have hierarchies that give upper bounds to (1.1), and dually lower bounds to (1.3), and converge to the optimal value/solutions of (1.1). These inner approximations are uniform (i.e., problem-independent) approximations, giving rise to either LP or SDP problems. See also [26] for a more thorough treatment of inner approximations. An extra step taken as an adaptive linear approximation algorithm was proposed in [9]. This uses information obtained from an upper bound approximation to selectively refine the hierarchy, leading to problem-dependent LP approximations.

In this paper, we propose a new inner approximation scheme to $\mathcal{CP}^n$ that is based on *second-order cone programming* (SOCP) problems and can be either uniform or problem-dependent. Our approach is motivated by the recent work in [3, 4] that uses the cone of scaled diagonally dominant matrices for inner-approximating the cone of positive semidefinite matrices. Specifically, we use the cones of nonnegative scaled diagonally dominant matrices and their projections as a natural inner approximation to $\mathcal{CP}^n$, and derive a new SOCP-based approximation scheme for completely positive and copositive programming. Our approximation scheme has a natural graphical interpretation. By exploiting this interpretation, we can flexibly expand or trim the SOCP problems in our hierarchy, leading to both uniform and problem-dependent approximation schemes. The use of SOCP offers a compromise between the expressive power of SDP, that comes at a significant computational cost, and the speed of LP approaches, that have inherently lower expressive power. Numerical experiments on solving random instances, standard quadratic programs and the stable set problem demonstrate the effectiveness of our approximation schemes.

The rest of the paper is organized as follows. We present notation and state our blanket assumptions concerning (1.1) and (1.3) in Section 1.1. Properties of the scaled diagonally dominant matrices are reviewed in Section 2, and a graphical refinement scheme is discussed. We derive our uniform inner approximation schemes in Section 3 with a convergence analysis, and discuss several problem-dependent inner approximation schemes in Section 4. Numerical experiments are reported in Section 5.

1.1 Notation and blanket assumptions

In this paper, we use $\mathcal{S}^n$ to denote the space of $n \times n$ symmetric matrices. Matrices are denoted by upper case letters, and their entries are represented in the corresponding lower case letters, e.g., d_{ij} as the (i,j)th entry of the matrix D; we also use lower case letters to denote vectors. For vectors u, $v \in \mathbb{R}^n$, we write $u \geq 0$ if u is elementwise nonnegative, and use $[u,v]$ to denote the line segment between u and v, i.e.,

$$[u,v] := \{tu + (1-t)v : \ t \in [0,1]\}.$$

For an $X \in \mathcal{S}^n$, we write $X \succeq 0$ if X is positive semidefinite, and write $X \geq 0$ if X is elementwise nonnegative. We also write the trace of X as $\mathrm{tr}(X)$. We use E and I to denote the square matrix of all ones and the identity matrix, respectively, whose dimensions should be clear from the context. Finally, for a linear map $\mathcal{A} : \mathcal{S}^n \to \mathcal{S}^m$, we use $\mathcal{A}^*$ to denote its adjoint.

The cone of positive semidefinite $n \times n$ matrices is denoted by $\mathcal{S}^n_+$. We also use $\mathcal{N}^n$ to denote the cone of $n \times n$ symmetric nonnegative matrices, i.e.,

$$\mathcal{N}^n := \{X \in \mathcal{S}^n : \ X \geq 0\},$$

and an $n \times n$ real symmetric matrix is doubly nonnegative if it is positive semidefinite and entrywise nonnegative. It is known that the cones in (1.2) and (1.4) are dual to each other, i.e., $\mathcal{CP}^n = (\mathcal{COP}^n)^*$ and $\mathcal{COP}^n = (\mathcal{CP}^n)^*$; here,

$$\mathcal{C}^* := \{Y \in \mathcal{S}^n : \ \mathrm{tr}(XY) \geq 0, \quad \forall X \in \mathcal{C}\}$$

for a closed convex cone $\mathcal{C} \subseteq \mathcal{S}^n$. Moreover, it is also known that the cone of positive semidefinite matrices and the cone of symmetric nonnegative matrices are self-dual, i.e., $\mathcal{S}^n_+ = (\mathcal{S}^n_+)^*$ and $\mathcal{N}^n = (\mathcal{N}^n)^*$.

Throughout this paper, we make the following **blanket assumptions** concerning (1.1) and (1.3):

A1. Problem (1.1) is feasible.
A2. The mapping $X \mapsto (\mathrm{tr}(A_1X), \ldots, \mathrm{tr}(A_mX))$ is surjective.
A3. Problem (1.3) is strictly feasible, i.e., there exists $\bar{y}$ satisfying

$$C - \sum_{i=1}^m \bar{y}_i A_i \in \mathrm{int}\, \mathcal{COP}^n.$$

Under these assumptions, the dual Slater condition holds. Therefore we have $v_p = v_d$, with both values being finite and the primal optimal value v_p being attained.

2 The scaled diagonally dominant cone and beyond

In this section, we present the basis for our construction of inner approximations in Sections 3 and 4. Our construction is motivated by the work in [3, 4], which studied inner approximations of the cone of positive semidefinite matrices based on the cones of diagonally dominant and scaled diagonally dominant matrices. While their work can be directly applied to the existing SOS hierarchies to yield outer approximations of $\mathcal{CP}^n$ (see [4, Section 4.2]) we show an alternative approach, based on the same cones but using them in a fundamentally different way, in order to obtain an inner approximation to $\mathcal{CP}^n$. We first recall the following definition of diagonally dominant and scaled diagonally dominant matrices from [4, Definition 3.3].

Definition 1 A symmetric matrix A is diagonally dominant[1] if $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$ for all i, and is said to be scaled diagonally dominant (sdd) if there exists a diagonal matrix D with positive diagonal entries such that DAD is diagonally dominant.

Let SDD^n be the cone of $n \times n$ sdd matrices. It is clear that SDD^1 is just the set of nonnegative real numbers. One can deduce from [5, Theorem 9] a convenient characterization of sdd matrices for $n \geq 2$. In fact one can show that the cone SDD^n is given by

$$\mathrm{SDD}^n := \sum_{1 \leq i < j \leq n} \iota_{ij}(\mathcal{S}^2_+), \tag{2.1}$$

where $\iota_{ij} : \mathcal{S}^2 \to \mathcal{S}^n$ is the map that sends an $S \in \mathcal{S}^2$ to the matrix D given by

$$d_{rs} := \begin{cases} s_{11} & \text{if } (r,s) = (i,i), \\ s_{12} & \text{if } (r,s) = (i,j), \\ s_{21} & \text{if } (r,s) = (j,i), \\ s_{22} & \text{if } (r,s) = (j,j), \\ 0 & \text{otherwise.} \end{cases}$$

This cone is therefore given in terms of 2×2 semidefinite constraints or, in other words, second-order cone constraints, which makes it quite suitable to use in convex optimization. One can prove the following basic properties of SDD^n, and of the set $\mathrm{SDD}^n_+ := \mathrm{SDD}^n \cap \mathcal{N}^n$. Note that item (i) in Proposition 1 below can be found in [3], and a more general version of it can be found in [21, Lemma 5]. We include it here for completeness. In what follows ι^*_{ij} denotes the adjoint of the map ι_{ij}, which in this case can be defined by saying that $\iota^*_{ij}(S)$ is the 2×2 submatrix of S indexed by rows and columns i and j.

Proposition 1 *For $n \geq 2$, the following statements hold.*

(i) $(\mathrm{SDD}^n)^* = \{Q \in \mathcal{S}^n : \ \iota^*_{ij}(Q) \succeq 0, \quad \forall 1 \leq i < j \leq n\}$.
(ii) $(\mathrm{SDD}^n_+)^* = (\mathrm{SDD}^n)^* + \mathcal{N}^n$.
(iii) $\mathrm{SDD}^n_+ = \sum_{1 \leq i < j \leq n} \iota_{ij}(\mathcal{S}^2_+ \cap \mathcal{N}^2)$.

Proof We first prove (i). Recall from (2.1) that $\mathrm{SDD}^n = \sum_{1 \leq i < j \leq n} \iota_{ij}(\mathcal{S}^2_+)$. Thus, we have from [22, Corollary 16.3.2] that

$$(\mathrm{SDD}^n)^* = \bigcap_{1 \leq i < j \leq n} (\iota_{ij}(\mathcal{S}^2_+))^*,$$

from which the desired equality follows immediately.

Next, we prove (ii). Note that

$$\sum_{1 \leq i < j \leq n} \iota_{ij}(E) \in \mathrm{SDD}^n \cap \operatorname{int} \mathcal{N}^n.$$

Thus, we conclude from [22, Corollary 16.3.2] that

$$(\mathrm{SDD}^n_+)^* = (\mathrm{SDD}^n \cap \mathcal{N}^n)^* = (\mathrm{SDD}^n)^* + \mathcal{N}^n.$$

Finally, we prove (iii). It is clear that $\mathrm{SDD}^n_+ \supseteq \sum_{1 \leq i < j \leq n} \iota_{ij}(\mathcal{S}^2_+ \cap \mathcal{N}^2)$. For the converse inclusion, consider any $Q \in \mathrm{SDD}^n_+$. Then Q is nonnegative and can be written as $\sum_{1 \leq i < j \leq n} \iota_{ij}(S_{ij})$ for some $S_{ij} \in \mathcal{S}^2_+$, $1 \leq i < j \leq n$. Observe that each S_{ij} has nonnegative diagonal entries, and moreover, its nondiagonal entry equals the (i,j)th entry of Q, which is also nonnegative. Thus, $S_{ij} \in \mathcal{S}^2_+ \cap \mathcal{N}^2$ and hence $Q \in \sum_{1 \leq i < j \leq n} \iota_{ij}(\mathcal{S}^2_+ \cap \mathcal{N}^2)$. This completes the proof. ∎

[1] Note that our definition is different from the classical definition of diagonal dominance (see [15, Definition 6.1.9]) in that we require the diagonal entries of A to be nonnegative.

