A New Sparse SOS Decomposition Algorithm Based on Term Sparsity *

Jie Wang, Haokun Li and Bican Xia
School of Mathematical Sciences, Peking University
wangjie212@pku.edu.cn, ker@protonmail.ch, xbc@math.pku.edu.cn

ABSTRACT

A new sparse SOS decomposition algorithm is proposed based on a new sparsity pattern, called cross sparsity patterns. The new sparsity pattern focuses on the sparsity of terms and thus is different from the well-known correlative sparsity pattern which focuses on the sparsity of variables though the sparse SOS decomposition algorithms based on these two sparsity patterns both take use of chordal extensions/chordal decompositions. Moreover, it is proved that the SOS decomposition obtained by the new sparsity pattern is always a refinement of the block-diagonalization obtained by the sign-symmetry method. Various experiments show that the new algorithm dramatically saves the computational cost compared to existing tools and can handle some really huge polynomials.

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*Algebraic Algorithms*

General Terms

Algorithms, Theory

Keywords

nonnegative polynomial, sparse polynomial, cross sparsity pattern, sum of squares, chordal graph

1. INTRODUCTION

Certificates of nonnegative polynomials and polynomial optimization problems (POPs) arise from many fields such as mathematics, control, engineering, probability, statistics and physics. A classical method for these problems is using sums of squares (SOS). For a polynomial $f \in \mathbb{R}[\mathbf{x}] = \mathbb{R}[x_1,\ldots,x_n]$ and a given monomial basis $M = \{\mathbf{x}^{\omega_1},\ldots,\mathbf{x}^{\omega_r}\}$, the SOS condition for f can be converted to the problem of deciding if there exists a positive semidefinite matrix Q (Gram matrix) such that $f = M^T Q M$ which can be effectively solved by semidefinite programming (SDP) [20, 21].

When the given polynomial has many variables and a high degree, the corresponding SDP problem is hard to be dealt with by existing SDP solvers due to the very large size of the corresponding SDP matrix. On the other hand, most polynomials coming from practice have certain structures including symmetry and sparsity. So it is very important to take full advantage of structures of polynomials to reduce the size of corresponding SDP problems. In recent years, a lot of work has been done on this subject.

In the literature, there are three kinds of approaches to reduce computations by exploiting sparsity. One approach is reducing the size of the monomial basis M; such techniques include computing Newton polytopes [25], using the diagonal inconsistency [17], the iterative elimination method [15], and the facial reduction [23].The second approach is exploiting the non-diagonal sparsity of the Gram matrix Q; such techniques include using the sparsity of variables [6, 18, 19, 26, 28], using the symmetry property [9], using the split property [7], and minimal coordinate projections [24]. The third approach is exploiting the sparsity of constrained conditions of corresponding SDP problems, such as coefficient matching conditions [4, 13, 29].

In this paper, a new sparse SOS decomposition algorithm is proposed based on a new sparsity pattern, called *cross sparsity patterns*. Given a polynomial $f \in \mathbb{R}[\mathbf{x}]$ with the support $\mathscr{A} \subseteq \mathbb{N}^n$ and a monomial basis $M = \{\mathbf{x}^{\omega_1},\ldots,\mathbf{x}^{\omega_r}\}$, the cross sparsity pattern associated with $\mathscr{A}$ is represented by an $r \times r$ symmetric $(0,1)$-matrix $R_{\mathscr{A}}$ whose elements are defined by

$$R_{ij} = \begin{cases} 1, & \omega_i + \omega_j \in \mathscr{A} \cup 2\mathscr{B}, \\ 0, & \text{otherwise}, \end{cases} \tag{1.1}$$

where $2\mathscr{B} = \{2\omega_1,\ldots,2\omega_r\}$.

It can be seen that the new sparsity pattern focuses on the sparsity of *terms* and thus is different from the well-known correlative sparsity pattern [26] which focuses on the sparsity of *variables*. For example, for a polynomial $f \in \mathbb{R}[\mathbf{x}]$, if f contains a term involving all variables $x_1,\ldots,x_n$, then f is not sparse in the sense of correlative sparsity patterns and hence the corresponding SDP matrix for the SOS decomposition of f cannot be block-diagonalized. But f may still be sparse in the sense of cross sparsity patterns (see Example 3.4).

Following the chordal sparsity approaches, we associate the matrix $R_{\mathscr{A}}$ with an undirected graph $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ where

$$V_{\mathscr{A}} = \{1,2,\ldots,r\} \text{ and } E_{\mathscr{A}} = \{\{i,j\} \mid i,j \in V_{\mathscr{A}}, i < j, R_{ij} = 1\}$$

and generate a chordal extension of $G(V_{\mathscr{A}}, E_{\mathscr{A}})$. Then as usual, we use matrix decompositions for positive semidefinite matrices with chordal sparsity patterns to construct sets of supports for a blocking

*This work was supported partly by NSFC under grants 61732001 and 61532019.

SOS decomposition. We prove that the blocking SOS decomposition obtained by cross sparsity patterns is always a refinement of the block-diagonalization obtained by the sign-symmetry method [17].

We test the new algorithm on various examples. It turns out that the new algorithm dramatically reduces the computational cost compared to existing tools and can handle really huge polynomials which are unsolvable by any existing SDP solvers even exploiting sparsity.

The rest of this paper is organized as follows. Section 2 introduces some basic notions from nonnegative polynomials and graph theory. Section 3 defines a cross sparsity pattern matrix and a cross sparsity pattern graph associated with a sparse polynomial. We show that how we can exploit this sparsity pattern to obtain a block SOS decomposition for a sparse nonnegative polynomial. Moreover, we compare our approach with other methods to exploit sparsity in SOS decompositions, including correlative sparsity patterns and sign-symmetries. We discuss in Section 4 when the sparse SOS relaxation obtain the same optimal values as the dense SOS relaxation for polynomial optimization problems. The algorithm is given in Section 5. Section 6 includes numerical results on various examples. We show that the proposed `SparseSOS` algorithm exhibits a significantly better performance in practice. Finally, the paper is concluded in Section 7.

