Solving sparse polynomial optimization problems with chordal structure using the sparse, bounded-degree sum-of-squares hierarchy

Ahmadreza Marandi[a], Etienne de Klerk[a,b], Joachim Dahl[c]

[a] *Tilburg School of Economics and Management, Tilburg University, The Netherlands*
[b] *Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands*
[c] *MOSEK ApS, Copenhagen O, Denmark*

Abstract

The sparse bounded degree sum-of-squares (sparse-BSOS) hierarchy of Weisser, Lasserre and Toh [arXiv:1607.01151,2016] constructs a sequence of lower bounds for a sparse polynomial optimization problem. Under some assumptions, it is proven by the authors that the sequence converges to the optimal value. In this paper, we modify the hierarchy to deal with problems containing equality constraints directly, without eliminating or replacing them by two inequalities. We also evaluate the sparse-BSOS hierarchy on a well-known bilinear programming problem, called the pooling problem.

Keywords: Polynomial optimization, Sparse sum-of-squares hierarchy, Semi-definite programming, Pooling problem, Chordal sparsity structure

2010 MSC: 90C20, 90C22, 90C26, 90C30

1. Introduction

A polynomial optimization problem (POP) is a mathematical optimization problem in which all constraints and the objective function are multi-variate polynomials. POPs include non-convex quadratic programming problems, which were proved to be $NP-$hard by Pardalos and Vavasis [19].

Many approaches are available for constructing lower bounds for the optimal value of a POP (denoted by f^*). Kim, Kojima and Waki [11] proposed a relaxation of a POP using a generalized Lagrangian dual. Lasserre [14] introduced an LP hierarchy that constructs a sequence of lower bounds for f^*. Using the Krivine *positivstellensatz* [12], Lasserre showed that under some assumptions the sequence converges to f^*. In the hope of getting a tighter lower bound, Lasserre, Toh,

Email addresses: a.marandi@uvt.nl (Ahmadreza Marandi), e.deklerk@uvt.nl (Etienne de Klerk), joachim.dahl@mosek.com (Joachim Dahl)

and Yang [15] extended the LP hierarchy to an SDP one, called the bounded degree sum-of-squares (BSOS) hierarchy. The advantage of the BSOS hierarchy is that it contains one semidefinite matrix variable, which has a fixed size that is independent of the level of the hierarchy. A major drawback of the BSOS hierarchy lies in the fact that the number of linear variables grows quickly when the level of the hierarchy increases. In an effort to resolve this issue, Weisser, Lasserre and Toh [23] introduced a modification of the BSOS hierarchy, called the sparse-BSOS hierarchy, for POPs with a particular structural sparsity, which satisfies *running intersection property* (RIP).

The RIP is a well-known concept in graph theory. In the literature of positive semi-definite (PSD) matrices and polynomial optimization, exploiting a sparsity that satisfies RIP is done by studying the corresponding chordal graphs, see [8] for PSD matrices and [22] for polynomial optimization. The results in [23] can be seen as a combination of the results in the papers [15] and [22].

POPs have many real-life applications. Some of these applications were studied in the recent paper by Ahmadi and Majumdar [1]. In this paper, we analyze the behavior of the sparse-BSOS hierarchy on a class of bilinear programming problems, called pooling problems. Solving the pooling problem is attracting considerable interest due to their applications in many real-life optimization problems, like oil refinery planning, chemical process, and water-waste network design. There are many formulations for the pooling problem. Haverly [10] proposed a formulation, called the P-formulation. It was shown by Alfaki and Haugland [3] that the P-formulation problem is NP-hard. One way of finding a lower bound for the P-formulation problem is by using the McCormick relaxation of each bilinear term, which can be reformulated as a Mixed Integer Linear Programing problem. In order to tighten this relaxation, Sahinidis and Tawarmalani [20] proposed the *PQ-formulation*. Dey and Gupte [6] proved that even for the PQ-formulation, using the McCormick relaxations of the bilinear terms might yield a lower bound that is far from the optimal value of the problem. There are other formulations with different characteristics for the pooling problem, like Q-, TP- formulations. Detailed discussion can be found in the surveys by Misener and Floudas [17] and Gupte et al. [9], and the Ph.D. thesis by Alfaki [2].