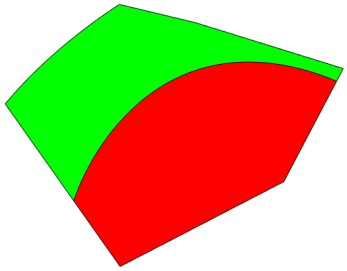

Fig. 1 Comparison of $\mathcal{S}^5_+ \cap \mathcal{N}^5$ with SDD^5_+

Since 2×2 nonnegative positive semidefinite matrices are completely positive, we see from Proposition 1(iii) that SDD^n_+ is an inner approximation to $\mathcal{CP}^n$. In Figure 1 we show a random 2-dimensional slice of the cone of doubly nonnegative 5×5 matrices (i.e., $\mathcal{S}^5_+ \cap \mathcal{N}^5$) with the slice of SDD^5_+ highlighted in red. The cone $\mathcal{CP}^5$ is sandwiched between them.

This simple inner approximation can be used as a basis to construct more general inner second-order cone approximations for $\mathcal{CP}^n$. To do that we consider a useful variant of SDD^n_+ that will help us construct inner approximations of $\mathcal{CP}^n$.

Definition 2 Let $U \in \mathbb{R}^{t\times n}_+$ have row sum 1. Define

$$\mathrm{SDD}^n_+(U) := \{U^T Y U : \ Y \in \mathrm{SDD}^t_+\} = U^T(\mathrm{SDD}^t_+)U. \tag{2.2}$$

The above definition is similar to the development in [3, Section 3.1], which makes use of the so-called $\mathrm{DD}(U)$. Here we assume that U has nonnegative entries so that $\mathrm{SDD}^n_+(U)$ will be a subcone of $\mathcal{CP}^n$; see Proposition 2 below. In addition, we assume that the rows of U have sum one: we can then always think of the rows of U as points in the simplex Δ^n. This is no less general than just considering $U \in \mathbb{R}^{t\times n}_+$ with nonzero rows, because scaling rows of U by positive scalars does not change $\mathrm{SDD}^n_+(U)$. Note that $\mathrm{SDD}^n_+(U)$ is simply a linear image of SDD^t_+ into $\mathcal{S}^n$.

Some basic properties of this set are that $\mathrm{SDD}^n_+(I_n) = \mathrm{SDD}^n_+$, and that if $U \in \mathbb{R}^{t\times n}_+$ is a submatrix of $\tilde{U} \in \mathbb{R}^{s\times n}_+$ then $\mathrm{SDD}^n_+(U) \subseteq \mathrm{SDD}^n_+(\tilde{U})$. We show in the next example that $\mathrm{SDD}^n_+(U)$ can be strictly larger than SDD^n_+ in general.

Example 1 One can see that the matrix

$$M = \begin{bmatrix} 6 & 5 & 5 \\ 5 & 6 & 5 \\ 5 & 5 & 6 \end{bmatrix}$$

is in $\mathcal{S}^3_+ \cap \mathcal{N}^3$. However, $M \notin \mathrm{SDD}^3_+$; indeed, if we define

$$W := \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix},$$

then $\mathrm{tr}(WM) < 0$ but $W \in (\mathrm{SDD}^3_+)^*$ thanks to Proposition 1(ii), showing that $M \notin \mathrm{SDD}^3_+$.

Now, suppose we set U to be the 4×3 matrix constructed from concatenating the identity I_3 with an all $\frac{1}{3}$ row vector, i.e.,

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix},$$

and consider the set $\mathrm{SDD}^3_+(U)$. Then we know $\mathrm{SDD}^3_+ \subseteq \mathrm{SDD}^3_+(U)$ because I_3 is a submatrix of U. Furthermore, we have

$$M = U^T \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 3 \\ 3 & 3 & 3 & 27 \end{bmatrix} U \in \mathrm{SDD}^3_+(U),$$

where the inclusion holds because

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 3 \\ 3 & 3 & 3 & 27 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 9 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 9 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 3 & 9 \end{bmatrix} \in \mathrm{SDD}^4_+.$$

Consequently, $\mathrm{SDD}^3_+(U)$ is a strictly larger set than SDD^3_+.

We next give an important characterization of $\mathrm{SDD}^n_+(U)$ that is crucial in our development of inner approximation schemes in Sections 3 and 4. Recall from (1.2) that $\mathcal{CP}^n$ can be seen as the convex hull of all vv^T with $v \in \mathbb{R}^n_+$. The next theorem shows that one can think of $\mathrm{SDD}^n_+(U)$ similarly.

Theorem 1 *Let $U \in \mathbb{R}^{t\times n}_+$ have row sum 1. Then $\mathrm{SDD}^n_+(U)$ is the conic hull of all vv^T with v belonging to some line segment $[u_i, u_j]$, where u_i is the i-th row of U.*

Proof For $t = 1$, the result follows immediately from the definition, so assume $t \geq 2$. Note from Proposition 1(iii) and (2.2) that any matrix in $\mathrm{SDD}^n_+(U)$ can be written as

$$\sum_{1\leq i<j\leq t} U^T \iota_{ij}(S_{ij})U$$

for some $S_{ij} \in \mathcal{S}^2_+ \cap \mathcal{N}^2$. Moreover, any matrix $S \in \mathcal{S}^2_+ \cap \mathcal{N}^2$ can be written as $S = v_1v_1^T + v_2v_2^T$ for some nonnegative vectors $v_i \in \mathbb{R}^2_+$. Furthermore, we know that for any $v \in \mathbb{R}^2$, it holds that $\iota_{ij}(vv^T) = ww^T$ where $w \in \mathbb{R}^t$ is the vector whose ith entry is v_1, jth entry equals v_2, and is zero otherwise. Hence, we deduce that any matrix in $\mathrm{SDD}^n_+(U)$ can be written as

$$\sum_{k=1}^{N} U^T w_k w_k^T U,$$

where each $w_k \in \mathbb{R}^t$ is nonnegative and has a support of cardinality at most 2. Conversely, it is easy to see that any matrix that can be written as such a sum is in $\mathrm{SDD}^n_+(U)$. But each U^Tw, with $w \neq 0$, is simply a (nonzero) conic combination of two rows of U, u_i and u_j; so, up to positive scaling, it is in $[u_i, u_j]$, proving our claim. ∎

We can now prove the following properties of $\mathrm{SDD}^n_+(U)$.

Proposition 2 *Let $U \in \mathbb{R}^{t\times n}_+$ have row sum 1. Then the following statements hold.*

(i) *The cone $\mathrm{SDD}^n_+(U)$ is a closed sub-cone of $\mathcal{CP}^n$.*
(ii) $(\mathrm{SDD}^n_+(U))^* = \{Y : UYU^T \in (\mathrm{SDD}^t_+)^*\} = \{Y : UYU^T \in (\mathrm{SDD}^t)^* + \mathcal{N}^t\}$.

Proof From Theorem 1, it follows that $\mathrm{SDD}^n_+(U)$ is a sub-cone of $\mathcal{CP}^n$. It remains to prove closedness. Since U is nonnegative and has no zero rows, the origin is not in the convex hull of vv^T, where v belongs to some $[u_i, u_j]$, and u_i is the i-th row of U. Hence $\mathrm{SDD}^n_+(U)$ is the conic hull of a compact convex set not containing the origin. Thus, it is closed.

To prove (ii), recall that $\mathrm{SDD}_+^n(U) = U^T(\mathrm{SDD}_+^t)U$. From this we see that $Y \in (\mathrm{SDD}_+^n(U))^*$ if and only if

$$\mathrm{tr}(Y(U^TWU)) \geq 0 \quad \forall W \in \mathrm{SDD}_+^t,$$

which is the same as $UYU^T \in (\mathrm{SDD}_+^t)^*$. This proves the first equality. The second equality in (ii) is trivial when $t = 1$ and follows from Proposition 1(ii) when $t \geq 2$. This completes the proof. ∎

Note that the construction of $\mathrm{SDD}_+^n(U)$ is fairly general. Anytime we have a cone $\mathcal{C} \subseteq \mathcal{CP}^t$ and a matrix $U \in \mathbb{R}_+^{t\times n}$ whose rows have sum one, one can define the cone

$$\mathcal{C}(U) := \{U^TYU : Y \in \mathcal{C}\} = U^T\mathcal{C}U. \tag{2.3}$$

This is easily seen to always verify $\mathcal{C}(U) \subseteq \mathcal{CP}^n$, since $\mathcal{C}(U) \subseteq U^T\mathcal{CP}^tU \subseteq \mathcal{CP}^n$. It is helpful to state in this language the usual LP inner approximations to $\mathcal{CP}^n$. Let Diag_+^n be the set of nonnegative $n \times n$ diagonal matrices. Clearly $\mathrm{Diag}_+^n \subseteq \mathcal{CP}^n$, so we can define

$$\mathrm{Diag}_+^n(U) := \{U^TYU : Y \in \mathrm{Diag}_+^t\}. \tag{2.4}$$

This is nothing more than the conic hull of the matrices $u_iu_i^T$, $i = 1, \ldots, t$, where u_i is the i-th row of U. The use of (2.4) for inner approximation corresponds to the standard LP approximation strategy used, for example, in [9], where strategies for efficient choices of U were explored.