2. PRELIMINARIES

2.1 Nonnegative polynomials

Let $\mathbb{R}[\mathbf{x}] = \mathbb{R}[x_1,\dots,x_n]$ be the ring of real n-variate polynomials. For a finite set $\mathscr{A} \subset \mathbb{N}^n$, we denote by $\mathrm{conv}(\mathscr{A})$ the convex hull of $\mathscr{A}$, and by $V(\mathscr{A})$ the vertices of the convex hull of $\mathscr{A}$. Also we denote by $V(P)$ the vertex set of a polytope P. A polynomial $f \in \mathbb{R}[\mathbf{x}]$ can be written as $f(\mathbf{x}) = \sum_{\alpha\in\mathscr{A}} c_\alpha \mathbf{x}^\alpha$ with $c_\alpha \in \mathbb{R}, \mathbf{x}^\alpha = x_1^{\alpha_1}\cdots x_n^{\alpha_n}$. The support of f is defined by $\mathrm{supp}(f) = \{\alpha \in \mathscr{A} \mid c_\alpha \neq 0\}$, the degree of f is defined by $\deg(f) = \max\{\sum_{i=1}^n \alpha_i : \alpha \in \mathrm{supp}(f)\}$, and the Newton polytope of f is defined as $\mathrm{New}(f) = \mathrm{conv}(\{\alpha : \alpha \in \mathrm{supp}(f)\})$.

A polynomial $f \in \mathbb{R}[\mathbf{x}]$ which is nonnegative over $\mathbb{R}^n$ is called a *nonnegative polynomial*. The class of nonnegative polynomials is denoted by PSD, which is a convex cone.

A vector $\alpha \in \mathbb{N}^n$ is *even* if α_i is an even number for $i = 1,\dots,n$. A necessary condition for a polynomial $f(\mathbf{x})$ to be nonnegative is that every vertex of its Newton polytope is an even vector, i.e. $V(\mathrm{New}(f)) = V(\mathrm{supp}(f)) \subseteq (2\mathbb{N})^n$ [25].

For a nonempty finite set $\mathscr{B} \subseteq \mathbb{N}^n$, $\mathbb{R}[\mathscr{B}]$ denotes the set of polynomials in $\mathbb{R}[\mathbf{x}]$ whose supports are contained in $\mathscr{B}$, i.e., $\mathbb{R}[\mathscr{B}] = \{f \in \mathbb{R}[\mathbf{x}] \mid \mathrm{supp}(f) \subseteq \mathscr{B}\}$ and we use $\mathbb{R}[\mathscr{B}]^2$ to denote the set of polynomials which are sums of squares of polynomials in $\mathbb{R}[\mathscr{B}]$. The set of $r\times r$ symmetric matrices is denoted by S^r and the set of $r\times r$ positive semidefinite matrices is denoted by S^r_+. Let $\mathbf{x}^{\mathscr{B}}$ be the $|\mathscr{B}|$-dimensional column vector consisting of elements $\mathbf{x}^\beta, \beta \in \mathscr{B}$, then

$$\mathbb{R}[\mathscr{B}]^2 = \{(\mathbf{x}^{\mathscr{B}})^T Q \mathbf{x}^{\mathscr{B}} \mid Q \in S_+^{|\mathscr{B}|}\},$$

where the matrix Q is called the *Gram matrix*.

2.2 Chordal graphs

We introduce some basic notions from graph theory. A *graph* $G(V,E)$ consists of a set of nodes $V = \{1,2,\dots,r\}$ and a set of edges $E \subseteq V\times V$. A graph $G(V,E)$ is said to be *undirected* if and only if $(i,j) \in E \Leftrightarrow (j,i) \in E$. A *cycle* of length k is a sequence of nodes $\{v_1,v_2,\dots,v_k\} \subseteq V$ with $(v_k,v_1) \in E$ and $(v_i,v_{i+1}) \in E$, for $i = 1,\dots,k-1$. A *chord* in a cycle $\{v_1,v_2,\dots,v_k\}$ is an edge (v_i,v_j) that joins two nonconsecutive nodes in the cycle.

An undirected graph is called a *chordal graph* if all its cycles of length at least four have a chord. Chordal graphs include some common classes of graphs, such as complete graphs, line graphs and trees, and have applications in sparse matrix theory. Note that any non-chordal graph $G(V,E)$ can always be extended to a chordal graph $\widetilde{G}(V,\widetilde{E})$ by adding appropriate edges to E, which is called a *chordal extension* of $G(V,E)$. A *clique* $C \subseteq V$ is a subset of nodes where $(i,j) \in E$ for any $i,j \in C, i \neq j$. If a clique C is not a subset of any other clique, then it is called a *maximal clique*. It is known that maximal cliques of a chordal graph can be enumerated efficiently in linear time in the number of vertices and edges of the graph. See for example [5, 8, 10] for the details.

Given an undirected graph $G(V,E)$, we define an extended set of edges $E^\star := E \cup \{(i,i) \mid i \in V\}$ that includes all selfloops. Then, we define the space of symmetric sparse matrices as

$$S^r(E,0) := \{X \in S^r \mid X_{ij} = X_{ji} = 0 \text{ if } (i,j) \notin E^\star\} \tag{2.1}$$

and the cone of sparse PSD matrices as

$$S^r_+(E,0) := \{X \in S^r(E,0) \mid X \succeq 0\}. \tag{2.2}$$

Given a maximal clique C_k of $G(V,E)$, we define a matrix $P_{C_k} \in \mathbb{R}^{|C_k|\times r}$ as

$$(P_{C_k})_{ij} = \begin{cases} 1, & C_k(i) = j, \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

where $C_k(i)$ denotes the i-th node in C_k, sorted in the natural ordering. Note that $X_k = P_{C_k} X P_{C_k}^T \in S^{|C_k|}$ extracts a principal submatrix defined by the indices in the clique C_k, and $X = P_{C_k}^T X_k P_{C_k}$ inflates a $|C_k|\times|C_k|$ matrix into a sparse $r\times r$ matrix. Then, the following theorem characterizes the membership to the set $S^r_+(E,0)$ when the underlying graph $G(V,E)$ is chordal.