Recently, semi-definite programming (SDP) hierarchies have been used to find lower bounds for pooling problems. Frimannslund et al. [7] applied the hierarchy proposed by Lasserre [13] to pooling problems. As they pointed out, the fast increase in the sizes of the SDP variables in the problem, which is related to the level of the hierarchy, prevents the hierarchy from being applicable

for pooling problems. Recently, Marandi, Dahl, and De Klerk [16] evaluated the BSOS hierarchy on pooling problems. They found that the BSOS hierarchy is successful in acquiring the optimal values of small-sized instances, but because of the number of variables, the hierarchy does not work well on moderate and large-sized instances. In this paper, we evaluate the sparse-BSOS hierarchy on the P-formulation of the pooling problem and compare the results with BSOS.

The BSOS and sparse-BSOS hierarchies are applicable to POPs that do not contain any equality constraints. However, all pooling problem formulations contain many equality constraints. The standard way of dealing with equality constraints is elimination, or replacing them with two inequalities. A way of eliminating equality constraints in the P-formulation was proposed in [16]; however, the elimination may destroy the sparsity pattern. On the other hand, replacing any equality constraints with two inequalities keeps the sparsity pattern but increases the number of constraints in the problem, which is not desirable in the BSOS and sparse-BSOS hierarchies because it makes each level harder to solve. In this paper, we show how the hierarchies can be modified to deal with equality constraints directly so that the convergence results remain valid.

The remainder of the paper is organized as follows. Section 2 briefly defines the pooling problem and the P-formulation. Section 3 describes the sparse-BSOS hierarchy proposed in [23]. Section 4 demonstrates the link between graph theory and polynomial optimization. In particular, in Section 4.1 we mention some well-known results in graph theory, and in Section 4.2 we construct a graph corresponding to a POP and exploit a sparsity that satisfies the RIP. In Section 5, we show how to modify the BSOS and sparse-BSOS hierarchies to deal with equality constraints directly. A numerical evaluation of the results is provided in Section 6.

2. The P-formulation of the pooling problem

In this section, we describe the P-formulation of the pooling problem. The notation we are using is the same as in [9]. A pooling problem is a generalization of a network flow problem in which the inputs possess different specifications. The goal is to minimize the cost of mixing the inputs such that the demand with restrictions over specifications is satisfied. In the pooling problem there are three types of units: ones that store the inputs, which we call *inputs* and denote by $\mathcal{I}$, ones that mix the incoming flows from *inputs*, which we call *pools* and denote by $\mathcal{L}$, and the units that mix the incoming flows from *pools* and/or *inputs*, and store the outputs, which we call *output* and denote by $\mathcal{J}$. The pooling problem can be determined by an acyclic directed graph

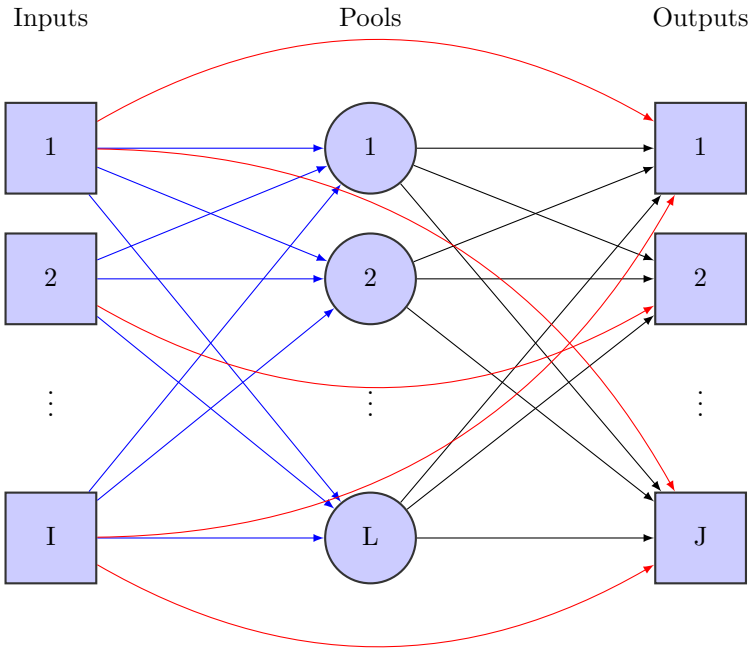


Figure 1: An example of a standard pooling problem with I inputs, L pools and J output.