Another possibility for obtaining an LP relaxation would be to use the cone of $n \times n$ symmetric nonnegative diagonally dominant matrices, denoted by DD_+^n. We have $\mathrm{Diag}_+^n \subseteq \mathrm{DD}_+^n \subseteq \mathrm{SDD}_+^n$. So, if we define

$$\mathrm{DD}_+^n(U) := \{U^TYU : Y \in \mathrm{DD}_+^t\}, \tag{2.5}$$

we would get $\mathrm{Diag}_+^n(U) \subseteq \mathrm{DD}_+^n(U) \subseteq \mathrm{SDD}_+^n(U)$. However, since one can easily see that DD_+^n is the conic hull of $(e_i + e_j)(e_i + e_j)^T$ for $1 \leq i \leq j \leq n$, it is not hard to see that $\mathrm{DD}_+^n(U)$ is simply the conic hull of $(u_i + u_j)(u_i + u_j)^T$ for $1 \leq i \leq j \leq t$, and hence can be expressed in terms of $\mathrm{Diag}_+^n(U')$ for some U' that contains U as a submatrix.

Other choices would be to use not submatrices in $\mathcal{S}_+^2$, as we did for SDD_+^n, but matrices in $\mathcal{S}_+^3$ or $\mathcal{S}_+^4$. Note that it is still true in these two cases that $\mathcal{S}_+^i \cap \mathcal{N}^i \subseteq \mathcal{CP}^i$. These cones would give better approximations, but we would get a much higher number of constraints that would not be second-order cone constraints but fully semidefinite. While the semidefinite constraints would still be small, the process would become more cumbersome and significantly less tractable.

2.1 A graphical refinement

We saw above that $\mathrm{SDD}_+^n(U)$ is a natural inner approximation to $\mathcal{CP}^n$. Furthermore, Theorem 1 suggests that the fundamental property of U that guides the approximation is the collection of segments $[u_i, u_j]$. We might associate to the points u_i vertices of a graph, and to the segments its edges, and think of the collection of points and segments as a concrete realization of the graph in $\mathbb{R}^n$. This insight can be used to refine the approximation, making it more flexible. We start by generalizing the notion of SDD.

Given a graph G with vertex set $\{1, \ldots, n\}$ and edge set $\mathcal{E}$, we define

$$\mathrm{SDD}^G := \sum_{\{i,j\}\in\mathcal{E}} \iota_{ij}(\mathcal{S}_+^2),$$

and we set $\mathrm{SDD}^G := \{0\}$ if $\mathcal{E} = \emptyset$ by convention. The graph G simply encodes which principal 2×2 submatrices will be required to be semidefinite. In particular, if we consider G to be the complete graph K^n, for $n \geq 2$, this is simply SDD^n. We can define SDD_+^G as the nonnegative matrices in SDD^G, similarly as before. Then we can naturally define a generalization of $\mathrm{SDD}_+^n(U)$:

Definition 3 For a graph G with t vertices and a matrix $U \in \mathbb{R}_+^{t\times n}$ whose rows have sum one, we define the cone $\mathrm{SDD}_+^G(U)$ as

$$\mathrm{SDD}_+^G(U) := \{U^T Y U : \ Y \in \mathrm{SDD}_+^G\} = U^T(\mathrm{SDD}_+^G)U.$$

It will be helpful to think of the rows of U as points in the standard simplex Δ^n (i.e. with nonnegative coordinates summing to one). These points correspond to vertices of the graph G, and the edge set of G simply encodes which pairs of rows of U (vertices) are "connected". In other words, the pair (G,U) is a realization of the graph G inside Δ^n with segments for edges. We will denote by $\mathrm{seg}(G,U)$ the set of points in some of the segments, i.e,

$$\mathrm{seg}(G,U) = \bigcup_{\{i,j\}\in\mathcal{E}} [u_i, u_j],$$

where u_i is the i-th row of U. This set completely controls the geometry of the cone. Based on this notion and the proof of Theorem 1, we can immediately obtain the following refinement of Theorem 1 for the representation of $\mathrm{SDD}_+^G(U)$.

Theorem 2 *Let G be a graph with t vertices and $U \in \mathbb{R}_+^{t\times n}$ be a matrix whose rows have sum one. Then $\mathrm{SDD}_+^G(U)$ is the conic hull of all vv^T with $v \in \mathrm{seg}(G,U)$.*

Theorem 2 gives a simple way of translating results from the graph language to results about cones. In particular if we have $\mathrm{seg}(G,U) \subseteq \mathrm{seg}(G',U')$, we have $\mathrm{SDD}_+^G(U) \subseteq \mathrm{SDD}_+^{G'}(U')$, and furthermore $\mathrm{SDD}_+^G(U) \subseteq \mathrm{SDD}_+^{K^t}(U) = \mathrm{SDD}_+^n(U) \subseteq \mathcal{CP}^n$, for all graphs G with t vertices and matrices $U \in \mathbb{R}_+^{t\times n}$ whose rows have sum one. On the other hand, if every node of the graph G is covered by some edges, then $\mathrm{SDD}_+^G(U) \supseteq \mathrm{Diag}_+^n(U)$, the usual LP inner approximation. Thus, the graphical notation allows us to construct intermediate approximations somewhere in between the simple LP inner approximation and the full $\mathrm{SDD}_+^n(U)$ version.

We end the section by noting that most of our other previous results concerning SDD_+^n and $\mathrm{SDD}_+^n(U)$ can be adapted with no effort to this new cone.

Theorem 3 *Given a graph G with t vertices and edge set $\mathcal{E}$, and a matrix $U \in \mathbb{R}_+^{t\times n}$ whose rows have sum one, we have the following properties.*

(i) $(\mathrm{SDD}^G)^* = \{Q \in \mathcal{S}^n : \ \iota_{ij}^*(Q) \succeq 0 \ \ \forall\{i,j\} \in \mathcal{E}\}$;
(ii) $\mathrm{SDD}_+^G = \sum_{\{i,j\}\in\mathcal{E}} \iota_{ij}(S_+^2 \cap \mathcal{N}^2)$;
(iii) $(\mathrm{SDD}_+^G(U))^* = \{Y : \ UYU^T \in (\mathrm{SDD}_+^G)^*\}$;
(iv) $\mathrm{SDD}_+^G(U)$ *is a closed sub-cone of* $\mathcal{CP}^n$.

Proof Immediate from the proofs of Proposition 1 and Proposition 2. ∎

3 Inner approximation schemes for the completely positive cone

The main idea of this section is to approximate the solution to (1.1) by using the cones $\mathrm{SDD}_+^G(U)$ to replace $\mathcal{CP}^n$. More concretely our scheme is based on the following family of optimization problems, which depends on a graph G on t vertices and a $U \in \mathbb{R}_+^{t\times n}$ whose rows have sum one:

$$\begin{aligned} v_p(G,U) := \min\ & \mathrm{tr}(CX) \\ \text{s.t. } & \mathrm{tr}(A_i X) = b_i, \ i = 1,\dots,m, \\ & X \in \mathrm{SDD}_+^G(U), \end{aligned} \tag{3.1}$$

and its dual problem given by

$$\begin{aligned} v_d(G,U) := \max\ & b^T y \\ \text{s.t. } & C - \textstyle\sum_{i=1}^m y_i A_i \in (\mathrm{SDD}_+^G(U))^*. \end{aligned} \tag{3.2}$$

Note that the semidefinite constraints in (3.1) are imposed only on 2×2 matrices. Thus, these problems are SOCP problems.

Recall from Theorem 3 that $\mathrm{SDD}_+^G(U)$ and $(\mathrm{SDD}_+^G(U))^*$ are both closed convex cones. Also, notice that (3.2) has a strictly feasible point due to Assumption **A**3 and the fact that $\mathcal{COP}^n \subseteq (\mathrm{SDD}_+^G(U))^*$ (which follows from $\mathrm{SDD}_+^G(U) \subseteq \mathcal{CP}^n$). Consequently, if Problem (3.1) is feasible, then $v_p(G,U) = v_d(G,U)$, both values are finite and $v_p(G,U)$ is attained. Moreover, we conclude from $\mathrm{SDD}_+^G(U) \subseteq \mathcal{CP}^n$ that $v_p(G,U) \geq v_p$. Furthermore, we have already pointed out that augmenting the embedded graph (G,U) leads to an enlargement in $\mathrm{SDD}_+^G(U)$. In view of these observations, we will discuss strategies for constructing an "enlarging" sequence of graphs $\{(G^k,U^k)\}$ to possibly tighten the gap $v_p(G^k,U^k) - v_p$ as k increases.

To simplify our terminology, we make the following definition.

Definition 4 A sequence of embedded graphs $\{(G^k,U^k)\}$ is called a positively enlarging sequence if $\mathrm{seg}(G^k,U^k) \subseteq \mathrm{seg}(G^{k+1},U^{k+1})$, each U is a nonnegative matrix having at least n rows, each row of U (the realizations of vertices of G) sums to one, and each node of G is covered by at least one edge.