Theorem 2.1 ([1]). *Let $G(V,E)$ be a chordal graph and $\{C_1,\dots,C_t\}$ be all of the maximal cliques of $G(V,E)$. Then $X \in S^r_+(E,0)$ if and only if there exist $X_k \in S_+^{|C_k|}$ for $k = 1,\dots,t$ such that $X = \sum_{k=1}^t P_{C_k}^T X_k P_{C_k}$.*

3. EXPLOITING TERM SPARSITY IN SOS DECOMPOSITIONS

A convenient but incomplete algorithm for checking global nonnegativity of multivariate polynomials, as introduced by Parrilo in [20], is the use of sums of squares as a suitable replacement for nonnegativity. Given a polynomial $f(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$, if there exist polynomials $f_1(\mathbf{x}),\dots,f_m(\mathbf{x})$ such that

$$f(\mathbf{x}) = \sum_{i=1}^m f_i(\mathbf{x})^2, \tag{3.1}$$

then we say $f(\mathbf{x})$ is a *sum of squares* (SOS). The existence of an SOS decomposition of a given polynomial gives a certificate for its global nonnegativity. For $d \in \mathbb{N}$, let $\mathbb{N}^n_d := \{\alpha \in \mathbb{N}^n \mid \sum_{i=1}^n \alpha_i \le d\}$ and assume $f \in \mathbb{R}[\mathbb{N}^n_{2d}]$. The SOS condition (3.1) can be converted to the problem of deciding if there exists a positive semidefinite matrix Q such that

$$f(\mathbf{x}) = (\mathbf{x}^{\mathbb{N}^n_d})^T Q \mathbf{x}^{\mathbb{N}^n_d}, \tag{3.2}$$

which is a semidefinite programming (SDP) problem.

We say that a polynomial $f \in \mathbb{R}[\mathbb{N}^n_{2d}]$ is *sparse* if the number of elements in its support $\mathscr{A} = \mathrm{supp}(f)$ is much less than the number

of elements in $\mathbb{N}^n_{2d}$ that forms a support of fully dense polynomials in $\mathbb{R}[\mathbb{N}^n_{2d}]$. When $f(\mathbf{x})$ is a sparse polynomial in $\mathbb{R}[\mathbb{N}^n_{2d}]$, the size of the SDP problem (3.2) can be reduced by eliminating redundant elements from $\mathbb{N}^n_d$. In fact, $\mathbb{N}^n_d$ in problem (3.2) can be replaced by [25]

$$\mathscr{B} = \text{conv}(\{\frac{\alpha}{2} \mid \alpha \in V(\mathscr{A})\}) \cap \mathbb{N}^n \subseteq \mathbb{N}^n_d. \tag{3.3}$$

There are also other methods to reduce the size of $\mathscr{B}$ further, see for example [15, 23].

3.1 Cross sparsity pattern

To exploit the term sparsity of polynomials in SOS decompositions, we introduce the notion of cross sparsity patterns, which, roughly speaking, is measured by the different kinds of cross products of monomials arising in the objective polynomial $f(\mathbf{x})$.

Definition 3.1. *Let $f(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ with $supp(f) = \mathscr{A}$. Assume that $\mathbf{x}^{\mathscr{B}} = \{\mathbf{x}^{\omega_1}, \ldots, \mathbf{x}^{\omega_r}\}$ is a monomial basis. An $r \times r$* cross sparsity pattern matrix *$\mathbf{R}_{\mathscr{A}} = (R_{ij})$ is defined by*

$$R_{ij} = \begin{cases} 1, & \omega_i + \omega_j \in \mathscr{A} \cup 2\mathscr{B}, \\ 0, & otherwise, \end{cases} \tag{3.4}$$

where $2\mathscr{B} = \{2\omega_1, \ldots, 2\omega_r\}$.

Given a cross sparsity pattern matrix $\mathbf{R}_{\mathscr{A}} = (R_{ij})$, the graph $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ where

$$V_{\mathscr{A}} = \{1, 2, \ldots, r\} \text{ and } E_{\mathscr{A}} = \{\{i, j\} \mid i, j \in V_{\mathscr{A}}, i < j, R_{ij} = 1\}$$

is called the cross sparsity pattern graph.

To apply Theorem 2.1, we generate a chordal extension $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$ of the cross sparsity pattern graph $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ and use the extended cross sparsity pattern graph $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$ instead of $G(V_{\mathscr{A}}, E_{\mathscr{A}})$.

Remark 3.2. *Given a graph $G(V_{\mathscr{A}}, E_{\mathscr{A}})$, there may be many different chordal extensions and choosing anyone of them is valid for deriving the sparse SOS decompositions presented in this paper. For example, we can add edges to all of the connected components of $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ such that every connected component becomes a complete subgraph to obtain a chordal extension. The chordal extension with the least number of edges is called the* minimum chordal extension. *Finding the minimum chordal extension of a graph is an NP-hard problem in general. Finding a chordal extension of a graph is equivalent to calculating the symbolic sparse Cholesky factorization of its adjacency matrix. The resulted sparse matrix represents a chordal extension. The minimum chordal extension corresponds to the sparse Cholesky factorization with the minimum fill-ins. Fortunately, several heuristic algorithms, such as the minimum degree ordering, are known to efficiently produce a good approximation. For more information on symbolic Cholesky factorizations with the minimum degree ordering and minimum chordal extensions, see [2, 3, 12].*

3.2 Sparse SOS relaxations

Given $\mathscr{A} \subseteq \mathbb{N}^n$ with $V(\mathscr{A}) \subseteq (2\mathbb{N})^n$, assume that $\mathscr{B}$ is the support set of a monomial basis. Let the set of SOS polynomials supported on $\mathscr{A}$ be

$$\Sigma(\mathscr{A}) := \{f \in \mathbb{R}[\mathscr{A}] \mid \exists Q \in S^r_+ \text{ s.t. } f = (\mathbf{x}^{\mathscr{B}})^T Q \mathbf{x}^{\mathscr{B}}\}.$$

Generally the Gram matrix Q for a sparse SOS polynomial $f(\mathbf{x})$ can be dense. Let $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ be the cross sparsity pattern graph and $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$ a chordal extension. To maintain the sparsity of $f(\mathbf{x})$ in the Gram matrix Q, we consider a subset of SOS polynomials

$$\widetilde{\Sigma}(\mathscr{A}) := \{f \in \mathbb{R}[\mathscr{A}] \mid \exists Q \in S^r_+(\widetilde{E}_{\mathscr{A}}, 0) \text{ s.t. } f = (\mathbf{x}^{\mathscr{B}})^T Q \mathbf{x}^{\mathscr{B}}\}.$$

By virtue of Theorem 2.1, the following theorem gives the blocking SOS decompositions for polynomials in $\widetilde{\Sigma}(\mathscr{A})$.

Theorem 3.3. *Given $\mathscr{A} \subseteq \mathbb{N}^n$ with $V(\mathscr{A}) \subseteq (2\mathbb{N})^n$, assume that $\mathscr{B} = \{\omega_1, \ldots, \omega_r\}$ is the support set of a monomial basis and a chordal extension of the cross sparsity pattern graph is $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$. Let $C_1, C_2, \ldots, C_t \subseteq V_{\mathscr{A}}$ denote the maximal cliques of $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$ and $\mathscr{B}_k = \{\omega_i \in \mathscr{B} \mid i \in C_k\}, k = 1, 2, \ldots, t$. Then, $f(\mathbf{x}) \in \widetilde{\Sigma}(\mathscr{A})$ if and only if there exist $f_k(\mathbf{x}) \in \mathbb{R}[\mathscr{B}_k]^2$ for $k = 1, \ldots, t$ such that*

$$f(\mathbf{x}) = \sum_{k=1}^{t} f_k(\mathbf{x}). \tag{3.5}$$

Proof. By Theorem 2.1, $Q \in S^r_+(\widetilde{E}_{\mathscr{A}}, 0)$ if and only if there exist $Q_k \in S^{|C_k|}_+, k = 1, \ldots, t$ such that $Q = \sum_{k=1}^t P_{C_k}^T Q_k P_{C_k}$. So $f(\mathbf{x}) \in \widetilde{\Sigma}(\mathscr{A})$ if and only if there exist $Q_k \in S^{|C_k|}_+, k = 1, \ldots, t$ such that

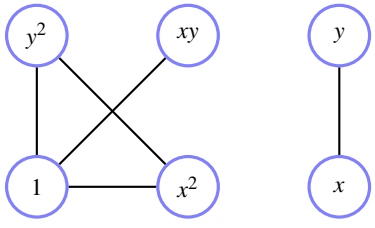
$$\begin{aligned} f(\mathbf{x}) &= (\mathbf{x}^{\mathscr{B}})^T (\sum_{k=1}^{t} P_{C_k}^T Q_k P_{C_k}) \mathbf{x}^{\mathscr{B}} \\ &= \sum_{k=1}^{t} (P_{C_k} \mathbf{x}^{\mathscr{B}})^T Q_k (P_{C_k} \mathbf{x}^{\mathscr{B}}) \\ &= \sum_{k=1}^{t} (\mathbf{x}^{\mathscr{B}_k})^T Q_k \mathbf{x}^{\mathscr{B}_k}, \end{aligned}$$

which is equivalent to that there exist $f_k(\mathbf{x}) \in \mathbb{R}[\mathscr{B}_k]^2$ for $k = 1, \ldots, t$ such that $f(\mathbf{x}) = \sum_{k=1}^t f_k(\mathbf{x})$. □

3.3 Comparison with correlative sparsity patterns

The notion of correlative sparsity patterns was introduced by Waki et al. [26] to exploit variable sparsity of polynomials in SOS programming, which also takes use of chordal extensions/chordal decompositions. An interpretation of correlative sparsity patterns in terms of the sparsity of Gram matrices was recently given in [30]. It should be emphasized that the angles of correlative sparsity patterns and cross sparsity patterns to exploit sparsity are different. Correlative sparsity patterns focus on the sparsity of variables, while cross sparsity patterns focus on the sparsity of terms. For example, for a polynomial $f \in \mathbb{R}[\mathbf{x}]$, if f contains a term involving all variables $x_1, \ldots, x_n$, then f is not sparse in the sense of correlative sparsity patterns and hence the corresponding SDP matrix for the SOS decomposition of f cannot be block-diagonalized. But f may still be sparse in the sense of cross sparsity patterns.

Example 3.4. *Consider the polynomial $f = x^2y^2 + x^2 + y^2 + 1 - xy$. A monomial basis for f is $\{1, x, y, xy, x^2, y^2\}$. The correlative sparsity pattern graph of f is a complete graph, and hence the corresponding Gram matrix of f cannot be blocked. On the other hand, the cross sparsity pattern graph of f has three maximal cliques, corresponding to $\{1, x^2, y^2\}$, $\{1, xy\}$ and $\{x, y\}$ respectively. Hence, the corresponding Gram matrix of f can be blocked into one 3×3 submatrix and two 2×2 submatrices.*



3.4 Comparison with sign-symmetries

In [17], sign-symmetries are exploited to block diagonalize sums of squares programming ([17, Theorem 3]), which is implemented in `Yalmip`. Given a polynomial $f \in \mathbb{R}[\mathbf{x}]$ with $\mathrm{supp}(f) = \mathscr{A}$. The *sign-symmetries* of f are defined by all vectors $\mathbf{r} \in \{0,1\}^n$ such that $\mathbf{r}^T\alpha \equiv 0 \pmod 2$ for all $\alpha \in \mathscr{A}$.

By virtue of sign-symmetries, SOS programming can be blocked as follows.

Theorem 3.5 ([17]). *Given a polynomial $f \in \mathbb{R}[\mathbf{x}]$ with $\mathrm{supp}(f) = \mathscr{A}$, assume that $\mathscr{B} = \{\omega_1, \ldots, \omega_r\}$ is the support set of a monomial basis and the sign-symmetries of f are defined by the binary matrix $R = [\mathbf{r}_1, \ldots, \mathbf{r}_s]$. Then $\mathbf{x}^{\mathscr{B}}$ can be blocked in the SOS programming of f and $\mathbf{x}^{\omega_i}, \mathbf{x}^{\omega_j}$ belong to the same block if and only if $R^T\omega_i \equiv R^T\omega_j \pmod 2$.*

We show in Theorem 3.6 that the blocking decomposition obtained by cross sparsity patterns is always a refinement of the block-diagonalization obtained by sign-symmetries.