Positively enlarging sequences verify $v_p(G^k,U^k) \geq v_p(G^{k+1},U^{k+1}) \geq v_p$ by construction. Furthermore, once (3.1) is feasible for some $k = k_0$, it will remain feasible whenever $k \geq k_0$, since the sequence of sets $\{\mathrm{SDD}_+^{G_k}(U^k)\}$ are monotonically increasing. Moreover, we have noted above that we might think of the rows of U to be in the simplex Δ^n so that we can think of this as an enlarging family of graphs embedded in Δ^n.

We next study convergence of our inner approximation schemes for (1.1) based on (3.1) when $\{(G^k,U^k)\}$ is a positively enlarging sequence. We first prove a convergence result concerning a similar approximation scheme, which uses $\mathrm{Diag}_+^n(U)$ (as defined in (2.4)) in place of $\mathrm{SDD}_+^G(U)$ in (3.1). This strategy was used in [9], which studied the pairs (3.1) and (3.2) with $\mathrm{Diag}_+^n(U)$ in place of $\mathrm{SDD}_+^G(U)$, and constructed an "enlarging" sequence $\{U^k\}$ by adding new rows to U^k from Δ^n at each step. To determine what rows to add, they solve another LP approximation scheme based on U, which they see as the set of vertices of a simplicial partition of Δ^n, and use its results to construct a sequence of $\{U^k\}$ with an increasing number of rows. In studying the convergence of that method they proved a version of the following result for copositive programming problems in [9, Theorem 4.2]. The version presented below will be useful for studying convergence of our inner approximation schemes for (1.1).

Theorem 4 *Assume that* (1.1) *is strictly feasible. Let* $\{U^k\}$ *be a sequence of matrices whose rows have sum one, where for each* k, $U^k \in \mathbb{R}_+^{t_k \times n}$ *for some* $t_k \geq n$. *Suppose that*

$$\lim_{k\to\infty} \max_{x\in\Delta^n} \min_{i=1,\ldots,t_k} \|x - u_i^k\| = 0, \tag{3.3}$$

where u_i^k *is the* i*-th row of* U^k. *Consider for each* k *the following problem:*

$$\begin{aligned} \tilde{v}_p(U^k) := \min\ & \mathrm{tr}(CX) \\ \text{s.t. } & \mathrm{tr}(A_iX) = b_i,\ i = 1,\ldots,m, \\ & X \in \mathrm{Diag}_+^n(U^k). \end{aligned} \tag{3.4}$$

Then the following statements hold.

(i) $\tilde{v}_p(U^k)$ *is finite for all sufficiently large* k *and* $\lim_{k\to\infty} \tilde{v}_p(U^k) = v_p$.
(ii) *The solution set of* (3.4) *is nonempty and uniformly bounded for all sufficiently large* k.
(iii) *Let* X^k *be a solution of* (3.4) *whenever the solution set is nonempty. Then any accumulation point of* $\{X^k\}$ *is a solution of* (1.1).

Proof Note that the $\mathrm{Diag}_+^n(U^k)$ defined in (2.4) is the conic hull of $u_i^k {u_i^k}^T$, where u_i^k are rows of U^k. Note also that any element X in $\mathcal{CP}^n$ can be written as the conic combination of $\frac{n(n+1)}{2}$ matrices vv^T, with $v \in \Delta^n$. Thus, in view of (3.3), X can then be written as the limit of a sequence $\{X^k\}$, where $X^k \in \mathrm{Diag}_+^n(U^k)$ for

each k. This together with $\mathrm{Diag}_+^n(U^k) \subseteq \mathcal{CP}^n$ shows that the sequence of sets $\{\mathrm{Diag}_+^n(U^k)\}$ converges to $\mathcal{CP}^n$ in the sense of Painlevé-Kuratowski [23, Chapter 4B].

Since the mapping $X \mapsto \mathcal{A}(X) := (\mathrm{tr}(A_1X), \ldots, \mathrm{tr}(A_mX))$ is surjective by Assumption **A**2 and (1.1) is strictly feasible, the vector b and the set $\mathcal{A}(\mathcal{CP}^n)$ cannot be separated in the sense of [23, Theorem 2.39]. Thus, [23, Theorem 4.32] shows that the sequence of feasible sets of (3.4) converges to the feasible set of (1.1) in the sense of Painlevé-Kuratowski.

It now follows from [23, Theorem 4.10(a)] and the nonemptiness of the feasible set of (1.1) that the feasible sets of (3.4) are nonempty for all sufficiently large k. Hence $\tilde{v}_p(U^k) < \infty$ for all sufficiently large k. Note that for each k, the dual problem to (3.4) is dual strictly feasible because of Assumption **A**3 and $\mathcal{COP}^n \subseteq (\mathrm{Diag}_+^n(U^k))^*$. Thus, $\tilde{v}_p(U^k)$ is indeed finite for all sufficiently large k. Moreover, thanks to the dual strict feasibility, the solution sets of (3.4) are nonempty whenever $\tilde{v}_p(U^k)$ is finite hence, in particular, are nonempty for all sufficiently large k.

Next, note that by Assumption **A**3 the dual problems of (3.4) for each k actually have a *common* Slater point, i.e., there exists a matrix

$$\bar{Y} := C - \sum_{i=1}^m \bar{y}_i A_i \in \mathrm{int}\, \mathcal{COP}^n \subseteq \mathrm{int}\, (\mathrm{Diag}_+^n(U^k))^*.$$

Therefore, there exists $\epsilon > 0$ so that $\bar{Y} + \epsilon \mathbf{B} \subseteq \mathrm{int}\, \mathcal{COP}^n$, where $\mathbf{B}$ is the unit closed ball centered at the origin (in Fröbenius norm). Consequently, for any $X \in \mathcal{CP}^n$, it holds that $\mathrm{tr}(\bar{Y}X) \geq \epsilon \|X\|_F$. We now argue that the solution sets of (3.4) are uniformly bounded for all k. Indeed, fix any k so that the solution set of (3.4) is nonempty, and let X^k be a solution. Then X^k is a Lagrange multiplier for the dual problem. In particular,

$$\tilde{v}_p(U^k) = \max_y \left\{ b^T y + \mathrm{tr}\left(X^k \left[C - \sum_{i=1}^m y_i A_i \right] \right) \right\} \geq b^T \bar{y} + \mathrm{tr}(X^k \bar{Y}) \geq b^T \bar{y} + \epsilon \|X^k\|_F,$$

where the last inequality holds because $X^k \in \mathrm{Diag}_+^n(U^k) \subseteq \mathcal{CP}^n$. Since $\{\tilde{v}_p(U^k)\}$ is nonincreasing, we conclude from the above inequality that $\{X^k\}$ can be bounded above by a constant independent of k. Thus, the solution sets of (3.4) are uniformly bounded for all k.

Finally, since the sequence of sets $\{\mathrm{Diag}_+^n(U^k)\}$ is monotonically increasing, we see from [23, Proposition 7.4(c)] that the objective function (with the constraint considered as the indicator function) of (3.4) epi-converges to that of (1.1) in the sense of [23, Definition 7.1]. The desired conclusion concerning limits of $\{\tilde{v}_p(U^k)\}$ and $\{X^k\}$ now follows from [23, Theorem 7.31(b)]. ■

Since $\mathrm{Diag}_+^n(U) \subseteq \mathrm{SDD}_+^G(U)$ if the edges of G cover all nodes, we get the convergence of the sequence of problems (3.1) for a positively enlarging sequence $\{(G^k, U^k)\}$ under the same assumptions on U^k. But we can actually obtain the desired convergence result under a weaker condition.

Theorem 5 *Assume that* (1.1) *is strictly feasible. Let* $\{(G^k, U^k)\}$ *be a positively enlarging sequence such that*

$$\lim_{k\to\infty} \max_{x \in \Delta^n} \min_{y \in \mathrm{seg}(G^k, U^k)} \|x - y\| = 0. \tag{3.5}$$

Then it holds that:

(i) $v_p(G^k, U^k)$ *is finite for all sufficiently large* k *and* $\lim_{k\to\infty} v_p(G^k, U^k) = v_p$.

(ii) *The solution set of* (3.1) *with* $(G, U) = (G^k, U^k)$ *is nonempty and uniformly bounded for all sufficiently large* k.

(iii) *Let* X^k *be a solution of* (3.1) *with* $(G, U) = (G^k, U^k)$ *whenever the solution set is nonempty. Then any accumulation point of* $\{X^k\}$ *is a solution of* (1.1).

Proof Note that from Theorem 2 and the description of $\mathrm{Diag}_+^n(U)$ as the conic hull of all matrices $u_i u_i^T$ where u_i is a row of U, if every node of G is covered by some edges and if we construct U' by adding rows such that each new row lies in $[u_i, u_j]$ for some $\{i,j\} \in \mathcal{E}$, we have $\mathrm{Diag}_+^n(U') \subseteq \mathrm{SDD}_+^G(U)$.