Theorem 3.6. *Given a polynomial $f \in \mathbb{R}[\mathbf{x}]$ with $\mathrm{supp}(f) = \mathscr{A}$, assume that $\mathscr{B} = \{\omega_1, \ldots, \omega_r\}$ is the support set of a monomial basis and the sign-symmetries of f are defined by the binary matrix $R = [\mathbf{r}_1, \ldots, \mathbf{r}_s]$. Then the blocking decomposition obtained by cross sparsity patterns is a refinement of the block-diagonalization obtained by sign-symmetries in the SOS programming of f.*

Proof. The block-diagonalization obtained by sign-symmetries can be represented by a graph $\overline{G}(V, \overline{E})$ with $V = \{1, \ldots, r\}$ and $(i,j) \in \overline{E}$ if and only if $R^T\omega_i \equiv R^T\omega_j \pmod 2$. Then by Theorem 3.5, the blocks obtained by sign-symmetries correspond to the connected components of $\overline{G}(V, \overline{E})$. To show that the blocking decomposition obtained by cross sparsity patterns is a refinement of the block-diagonalization obtained by sign-symmetries, we only need to prove that the cross sparsity pattern graph $G(V, E_{\mathscr{A}})$ is a subgraph of $\overline{G}(V, \overline{E})$, i.e. $E_{\mathscr{A}} \subseteq \overline{E}$.

By the definition of sign-symmetries, we have $R^T\alpha \equiv \mathbf{0} \pmod 2$ for all $\alpha \in \mathscr{A}$. By the definition of cross sparsity pattern graphs, $(i,j) \in E_{\mathscr{A}}$ if and only if $\omega_i + \omega_j \in \mathscr{A} \cup 2\mathscr{B}$. So if $(i,j) \in E_{\mathscr{A}}$, then either $\omega_i + \omega_j \in \mathscr{A}$ or $\omega_i + \omega_j \in 2\mathscr{B}$. In anyone of these two case, we always have $R^T(\omega_i + \omega_j) \equiv \mathbf{0} \pmod 2$, which is equivalent to $R^T\omega_i \equiv R^T\omega_j \pmod 2$. Thus $(i,j) \in \overline{E}$ as desired. □

4. WHEN DO $\Sigma(\mathscr{A})$ AND $\widetilde{\Sigma}(\mathscr{A})$ COINCIDE

Given $\mathscr{A} \subseteq \mathbb{N}^n$ with $V(\mathscr{A}) \subseteq (2\mathbb{N})^n$, we define in Section 3.2 two sets of SOS polynomials: $\Sigma(\mathscr{A})$ and $\widetilde{\Sigma}(\mathscr{A})$. Generally we have $\Sigma(\mathscr{A}) \supseteq \widetilde{\Sigma}(\mathscr{A})$. If $\Sigma(\mathscr{A}) = \widetilde{\Sigma}(\mathscr{A})$, then the sparse SOS relaxation and the dense SOS relaxation obtain the same optimal value for the optimization of a polynomial f with the support $\mathscr{A}$. The following theorem shows that in the quadratic case the equality $\Sigma(\mathscr{A}) = \widetilde{\Sigma}(\mathscr{A})$ holds.

Theorem 4.1. *If for any $\alpha \in \mathscr{A}$, $\sum_{i=1}^n \alpha_i \leq 2$, then $\Sigma(\mathscr{A}) = \widetilde{\Sigma}(\mathscr{A})$.*

Proof. Suppose $f \in \Sigma(\mathscr{A})$ is a quadratic polynomial with $\mathrm{supp}(f) = \mathscr{A}$. Let $M = [1, x_1, \ldots, x_n]$ be a monomial basis and assume $f = M^T Q M$ for a positive semidefinite matrix $Q = (q_{ij})_{i,j=0}^n$. Let $\mathbf{R} = (R_{ij})_{i,j=0}^n$ be the corresponding cross sparsity pattern matrix for f. To prove $\Sigma(\mathscr{A}) \subseteq \widetilde{\Sigma}(\mathscr{A})$, we need to show $Q \in S_+^{n+1}(\widetilde{E}_{\mathscr{A}}, 0)$, or $Q \in S_+^{n+1}(E_{\mathscr{A}}, 0)$. Note that $Q \in S_+^{n+1}(E_{\mathscr{A}}, 0)$ is equivalent to the proposition that $R_{ij} = 0$ implies $q_{ij} = 0$ for all i, j. Let $\{\mathbf{e}_k\}_{k=1}^n$ be the standard basis. If $i = 0, j > 0$, from $R_{0j} = 0$ we have $\mathbf{e}_j \notin \mathscr{A}$. If $i > 0, j = 0$, from $R_{i0} = 0$ we have $\mathbf{e}_i \notin \mathscr{A}$. If $i, j > 0, i \neq j$, from $R_{ij} = 0$ we have $\mathbf{e}_i + \mathbf{e}_j \notin \mathscr{A}$. In anyone of these three cases, we have $q_{ij} = 0$ as desired. □

5. ALGORITHM

According to Section 3, a sparse SOS decomposition procedure can be easily divided into the following four steps:

1. Compute the support set of a monomial basis $\mathscr{B}$;
2. Generate the cross sparsity pattern graph $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ and a chordal extension $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$;
3. Compute all of the maximal cliques of $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$ and obtain the blocking SOS problem;
4. Use an SDP solver to solve the blocking SOS problem.

In step 1, we compute the support set of a monomial basis $\mathscr{B}$ following the method in [17].

In step 2, different chordal extensions will lead to different blocking SOS decompositions. When implementing this step, we obtain a chordal extension $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$ by adding edges to $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ such that every connected component becomes a complete subgraph.

The above procedure is formally stated as Algorithm 1 (named `SparseSOS`) in the following. Obviously, since we use well-known and popular methods and tools for Step 1 and Step 4, the efficiency of `SparseSOS` essentially depends on Step 2 and Step 3. That is, if we may decompose the original problems into smaller subproblems via Step 2 and Step 3, the computation cost will certainly be decreased because the SDP solver in Step 4 receives smaller inputs. We will show in the next section that `SparseSOS` performs well on many examples.

Algorithm 1 `SparseSOS`

input: a polynomial f with $\mathrm{supp}(f) = \mathscr{A}$
output: a representation $f = \sum_{i=1}^m g_i^2$ or unknown

1: Compute the support set of a monomial basis $\mathscr{B} = \{\omega_1, \ldots, \omega_r\}$;
2: Generate the cross sparsity pattern graph $G(V_{\mathscr{A}}, E_{\mathscr{A}})$;
3: Take the connected components $\{C_1, \ldots, C_t\}$ of $G(V_{\mathscr{A}}, E_{\mathscr{A}})$ to obtain a chordal extension $\widetilde{G}(V_{\mathscr{A}}, \widetilde{E}_{\mathscr{A}})$;
4: Solve the blocking SOS problem

$$f = \sum_{k=1}^{t} f_k, \quad f_k \in \mathbb{R}[\mathscr{B}_k]^2, \quad (*)$$

where $\mathscr{B}_k = \{\omega_i \in \mathscr{B} \mid i \in C_k\}, k = 1, 2, \ldots, t$;
5: If $(*)$ is feasible, then **return** $f = \sum_{i=1}^m g_i^2$. Otherwise **return** unknown.

6. NUMERICAL EXPERIMENTS

In this section, we give numerical results to illustrate the effectiveness of the algorithm `SparseSOS`. The algorithm is implemented with C++ as a tool also named `SparseSOS`. It turns out that `SparseSOS` is extremely powerful and can deal with some really huge polynomials that cannot be handled by other tools.

6.1 Versions and Commands

Our tool `SparseSOS` can be downloaded at
https://gitlab.com/haokunli/sparsesos.
All the examples in the following subsections can be downloaded there as well. We illustrate by a very simple example how to

use SparseSOS. Suppose we want to check whether the following polynomial is SOS by SparseSOS:

$$36x_0^{10}x_1^2+4x_0^{10}x_2^2+81x_0^2x_1^8x_2^2-84x_0^8x_1^3+18x_0^2x_1^8x_2+49x_0^6x_1^4 +x_0^2x_1^8+36x_0^4x_1^4x_2^2+4x_0^4x_1^4x_2+4x_0^6x_2^2.$$

First, express the polynomial by $+,-,*,\hat{}$, integers and variables in a file, say example.txt, as follows:

```
36*x0^10*x1^2 + 4*x0^10*x2^2 + 81*x0^2*x1^8*x2^2 -
84*x0^8*x1^3 + 18*x0^2*x1^8*x2 +49*x0^6*x1^4
+ x0^2*x1^8 + 36*x0^4*x1^4*x2^2 + 4*x0^4*x1^4*x2
+ 4*x0^6*x2^2.
```

Then, we only need to type in:

```
is_sos example.txt
```

to run SparseSOS on the example.

SparseSOS uses mosek 8.1 as an LP solver and csdp 6.2 as an SDP solver. In the following subsections, we compare the performance on some examples of SparseSOS with that of Yalmip [16], SOSTOOLS [22], and SparsePOP [27] which also exploit sparsity in SOS decompositions. The versions of the tools and their LP and SDP solvers are listed here: Yalmip R20181012 (LP solver: gurobi 8.1; SDP solver: mosek 8.1), SOSTOOLS303 (SDP solver: sdpt 3.4) and SparsePOP301 (SDP solver: sdpt 3.4).

All numerical examples were computed on a 6-Core Intel Core i7-8750H@2.20GHz CPU with 16GB RAM memory and ARCH LINUX SYSTEM.

6.2 The polynomials B_m

Let

$$B_m=\left(\sum_{i=1}^{3m+2}x_i^2\right)\left(\left(\sum_{i=1}^{3m+2}x_i^2\right)^2-2\sum_{i=1}^{3m+2}x_i^2\sum_{j=1}^{m}x_{i+3j+1}^2\right),$$

where we set $x_{3m+2+r}=x_r$. Note that B_m is modified from [20]. For any $m\in\mathbb{N}\backslash\{0\}$, B_m is homogeneous and is an SOS polynomial. For these B_m's, SparseSOS dramatically reduces the problem sizes and the computation time (see Table 2).

Remark 6.1. *It is easy to see that, for $m\le 4$,* SOSTOOLS *and* SparsePOP *cannot block-diagonalize the corresponding Gram matrices for B_m while* Yalmip *and our tool* SparseSOS *reduce the Gram matrices to smaller submatrices of the same size. That is the reason why* Yalmip *and* SparseSOS *cost much less time on those problems. For $m\ge 5$, only* SparseSOS *can work out results and* Yalmip *fails to obtain a block-diagonalization.*

6.3 MCP polynomials $P_{i,j}$

Monotone Column Permanent (MCP) Conjecture was given in [11]. In the dimension 4, this conjecture is equivalent to decide whether particular polynomials $p_{1,2},p_{1,3},p_{2,2},p_{2,3}$ are nonnegative (the definitions of $p_{i,j}$ can be found in [14]). Actually, it was proved that every $p_{i,j}$ multiplied by a small particular polynomial is an SOS polynomial ([14]). Let

$$\begin{aligned}P_{1,2}&=(a^2+2b^2+c^2)\cdot p_{1,2},\\ P_{1,3}&=p_{1,3},\\ P_{2,2}&=(a^2+2b^2+c^2)\cdot p_{2,2},\\ P_{2,3}&=(a^2+2b^2+c^2)\cdot p_{2,3}.\end{aligned}$$

We use SparseSOS to certify nonnegativity of $P_{1,2},P_{1,3},P_{2,2},P_{2,3}$. The result is listed in Table 3.

Remark 6.2. *When we use the 'sparse' option,* SOSTOOLS *seems to make a mistake in computing a monomial basis for $P_{2,2}$ and fails to obtain a SOS decomposition for $P_{2,2}$.*

6.4 Randomly generated polynomials

Now we present the numerical results for randomly generated polynomials. A sparse randomly generated polynomial

$$f=\sum_{i=1}^{k}f_i^2\in\mathbf{randpoly}(n,d,k,p)$$

is constructed as follows: first generate a set of monomials M in the set $\mathbf{x}^{\mathbb{N}_d^n}$ with probability p, and then randomly assign the elements of M to $f_1,\dots,f_k$ with random coefficients between -10 and 10. We generate 18 random polynomials $F_1,\dots,F_{18}$ from 6 different classes, where

$$F_1,F_2,F_3\in\mathbf{randpoly}(10,6,10,0.01),$$

$$F_4,F_5,F_6\in\mathbf{randpoly}(10,6,10,0.015),$$

$$F_7,F_8,F_9\in\mathbf{randpoly}(10,10,10,0.001),$$

$$F_{10},F_{11},F_{12}\in\mathbf{randpoly}(10,8,20,0.002),$$

$$F_{13},F_{14},F_{15}\in\mathbf{randpoly}(10,8,20,0.005)$$

and

$$F_{16},F_{17},F_{18}\in\mathbf{randpoly}(10,8,20,0.01).$$

See Table 4 for the performance of Yalmip and SparseSOS on these polynomials. Since SOSTOOLS and SparsePOP can hardly handle these polynomials, we do not list the performance of them in the table.