For each U^k, subdivide each segment $[u_i^k, u_j^k]$ into segments no longer than $1/k$, and add these new points to U^k to form $\tilde{U}^k \in \mathbb{R}_+^{\tilde{t}_k \times n}$. Then for each $x \in \Delta^n$, we have

$$\min_{i=1,\ldots,\tilde{t}_k} \|x - \tilde{u}_i^k\| \leq \min_{y \in \mathrm{seg}(G^k, U^k)} \|x - y\| + \frac{1}{k},$$

where $\tilde{u}_i^k$ is the i-th row of $\tilde{U}^k$. Thus, the sequence $\{\tilde{U}^k\}$ satisfies the conditions of Theorem 4. Consequently, from the proof of Theorem 4, the sequence of sets $\{\mathrm{Diag}_+^n(\tilde{U}^k)\}$ converges to $\mathcal{CP}^n$ in the sense of Painlevé-Kuratowski. In view of this and [23, Exercise 4.3(c)], $\{\mathrm{SDD}_+^n(U^k)\}$ converges to $\mathcal{CP}^n$. The rest of the proof follows exactly the same arguments as in the proof of Theorem 4. ■

An obvious way of guaranteeing the satisfaction of the condition (3.5) in Theorem 5 is to consider the rows of U^k to be the set of points in $x \in \Delta^n$ such that $kx \in \mathbb{Z}^n$, i.e. an equally spaced distribution of points in the simplex, with a growing number of points. This is in fact the strategy explored in [25] with the linear programming approach. As guaranteed by Theorem 5, this is sufficient to get convergence in our case, independently of the edges considered, but we can get away with much less. Indeed, it is easy to see, for example, that we do not need to map vertices to the interior of the simplex to get convergence and, in fact, it is enough to uniformly sample the *boundary* of the simplex, and form a graph with all possible edges between the chosen vertices. Finding embedded graphs that optimally cover Δ^n in the sense of minimizing the maximum distance to a point of the simplex seems to be a hard problem with no obvious answer, but many different strategies can be attempted. For practical purposes, it might be helpful to use the problem structure to design strategies for constructing $\{(G^k, U^k)\}$; these may not satisfy condition (3.5) and hence the convergence behavior can be compromised, but their corresponding problem (3.1) may be easier to solve. Indeed, as discussed in [18, Section 1.4], the amount of work per iteration for solving (3.2) is $\mathcal{O}((m + t_k^2)^2(4|\mathcal{E}| + t_k^2))$ when $(G, U) = (G^k, U^k)$. Hence, we will explore some problem-dependent inner approximation schemes in the next section.

Before ending this section, we would like to point out that the approach in [25] using $\mathrm{Diag}_+^n(U)$ for (rows of) U equally distributed in the simplex is one of the few problem-independent inner approximations to $\mathcal{CP}^n$ presented in the literature. The only other approach is that of [17], which leads to SDP problems. Although conceptually very interesting and with guaranteed convergence, this latter approach performs poorly in practice, because the size of the constraints grows very fast and the small instances that can be reasonably computed give weak approximations. In some sense, our SOCP based approximation schemes may lend some of the power of semidefinite programming to the LP approximation without completely sacrificing computability.

4 Problem-dependent inner approximation schemes

In this section, we propose some problem-dependent heuristic schemes for constructing $\{(G^k, U^k)\}$. They typically lead to computationally more tractable problems than a positively enlarging sequence satisfying (3.5). As we shall see later in our numerical experiments, these problem-dependent schemes in general return solutions with reasonable quality, though their convergence behaviors are still unknown. A related problem-dependent approach was developed in [2] for semidefinite programming. In there, they proposed the use of the cone $\mathrm{SDD}^n(U)$ and progressively enlarge the U to obtain efficient inner approximations to $\mathcal{S}_+^n$. We propose in this section a related approach. The main difference is that in the semidefinite case considered in [2], enlarging the U is relatively simple, as we can always separate the dual solution to the inner approximation from $\mathcal{S}_+^n$, if it is not there. In the case of completely positive cone, however, there is

no realistic way of even checking if the dual solution is copositive. Thus, a direct separation procedure, like the one proposed in [2], is not viable.

4.1 Problem-dependent positively enlarging sequence

In this section, we describe a problem-dependent strategy for constructing a positively enlarging sequence $\{(G^k, U^k)\}$ that can potentially perform better on specific problem instances.

After solving (3.1) with a choice of (G^k, U^k), if the problem is feasible, one will obtain a solution $X \in \mathrm{SDD}_+^G(U)$. By Theorem 2, this X can be written as a conic combination of vv^T for $v \in \mathrm{seg}(G, U)$. Our plan here is to add these v as vertices to G and add some new edges from them, in order to increment the graph. The decomposition is not unique, so one has to carefully define what is meant by it.

First, note that for an $M \in \mathcal{S}_+^2 \cap \mathcal{N}^2$, there exist $a \geq 0$, $b \geq 0$ and $v \in \mathbb{R}_+^2$ so that

$$M = vv^T + \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}. \tag{4.1}$$

This is trivially true if any element in the diagonal of M is zero. For other matrices, the above decomposition can be realized by taking for example $v = (\sqrt{m_{11}}, m_{12}/\sqrt{m_{11}})$, implying $a = 0$ and $b = m_{22} - m_{12}^2/m_{11}$, which is greater than or equal to zero since $M \succeq 0$.

Now, for any $U \in \mathbb{R}_+^{t\times n}$, one can see that $U^T \iota_{ij}(M) U = a u_i u_i^T + b u_j u_j^T + (v_1 u_i + v_2 u_j)(v_1 u_i + v_2 u_j)^T$, where u_i is the ith row of U, $1 \leq i < j \leq t$. So, besides the vertices u_i and u_j, we need at most one point coming from each edge $[u_i, u_j]$ to describe $U^T \iota_{ij}(M) U$. Since elements of $\mathrm{SDD}_+^G(U)$ are sums of matrices of this type for $\{i, j\} \in \mathcal{E}$ by Theorem 2, we have the following Lemma refining Theorem 2.

Lemma 1 *Any element $X \in \mathrm{SDD}_+^G(U)$ can be written as*

$$X = \sum_{i=1}^{t} \lambda_i u_i u_i^T + \sum_{\{i,j\}\in\mathcal{E}} \gamma_{ij} w_{ij} w_{ij}^T$$

where u_i is the i-th row of $U \in \mathbb{R}_+^{t\times n}$, $w_{ij} \in [u_i, u_j]$ and $\lambda_i, \gamma_{ij} \geq 0$. Indeed, for the first sum, it suffices to sum over the i's that are covered by some edges.

A natural question to ask is which points we can pick in each segment. To answer this question, we assume without loss of generality that $m_{12} > 0$ (and hence $m_{11} > 0$ and $m_{22} > 0$) in (4.1) and demonstrate how the v there can be chosen. Note that $U^T vv^T U$ is supposed to correspond to a $\gamma_{ij} w_{ij} w_{ij}^T$ in the decomposition in Lemma 1.

Since $m_{12} > 0$, we must have $v_1 > 0$ and $v_2 > 0$. Then we just need to see what the ratio $r = v_1/v_2$ can be. What we saw above right after (4.1) was the largest case, where we get $r = m_{11}/m_{12}$. The smallest it can get is attained by setting $v = (m_{12}/\sqrt{m_{22}}, \sqrt{m_{22}})$, which gives us $r = m_{12}/m_{22}$. These two values for r can be seen by noting that any extremal ratio v_1/v_2 for the v in (4.1) must correspond to $a = 0$ or $b = 0$. A balanced option, defined in a way that the ratio between diagonal entries of vv^T preserves the ratio between the diagonal entries of M, is to take

$$v = \sqrt{m_{12}} \begin{bmatrix} \left(\frac{m_{11}}{m_{22}}\right)^{\frac{1}{4}} \\ \left(\frac{m_{22}}{m_{11}}\right)^{\frac{1}{4}} \end{bmatrix}, \tag{4.2}$$

which corresponds to $r = \sqrt{m_{11}/m_{22}}$, the geometric mean of the largest and smallest possible ratios.

Based on these observations, we can now describe a general strategy for an iterative procedure to obtain upper bounds for (1.1).

Scheme 1: Successive upper bound scheme for (1.1)
Step 0. Start with a *complete* graph G^0 and its embedding (G^0, I) in Δ^n. Set $k = 0$ and $U^0 = I$.
Step 1. For an optimal solution X^k of (3.1) with $(G, U) = (G^k, U^k)$, apply Lemma 1 to obtain points w_{ij} for some $\{i, j\} \in \mathcal{E}' \subseteq \mathcal{E}$ such that $X = X^k$ is a conic combination of $w_{ij}w_{ij}^T$ for $\{i, j\} \in \mathcal{E}'$ and $u_i u_i^T$ for the vertex i of G.
Step 2. Define a new graph embedding (G^{k+1}, U^{k+1}) by adding new vertices at the points w_{ij} (or at least some subset of them) and some new edges connecting those vertices to some of the previously defined ones, and possibly remove redundant edges and go to Step 1.

The general idea is therefore to, augment the graph at each step by adding some vertices in the edges that were active in the optimal solution and some edges incident with them. All the steps have, however, some subtleties that need to be addressed.

The initial embedding (G^0, U^0) is currently taken to be simply the embedding of K^n into the vertices of Δ^n, so that $\mathrm{SDD}_+^{G^0}(U^0) = \mathrm{SDD}_+^n$. If that is infeasible, however, the strategy does not work. Nevertheless, assuming strict feasibility of (1.1), we know from Theorem 4 that there is some small enough uniform simplicial partition of Δ^n that will make the problem feasible.

The decomposition obtained in Step 1 is not unique. There are two sources of variations. First, as discussed above, given a 2×2 semidefinite matrix M such that $\iota_{ij}(M)$ appears in the decomposition of X, we have some leeway on which point to pick in the edge $[u_i, u_j]$. Second, notice that even these matrices M are not uniquely defined. Since the matrices M will be a side result of the solution to (3.1), the choice of algorithm and the way the problem is encoded will have some impact in the decomposition. As for defining the v given the matrix M, we will use the balanced approach described above in (4.2) as it seems to perform well in practice.

The augmenting step (Step 2) is the most delicate of all. Different augmenting techniques will give rise to very different procedures. Here and in our numerical experiments, we consider two different approaches. We will present more implementation details in Section 5.

The maximalist approach: In this approach, we add some new vertices and then connect all vertices to form a complete graph. This is memory consuming and induces some redundancies: every node we add is in the middle of an already existing edge. Adding edges to those does not enlarge the cone $\mathrm{SDD}_+^G(U)$ and might lead to numerical inaccuracies, as we create multiple ways of writing points in a segment. Some pruning techniques could be applied.

The adaptive simplicial partition approach: This is mimicking the technique introduced in [9], which maintains the set of edges as that of a simplicial partition. At every step we would pick edges to subdivide and subdivide all the simplices containing that edge. The choice of nodes and edges to add to G^k in our approach is based on the solution we obtain from solving (3.1) for $(G, U) = (G^{k-1}, U^{k-1})$. This is different from [9], which relies solely on an outer approximation to guide the subdivision process.

Note that we do not have any guarantee of convergence for Scheme 1. However, geometrically one can see what must happen in order for the method to get stuck, i.e., for $\mathrm{SDD}_+^{G^k}(U^k) = \mathrm{SDD}_+^{G^{k+1}}(U^{k+1})$. As an immediate consequence of Theorem 2, this happens if and only if all the newly added edges in the embedding are contained in previously existing edges. This is because rank one nonnegative matrices are on the extreme rays of $\mathcal{CP}^n$(see [1]). Thus, we see from Theorem 2 that $\mathrm{SDD}_+^{G^k}(U^k) = \mathrm{SDD}_+^{G^{k+1}}(U^{k+1})$ if and only if $\mathrm{seg}(G^k, U^k) = \mathrm{seg}(G^{k+1}, U^{k+1})$. This is an extremely strong condition, that implies essentially (depending on the scheme chosen to enlarge the graph) that the scheme gets stuck if for some iteration the optimal solution can be attained as a combination of only the nodes, and no elements from the edges. Or, in other words, the problem (3.1) has the same solution if we replace $\mathrm{SDD}_+^{G^k}(U^k)$ by $\mathrm{Diag}_+^n(U^k)$. On passing, we would like to point out that, in occasions where convergence is a serious concern, one can modify Step 2

of Scheme 1 by adding a random vertex in Δ^n in addition to those w_{ij}: this resulting scheme is guaranteed to converge in view of Theorem 5 if (1.1) is also strictly feasible.

4.2 A forgetfulness scheme

The use of a positively enlarging sequence $\{(G^k, U^k)\}$ can lead to large-scale SOCP problems when k is huge. As a heuristic to alleviate the computational complexity, we propose a simple forgetfulness scheme.

In this approach, we maintain the complete graph throughout. However, we always form U^k by appending only the newly generated vertices to U^0, which we choose to be the identity matrix. The details are described below.

Scheme 2: A forgetfulness upper bound scheme for (1.1)
Step 0. Start with a *complete* graph G^0 and its embedding (G^0, I) in Δ^n. Set $k = 0$ and $U^0 = I$.
Step 1. For an optimal solution X^k of (3.1) with $(G, U) = (G^k, U^k)$, apply Lemma 1 to obtain points w_{ij} for some $\{i, j\} \in \mathcal{E}' \subseteq \mathcal{E}$ such that $X = X^k$ is a conic combination of $w_{ij}w_{ij}^T$ for $\{i, j\} \in \mathcal{E}'$ and $u_i u_i^T$ for the vertex i of G.
Step 2. Define a new graph embedding (G^{k+1}, U^{k+1}): starting with (G^0, I), add new vertices at the points w_{ij} and then add edges between each new vertex and all vertices in G^0. Go to Step 1.

Note that, in general, one cannot guarantee that the forgetfulness scheme is even monotone, as we are dropping the factors $u_i u_i^T$ that were a part of the representation of the optimal solution X in Step 1. However, in most studied random instances in our numerical experiments, the forgetfulness scheme appears to be monotone. The main reason could be that the algorithm tends to write X as a conic combination of just the matrices $w_{ij}w_{ij}^T$ for $\{i, j\} \in \mathcal{E}'$. When this happens, we are guaranteed that the next iteration will be non-increasing, but this need not always be the case.

5 Numerical simulations

In this section, we report on numerical experiments to test our proposed approaches. All experiments were performed in Matlab (R2017a) on a 64-bit PC with an Intel(R) Core(TM) i7-6700 CPU (3.40GHz) and 16GB RAM. We used the convex optimization software CVX [14] (version 2.1), running the solver MOSEK (version 8.0.0.60) to solve the conic optimization problems that arise.

In our tests, we specifically consider the following strategies:

Δ-partition: In this approach, controlled by a parameter $k \geq 2$, we generate the vertices of the graph G^k as the $\binom{n+k-1}{k}$ vertices in the uniform subdivision of the simplex Δ^n into simplices of size $\frac{1}{k}\Delta^n$. We then add edges between two vertices whenever their supports differ by 2.

Note that by Theorem 5, if (1.1) is in addition strictly feasible, then $v_p(G^k, U^k)$ will be close to v_p for all sufficiently large k, so this strategy is guaranteed to converge as k increases.

Max: This is a variant of Scheme 1. Specifically, in Step 1, we decompose X^k as described in Lemma 1 using the balanced option given in (4.2). Then, in Step 2, we add to G^k as new vertices all w_{ij} whose corresponding entry X^k_{ij} is sufficiently large as new vertices, and add edges between all vertices so that the new graph G^{k+1} is complete.

Max1: This is another variant of Scheme 1. Step 1 is the same as in **Max**. However, in Step 2, we *only* add the w_{ij} corresponding to the largest X^k_{ij} (if X^k_{ij} exceeds a certain threshold) as a new vertex. We then add edges between all vertices so that the new graph G^{k+1} is complete.

Adaptive Δ-partition: This is also a variant of Scheme 1. Step 1 is the same as in **Max**. For Step 2, the way of adding vertices is the same as in **Max1**. However, the way we add edges mimics the approach introduced in [9], which maintains the set of edges as that of a simplicial partition. Specifically, we subdivide the edge corresponding to the w_{ij} we added, and subdivide all the simplices containing that edge.

Forgetfulness: This is a variant of Scheme 2. We perform Step 1 as in **Max**. As for Step 2, we add all w_{ij} whose X_{ij}^k is sufficiently large as new vertices to the original graph G^0. We then add edges to join each newly added vertex to all vertices in G^0.

In Section 5.1, we compare the strategies **Max**, **Adaptive Δ-partition** and **Forgetfulness** on random instances of (1.1). We will also present results obtained via **Δ-partition** (with $k = 2$) as benchmark. In Section 5.2, we will look at how **Forgetfulness** performs on standard quadratic programs. In Section 5.3, we will first review the standard completely positive programming formulation of the stable set problem, and then examine how **Max1** performs for some standard test graphs.

5.1 Random instances

In order to test the performance of our method in a generic setting, we test it for randomly generated instances of problem (1.1). We generate our objective function by setting $C = M^T M$ where M is an $n \times n$ matrix with i.i.d. standard Gaussian entries, guaranteeing strict feasibility of (1.3). Furthermore, we generate the constraints by setting $A_i = (M_i + M_i^T)/2$, where the M_i are also $n \times n$ matrices with i.i.d. standard Gaussian entries, and choosing b_i such that $b_i = \mathrm{tr}(A_i(E + nI))$. This guarantees strict feasibility of (1.1).

For the first of our tests we varied the number of variables, n, and the number of constraints m, so that n is either 10 or 25 and m is either 5, 10 or 15. Given the complexity of copositive programming, there is actually no reliable way to find the true solution for these problems and there is no available implemented method that can generate lower bounds with which to compare our results. As a work-around, throughout this section we will compare the results we obtain with the classical (and somewhat coarse) lower bound provided by replacing $\mathcal{CP}^n$ by $\mathcal{S}_+^n \cap \mathcal{N}^n$ in problem (1.1). We will use the difference of our approximations to this lower bound, normalized by dividing it by the bound, as a proxy for the quality of the methods, and will simply denote it by relative gap. Precisely, this quantity is defined by $\mathrm{gap}(x) = \frac{x - x^*}{|x^*|}$, where x is the objective value attained by the method being studied and x^* the doubly nonnegative lower bound. This makes it somewhat easier to compare different methods across different instances of the problem. The drawback is that the gap we compute is actually the sum of the gaps of the proposed method and the doubly nonnegative approximation, which we don't know how to independently estimate.