Remark 6.3. *From Table 4, we can see that* SparseSOS *obtains block-diagonalizations for $F_1,\dots,F_{15}$ while* Yalmip *fails for all these polynomials. For polynomials F_{16},F_{17},F_{18}, both* SparseSOS *and* Yalmip *cannot obtain block-diagonalizations.*

For $F_1,\dots,F_6$ and $F_{10},\dots,F_{12}$, SparseSOS *succeeds in obtaining the final SOS decompositions while* Yalmip *fails on F_4,F_5 and F_{10}. Furthermore,* SparseSOS *is faster than* Yalmip *on all these polynomials except F_6. We observe that the reason why* Yalmip *is faster on F_6 lies in the efficiency of SDP solvers. Mosek is faster on F_6 than csdp.*

Although we select only three polynomials from each class of random polynomials, we notice that SparseSOS *performs similarly on polynomials from the same class. For example, for the classes* ***randpoly****$(10,6,10,0.01)$,* ***randpoly****$(10,6,10,0.015)$, and* ***randpoly*** $(10,8,20,0.002)$, SparseSOS *succeeds in obtaining the final SOS decompositions. For the classes* ***randpoly****$(10,10,10,0.001)$ and* ***randpoly****$(10,8,20,0.005)$,* SparseSOS *can obtain block-diagonalizations of the corresponding Gram matrices but cannot work out the final result. For the class* ***randpoly****$(10,8,20,0.01)$,* SparseSOS *cannot obtain block-diagonalizations.*

7. CONCLUSIONS

We exploit the term sparsity of polynomials in SOS Programming by virtue of cross sparsity patterns and prove a sparse SOS decomposition theorem for sparse polynomials via PSD matrix decompositions with chordal sparsity patterns. Based on this, a new sparse SOS algorithm is proposed and is tested on various examples. The experimental results show that the new algorithm is efficient and extremely powerful. The algorithm can be combined

Table 1: Notation

#supp	the number of support monomials of a polynomial
#block	the size of blocks obtained by SparseSOS
$i \times j$	i blocks of size j
*	a failure information to obtain a SOS decomposition
OM	an out-of-memory error

Table 2: Results for B_m

		SparseSOS		Yalmip		SOSTOOLS		SparsePOP	
m	#supp	#block	time	#block	time	#block	time	#block	time
1	35	$5\times5, 10\times1$	0.01s	$5\times5, 10\times1$	0.45s	1×35	0.95s	1×56	0.54s
2	104	$8\times8, 56\times1$	0.04s	$8\times8, 56\times1$	0.95s	1×120	2.59s	1×165	4.66s
3	242	$11\times11, 165\times1$	0.15s	$11\times11, 165\times1$	1.18s	1×286	34.00s	1×364	93.9s
4	476	$14\times14, 364\times1$	0.45s	$14\times14, 364\times1$	2.94s	1×560	423s	1×680	764s
5	833	$17\times17, 680\times1$	1.56s	1×969	OM	1×969	OM	OM	
10	5408	$32\times32, 4960\times1$	65.55s						

Table 3: Results for $P_{i,j}$

		SparseSOS		Yalmip		SOSTOOLS		SparsePOP	
	#supp	#block	time	#block	time	#block	time	#block	time
$P_{1,2}$	159	$1\times15, 2\times12, 7\times4,$ $1\times3, 2\times2, 3\times1$	0.29s	$1\times15, 2\times12, 7\times4,$ $1\times3, 2\times2, 3\times1$	1.86s	1×77	2.39s	1×112	2.56s
$P_{1,3}$	53	$1\times8, 4\times3,$ $2\times2, 5\times1$	0.08s	$1\times8, 4\times3,$ $2\times2, 5\times1$	0.41s	1×29	0.86s	$2\times30, 1\times29$	0.52s
$P_{2,2}$	144	$3\times12, 2\times4,$ $8\times2, 2\times1$	0.27s	$3\times12, 2\times4,$ $8\times2, 2\times1$	0.40s	1×25	*	1×97	2.23s
$P_{2,3}$	107	$2\times10, 1\times8, 1\times4,$ $1\times3, 8\times2, 2\times1$	0.19s	$2\times10, 1\times8, 1\times4,$ $1\times3, 8\times2, 2\times1$	0.40s	1×53	1.62s	$1\times65, 1\times60$	1.48s

Table 4: The result for randomly generated polynomials

		SparseSOS		Yalmip				SparseSOS		Yalmip	
	#supp	#block	time	#block	time		#supp	#block	time	#block	time
F_1	590	$187, 5,$ $6\times2, 44\times1$	179.2s	248	315.60s	F_4	873	$303, 8,$ $3\times2, 40\times1$	1850.54s	357	OM
F_2	310	$83, 3,$ $4\times2, 37\times1$	4.42s	131	16.34s	F_5	709	$238, 4,$ $4\times3, 12, 55\times1$	633.51s	331	OM
F_3	504	$162, 6, 4,$ $6\times2, 34\times1$	63.86s	218	116.09s	F_6	927	$231, 3,$ $2\times2, 23\times1$	470.40s	261	297.40s
F_7	1344	$4658, 7, 2\times5, 3\times4$ $7\times3, 16\times2, 29\times1$	OM	4769	OM	F_{10}	306	$110, 10, 6, 3\times4,$ $5\times3, 22\times2, 192\times1$	29.95s	389	OM
F_8	1392	$5012, 5, 3,$ $3\times2, 20\times1$	OM	5046	OM	F_{11}	255	$62, 8, 5, 4,$ $2\times3, 2\times2, 131\times1$	32.09s	220	185.35s
F_9	1845	$4528, 7, 3,$ $5\times2, 28\times1$	OM	4576	OM	F_{12}	228	$56, 13, 2\times6, 2\times4,$ $4\times3, 12\times2, 107\times1$	11.24s	232	200.43s
F_{13}	1446	$2394, 3,$ $8\times2, 37\times1$	OM	2450	OM	F_{16}	4777	8866	OM	8866	OM
F_{14}	1636	$2154, 3,$ $4\times2, 43\times1$	OM	2206	OM	F_{17}	4959	8415	OM	8415	OM
F_{15}	1085	$1800, 8, 4, 6\times3,$ $23\times2, 104\times1$	OM	1980	OM	F_{18}	4869	8712	OM	8712	OM

In this table, $1\times j$ is denoted by j for short. For example, the #block data 248 of Yalmip for F_1 stands for one block of size 248.

with other simplification methods, e.g. [4], to reduce computational costs further. We will apply the `SparseSOS` algorithm to solve large scale unconstrained and constrained polynomial optimization problems in future work.