		Max		Adaptive Δ-Partition		Forgetfulness		Δ-Partition	
n	m	time(sec)	Relative Gap	time(sec)	Relative Gap	time(sec)	Relative Gap	time(sec)	Relative Gap
10	5	8.2	5.035e-02	25.3	6.362e-02	13.0	2.006e-02	4.6	4.620e-01
10	10	19.5	2.281e-02	25.3	7.920e-02	23.0	1.849e-02	4.5	4.095e-01
10	15	41.7	1.212e-02	27.6	8.207e-02	27.0	1.179e-02	5.0	2.995e-01
25	5	23.0	6.748e-01	55.1	5.828e-01	38.4	2.975e-01	—	—
25	10	45.8	4.660e-01	62.9	7.841e-01	52.7	2.020e-01	—	—
25	15	71.8	3.715e-01	56.1	8.565e-01	61.5	1.545e-01	—	—

Table 1 Comparison of different iterative approaches

The results obtained can be seen in Table 1, where we present both the average gaps and the average running time for the studied methods. A few technical details are needed to be able to replicate the experiment. The results presented are averages of 30 instances per parameter pair. Moreover we fix the maximum

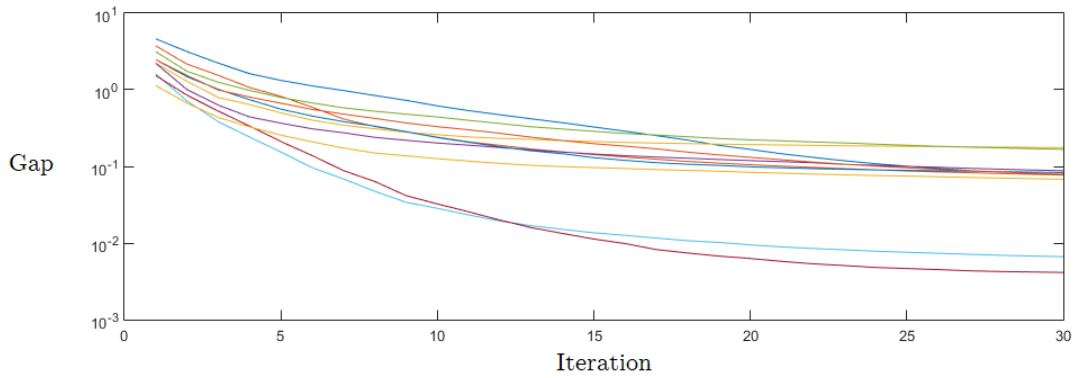

Fig. 2 Evolution of the gap for the forgetfulness scheme as iterations increase

number of iterations for the Max, Adaptive Δ-partition and Forgetfulness schemes as, respectively, 5, 20 and 15 for $n = 10$ and 5, 15 and 12 for $n = 25$. This was done (in an ad hoc way) to try to keep the average execution time as similar as possible across iterative methods, so that a fair comparison can be made. Also, since the maximalist approach can occasionally explode in size, we also stop this approach early when $t_{k+1} > 200$ (Recall that $U^k \in \mathbb{R}_+^{t_k \times n}$ for all k). For the forgetfulness approach, we prune the U^k in each step by removing redundant rows: we compute $\delta_{ij}^k := \|u_i^k - u_j^k\|_1$, where u_i^k and u_j^k are the i-th and the j-th rows of U^k respectively, $j > i$, and discard u_j^k if $\delta_{ij}^k < 10^{-6}$. We also stop this approach early when $t_{k+1} > 200$ for the U^{k+1} after pruning. The static Δ-partition is not computed for $n = 25$ as it takes too long.

These results show that the Forgetfulness scheme dominates the others in all categories as far as the relation quality/time is concerned. The relative gaps of the attained solutions jumps from between 1% and 2% for $n = 10$ to between 15% and 30% for $n = 25$. Once again, we stress that these are upper bounds for the Forgetfulness scheme quality as well as for the doubly nonnegative approximation quality, and we cannot separate the contributions from each method.

We also plot in Figure 2 the evolutions of the gaps for the Forgetfulness scheme for 10 random instances of the problem (1.1) with $n = 25$ and $m = 10$. We can see the logarithmic scale plot of the gap as iterations increase, and the diminishing returns in improvement percentage. Again, note that the true gap might actually be decreasing faster, as what we are seeing is the gap to the doubly nonnegative lower bound.

5.2 Standard Quadratic Program

We now focus on a class of more structured completely positive programs, those coming from standard quadratic programs. A standard quadratic optimization problem (SQP) consists of finding global minimizers for a quadratic form over the standard simplex. In other words, given $p(x) = x^T Qx$ for some $Q \in \mathcal{S}^n$, we want to find its minimum over the simplex $\Delta = \{x \in \mathbb{R}_+^n \ : \ \sum x_i = 1\}$. It is shown in [7] that this can be written as the completely positive program

$$\begin{aligned} p^* := \min \ & \operatorname{tr}(QX) \\ \text{s.t. } & \operatorname{tr}(EX) = 1, \\ & X \in \mathcal{CP}^n, \end{aligned} \tag{5.1}$$

where E is the all ones matrix. It is not difficult to see that these problems always verify the blanket assumptions presented in Section 1.1. Furthermore, since $\frac{1}{n} I_n$ is feasible, our hierarchy can always start from the base SDD relaxation.

To illustrate the behaviour of our method we start by taking the four concrete examples collected in [6] from several domains of application and applying the Forgetfulness scheme. We get the encouraging results shown in Table 2, where we can see the source of the examples, their size n, their true solution p^*, and the approximate solution obtained by the Forgetfulness scheme in 5 iterations (reported under the column approx.). We can see that the third example is the only one where there is a significant deviation from the optimal value, and all the results were attained in a few seconds.

Example	n	p^*	approx.
[6, Example 5.1] - independence number of pentagon	5	1/2	0.50000
[6, Example 5.2] - independence number of icosahedron complement	12	1/3	0.33333
[6, Example 5.3] - math. model of population genetics	5	$-16\frac{1}{3}$	-16.331
[6, Example 5.4] - portfolio optimization	5	0.4839	0.4839

Table 2 Applying the Forgetfulness scheme in four small SQP examples

To further explore the behaviour of our approach we followed the idea of [6] to generate random instances of SQP. In that paper they generate matrices Q to be 10×10, symmetric and with entries uniformly distributed in the interval $[0, 1]$. However, solving five thousand random examples of such problems to global optimality with CPLEX, we noticed that the true solutions seem to be commonly in the vertices of the simplex (48.5% of observed instances) or in edges (40.1% of observed instances). But in those two cases, by Theorem 1 our relaxation finds the optimal value at the first step. In other words, simply replacing $\mathcal{CP}^{10}$ by SDD_*^{10} gives us the exact solution in 88.6% of the times, with our iterative procedure only kicking in in the remaining instances.

To get a more meaningful test, we generated symmetric matrices Q with diagonal 1 and only off-diagonal entries uniformly distributed in the interval $[0, 1]$. Experimentally this virtually never gives rise to optimal solutions in the edges of the simplex, leading to non-trivial instances. We tested for both $n = 10$ and $n = 15$, comparing the results against the true value obtained using CPLEX. The parameters were chosen in the same way as in the previous section with the number of iterations being 15 for $n = 10$ and 12 for $n = 15$.

We ran 1000 instances for $n = 10$ and 100 for $n = 15$. The results are presented in Figure 3. On the top row we show the histograms for the ratios between the true value, computed using CPLEX, and our computed approximation, which provides an upper bound. We can see among other things that in both cases around half the instances were within 1% of the true value and four fifths were within 10%. If we want to more directly compare it with the results attained for the random instances in Table 1, one can compute the mean value of the true relative gap $\frac{\hat{p}-p^*}{p^*}$ where p^* is the true optimal value returned by CPLEX and $\hat{p}$ is the approximate value obtained by our approach. We get 4.823×10^{-2} and 5.747×10^{-2} for $n = 10$ and $n = 15$ respectively, very much in line to what we have seen before.

On the bottom row of Figure 3, as a rough reference, we have the boxplots of the CPU times (in seconds) taken by our method and CPLEX, presented here in logarithmic scale for readability. In both cases we can see that the Forgetfulness scheme is quite stable, as it will simply stop after a set number of iterations, while CPLEX has a huge number of outliers. While for $n = 10$ the exact CPLEX computation is faster, in $n = 15$ it becomes much slower, with several outliers taking many hours. For larger values of n it quickly becomes prohibitively slow compared to our approach.

5.3 Stable set problems

While in the previous section we focus on random problems, the main focus of the completely positive/copositive programming literature has been in highly structured combinatorial optimization problems. One of the most common applications is to the stable set problem, i.e., the problem of finding in a graph

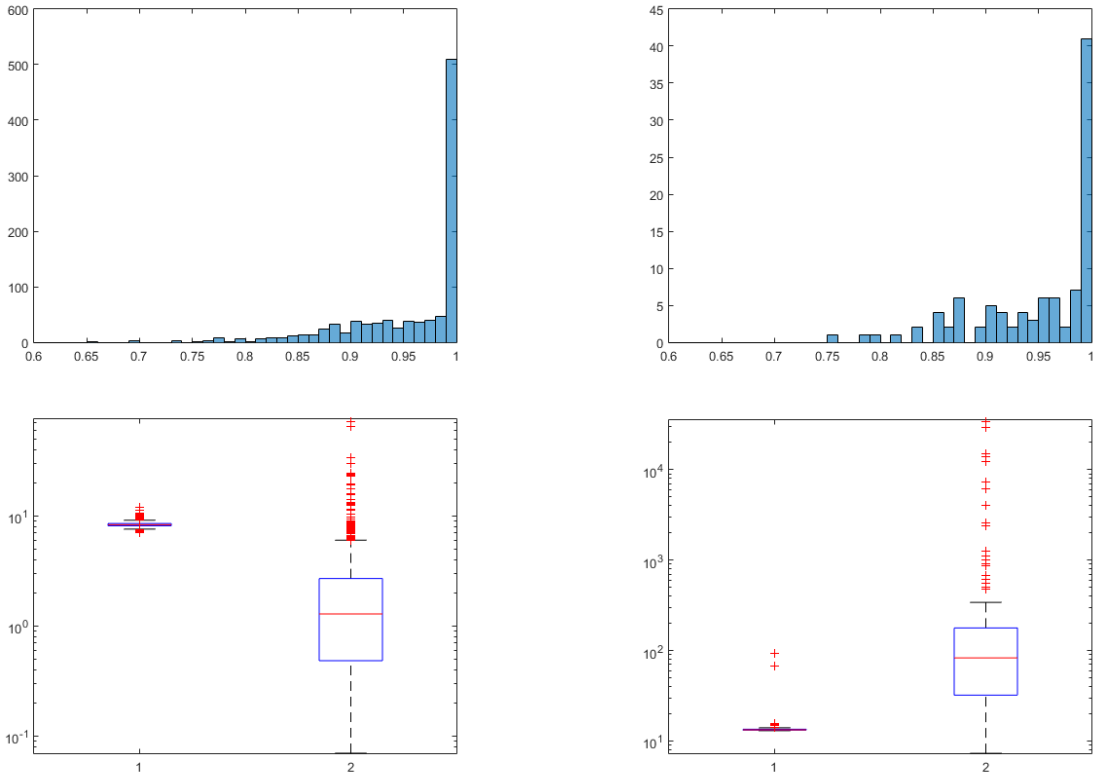

Fig. 3 Results of the random SQP tests for $n = 10$ (left) and $n = 15$ (right). The top graphs are histograms of the ratio between the true value and the attained upper bound, the bottom graphs box plots of the CPU times (in seconds) used by our method (1) and CPLEX (2).

G a set of vertices of maximal cardinality such that no two are connected with an edge. The cardinality of such a set is known as the independence number of G, denoted by $\alpha(G)$. In [12, Equation (8)], the following completely positive formulation was introduced for that problem.

$$\begin{aligned} \alpha(G) = \max\ & \operatorname{tr}(EX) \\ \text{s.t. } & \operatorname{tr}((A_G + I)X) = 1, \\ & X \in \mathcal{CP}^n, \end{aligned} \tag{5.2}$$

where A_G is the adjacency matrix of G.

In this setting we have a single constraint, so $m = 1$. Our inner approximations of $\mathcal{CP}^n$ will yield in this case lower bounds, from which one might be able to extract an actual feasible stable set with given cardinality. There are a number of good heuristic approaches to the stable set problem with good results, as there exist implementations of exact algorithms that can handle small to medium sized graphs, all performing necessarily much better than our all-purpose conic programming approach. However, we can still see how our approach performs on its own, to get some indication of its performance on low codimension structured problems.

In this class of problems, symmetry and structure likely imply that the growth of the matrix U in the greedier Maximalist approach but also in the Forgetfulness approach is too fast and adds too much redundancy. To avoid this phenomenon we take the Max1 approach: at every iteration we only add to U the vertex that has the largest weight in the solution found. This yields a greedy sort of algorithm, that

in practice tends to grow the stable set greedily one by one. We stopped as soon as the greedy process got stuck and there was no improvement in two consecutive iterations.

We computed both stability numbers, $\alpha(G)$, and clique numbers, $\omega(G)$, which are simply the stability numbers of the complementary graph. Following [9], we started by computing the clique numbers of the graphs where their method was tested. Our method yields the correct answers in a relatively short time, as can be seen in Table 3, where our results are presented under the column "result", and the column "$\omega(G)$" corresponds to the known clique numbers. Note that this is not too surprising, as finding a large stable set, or clique, is in a general sense computationally easier than proving that a larger one does not exist. In other words, lower bounding the stable set and clique numbers of particular graphs tends to be easier than upper bounding them, so our problem has a smaller scope than what was attempted in [9], leading to much faster times. The graphs in the table come from two sources, the first is a 17 vertex graph from [20] that is notoriously hard for upper bounding by convex approximations, the other five come from the 2nd DIMACS implementation challenge test instances [16], and only hamming6-4 and johnson8-2-4 could be solved by Bundfuss and Dür's method in less than two hours as reported in their paper [9].

graph	vertices	iterations	time(sec)	result	$\omega(G)$
pena17	17	5	13.8	6.0000	6
hamming6-2	64	31	836.7	32.0000	32
hamming6-4	64	3	64.0	4.0000	4
johnson8-2-4	28	3	11.7	4.0000	4
johnson8-4-4	70	13	322.5	14.0000	14
johnson16-2-4	120	7	637.0	8.0000	8

Table 3 Clique number for different graphs

To explore the limits of our approach we tried a few more instances of the stable set problem. We tried Paley graphs, known to mimic some properties of random graphs, with some degree of success, and a few small-sized instances of graphs derived from error correcting codes, available at [24]. The results are much worse in this family, with our algorithm failing in small instances, as can be seen in Table 4, where our results are reported under the column "result", and the true stability numbers are presented under the column "$\alpha(G)$". One word of caution is that the entire procedure is highly unstable, and simply changing the solver from MOSEK to SDPT3 can result in changes in the result, e.g. Paley_{137} becomes exact in SDPT3.

graph	vertices	iterations	time(sec)	result	$\alpha(G)$
Paley_{137}	137	4	977.4	5.0000	7
Paley_{149}	149	6	1841.6	7.0000	7
Paley_{157}	157	6	2254.1	7.0000	7
1tc.16	16	6	15.7	7.0000	8
1tc.32	32	10	85.5	11.0000	12
1dc.64	64	7	235.8	8.0000	10
1dc.128	128	13	2491.0	14.0000	16
2dc.128	128	4	823.6	5.0000	5

Table 4 Stability number for different graphs

References

1. Berman Abraham and Shaked-monderer Naomi. *Completely positive matrices.* World Scientific, 2003.

2. Amir Ali Ahmadi, Sanjeeb Dash, and Georgina Hall. Optimization over structured subsets of positive semidefinite matrices via column generation. *Discrete Optimization*, 24:129–151, 2017.
3. Amir Ali Ahmadi and Georgina Hall. *Sum of Squares Basis Pursuit with Linear and Second Order Cone Programming*. Contemporary Mathematics, 2017.
4. Amir Ali Ahmadi and Anirudha Majumdar. Dsos and sdsos optimization: more tractable alternatives to sum of squares and semidefinite optimization. *SIAM Journal on Applied Algebra and Geometry*, 3(2):193–230, 2019.
5. Erik G. Boman, Doron Chen, Ojas Parekh, and Sivan Toledo. On factor width and symmetric h-matrices. *Linear Algebra and its Applications*, 405:239–248, 2005.
6. Immanuel M. Bomze and Etienne de Klerk. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *Journal of Global Optimization*, 24(2):163–185, 2002.
7. Immanuel M. Bomze, Mirjam Dür, Etienne de Klerk, Cornelis Roos, Arie J. Quist, and Tamás Terlaky. On copositive programming and standard quadratic optimization problems. *Journal of Global Optimization*, 18(4):301–320, 2000.
8. Immanuel M. Bomze, Werner Schachinger, and Gabriele Uchida. Think co(mpletely) positive! Matrix properties, examples and a clustered bibliography on copositive optimization. *Journal of Global Optimization*, 52(3):423–445, 2012.
9. Stefan Bundfuss and Mirjam Dür. An adaptive linear approximation algorithm for copositive programs. *SIAM Journal on Optimization*, 20(1):30–53, 2009.
10. Samuel Burer. On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming*, 120(2):479–495, 2009.
11. Samuel Burer. Copositive programming. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 201–218. Springer, 2012.
12. Etienne de Klerk and Dmitrii V. Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM Journal on Optimization*, 12(4):875–892, 2002.
13. Mirjam Dür. Copositive programming – a survey. *Recent Advances in Optimization and its Applications in Engineering*, 320, 2010.
14. Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, 2014.
15. Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.
16. David J. Johnson and Michael A. Trick. Cliques, colorings and satisfiability. 2nd DIMACS implementation challenge, 1993. *American Mathematical Society*, 492:497, 1996.
17. Jean B. Lasserre. New approximations for the cone of copositive matrices and its dual. *Mathematical Programming*, 144(1):265–276, 2014.
18. Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *SIAM Journal on Optimization*, 12(4):875–892, 2002.
19. Pablo A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2000.
20. Javier Pena, Juan Vera, and Luis F. Zuluaga. Computing the stability number of a graph via linear and semidefinite programming. *SIAM Journal on Optimization*, 18(1):87–105, 2007.
21. Frank Permenter and Pablo Parrilo. Partial facial reduction: simplified, equivalent sdps via approximations of the psd cone. *Mathematical Programming*, 171:1–54, 2018.
22. Ralph Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
23. Ralph Tyrrell Rockafellar and J-B. Roger Wets. *Variational Analysis*. Springer, 1998.
24. Neil Sloane. Challenge problems: Independent sets in graphs. *Information Sciences Research Center, https://oeis.org/A265032/a265032.html*, 2005.
25. E. Alper Yıldırım. On the accuracy of uniform polyhedral approximations of the copositive cone. *Optimization Methods and Software*, 27(1):155–173, 2012.
26. E. Alper Yıldırım. Inner approximations of completely positive reformulations of mixed binary quadratic programs: a unified analysis. *Optimization Methods and Software*, 32(6):1163–1186, 2017.