8. REFERENCES

[1] J. Agler, W. Helton, S. McCullough, L. Rodman, *Positive semidefinite matrices with a given sparsity pattern*, Linear algebra and its applications, 107, 101-149 (1988).

[2] P. R. Amestoy, T. A. Davis, I. S. Duff, *Algorithm 837: AMD, an approximate minimum degree ordering algorithm*, ACM Transactions on Mathematical Software, 30(3), 381-388 (2004).

[3] A. Berry, J. R. S. Blair, P. Heggernes, B. W. Peyton, *Maximum cardinality search for computing minimal triangulations of graphs*, Algorithmica, 39(4), 287-298 (2004).

[4] D. Bertsimas, R. M. Freund, X. A. Sun, *An accelerated first-order method for solving SOS relaxations of unconstrained polynomial optimization problems*, Optim. Methods Softw., 28(3), 424-441 (2013).

[5] J. R. S. Blair, B. Peyton, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computation, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer-Verlag, New York, 1-29 (1993).

[6] J. S. Campos, P. Parpas, *A Multigrid Approach to SDP Relaxations of Sparse Polynomial Optimization Problems*, SIAM Journal on Optimization, 28(1), 1-29 (2016).

[7] L. Dai, B. Xia, *Smaller SDP for SOS decomposition*, Journal of Global Optimization, 63(2), 343-361 (2015).

[8] D. R. Fulkerson, O. A. Gross, *Incidence matrices and interval graphs*, Pacific J. Math., 15, 835-855 (1965).

[9] K. Gatermann, P. A. Parrilo, *Symmetry groups, semidefinite programs, and sums of squares*, Journal of Pure and Applied Algebra, 192(1), 95-128 (2002).

[10] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York (1980).

[11] J. Haglund, K. Ono, D. G. Wagner, *Theorems and conjectures involving rook polynomials with real roots*, In: Proceedings of Topics in Number Theory and Combinatorics, 207-221 (1997).

[12] P. Heggernes, *Minimal triangulations of graphs: a survey*, Discrete Mathematics, 306(3), 297-317 (2006).

[13] D. Henrion, J. Malick, *Projection methods in conic optimization*, in Handbook on Semidefinite, Conic and Polynomial Optimization, Springer, New York, 565-600 (2012).

[14] E. Kaltofen, Z. Yang, L. Zhi, *A proof of themonotone column permanent (mcp) conjecture for dimension 4 via sums-of-squares of rational functions*, In: Proceedings of the 2009 Conference on Symbolic Numeric Computation, 65-70, ACM, New York (2009).

[15] M. Kojima, S. Kim, H. Waki, *Sparsity in sums of squares of polynomials*, Math. Program., 103, 45-62 (2005).

[16] J. Löfberg, YALMIP: a toolbox for modeling and optimization in MATLAB, In 2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508), 284-289.

[17] J. Löfberg, *Pre- and Post-Processing Sum-of-Squares Programs in Practice*, IEEE Transactions on Automatic Control, 54(5), 1007-1011 (2009).

[18] A. Marandi, E. D. Klerk, J. Dahl, *Solving sparse polynomial optimization problems with chordal structure using the sparse bounded-degree sum-of-squares hierarchy*, Discrete Applied Mathematics (2017).

[19] J. Nie, J. Demmel, *Sparse SOS Relaxations for Minimizing Functions that are Summations of Small Polynomials*, SIAM Journal On Optimization, 19(4), 1534-1558 (2008).

[20] P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*, Ph.D. Thesis, California Institute of Technology (2000).

[21] P. A. Parrilo, B. Sturmfels, *Minimizing Polynomial Functions*, Proceedings of the Dimacs Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science, 32(1), 83-100 (2001).

[22] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler and P. A. Parrilo, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, available from `http://www.mit.edu/~parrilo/sostools`, 2013.

[23] F. Permenter, P. A. Parrilo, *Basis selection for SOS programs via facial reduction and polyhedral approximations*, Decision and Control. IEEE, 6615-6620 (2014).

[24] F. Permenter, P. A. Parrilo, *Finding sparse, equivalent SDPs using minimal coordinate projections*, In 54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 7274-7279 (2015).

[25] B. Reznick, *Extremal PSD forms with few terms*, Duke Math. J., 45, 363-374 (1978).

[26] H. Waki, S. Kim, M. Kojima, M. Muramatsu, *Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity*, SIAM Journal on Optimization, 17(1), 218-242 (2006).

[27] H. Waki, S. Kim, M. Kojima, M. Muramatsu and H. Sugimoto, *SparsePOP: a Sparse Semidefinite Programming Relaxation of Polynomial Optimization Problems*, ACM Transactions on Mathematical Software, 35(2), 1-13 (2008).

[28] T. Weisser, J. B. Lasserre, K. C. Toh, *Sparse-BSOS: a bounded degree SOS hierarchy for large scale polynomial optimization with sparsity*, Mathematical Programming Computation, 10(1), 1-32 (2018).

[29] Z. Yang, G. Fantuzzi, A. Papachristodoulou, *Exploiting Sparsity in the Coefficient Matching Conditions in Sum-of-Squares Programming Using ADMM*, IEEE Control Systems Letters, 1(1), 80-85 (2017).

[30] Z. Yang, G. Fantuzzi, A. Papachristodoulou, *Sparse sum-of-squares (SOS) optimization: A bridge between DSOS/SDSOS and SOS optimization for sparse polynomials*, 2018, arXiv:1807.05463.