$G = (\mathcal{N}, \mathcal{A})$, where $N = \mathcal{I} \cup \mathcal{L} \cup \mathcal{J}$, and $\mathcal{A}$ is the set of links between the units, which means $\mathcal{A} \subseteq (\mathcal{I} \times \mathcal{L}) \cup (\mathcal{I} \times \mathcal{J}) \cup (\mathcal{L} \times \mathcal{L}) \cup (\mathcal{L} \times \mathcal{J})$. In this paper, we consider the standard pooling problem, which has no link between the pools and hence $\mathcal{A} \cap \mathcal{L} \times \mathcal{L} = \emptyset$. Figure 1 shows an illustration of a standard pooling problem.

Let c_{ij} be the unit cost of sending one unit of flow from i to j. Also, let y_{ij} be the flow-rate between units i and j, where the link between the two units has the transmission capacity of at most u_{ij}. Also, each unit i may have a capacity C_i. Let the concentration of the kth specification in input i be λ_{ik}, and in the pool l be p_{lk}. If the concentration in the jth output has the lower and upper bounds restrictions of μ_{min} and μ_{max}, respectively, then the P-formulation of the pooling

problem is formulated as follows:

$$\min_{y,p} \quad \sum_{(i,j)\in\mathcal{A}} c_{ij} y_{ij} \tag{1a}$$

s.t.

$$\sum_{\substack{i\in\mathcal{I}:\\(i,l)\in\mathcal{A}}} y_{il} = \sum_{\substack{j\in\mathcal{J}:\\(l,j)\in\mathcal{A}}} y_{lj}, \quad l\in\mathcal{L}, \tag{1b}$$

$$\sum_{\substack{j\in\mathcal{L}\cup\mathcal{J}:\\(i,j)\in\mathcal{A}}} y_{ij} \le C_i, \quad i\in\mathcal{I}, \tag{1c}$$

$$\sum_{\substack{j\in\mathcal{J}:\\(l,j)\in\mathcal{A}}} y_{lj} \le C_l, \quad l\in\mathcal{L}, \tag{1d}$$

$$\sum_{\substack{i\in\mathcal{I}\cup\mathcal{L}:\\(i,j)\in\mathcal{A}}} y_{ij} \le C_j, \quad j\in\mathcal{J}, \tag{1e}$$

$$0 \le y_{ij} \le u_{ij}, \quad (i,j)\in\mathcal{A}, \tag{1f}$$

$$\sum_{\substack{i\in\mathcal{I}:\\(i,l)\in\mathcal{A}}} \lambda_{ik} y_{il} = p_{lk} \sum_{\substack{j\in\mathcal{J}:\\(l,j)\in\mathcal{A}}} y_{lj}, \quad l\in\mathcal{L},\ k\in\mathcal{K}, \tag{1g}$$

$$\sum_{\substack{i\in\mathcal{I}:\\(i,j)\in\mathcal{A}}} \lambda_{ik} y_{ij} + \sum_{\substack{l\in\mathcal{L}:\\(l,j)\in\mathcal{A}}} p_{lk} y_{lj} \le \mu_{jk}^{max} \sum_{\substack{i\in\mathcal{I}\cup\mathcal{L}:\\(i,j)\in\mathcal{A}}} y_{ij}, \quad j\in\mathcal{J},\ k\in\mathcal{K}, \tag{1h}$$

$$\sum_{\substack{i\in\mathcal{I}:\\(i,j)\in\mathcal{A}}} \lambda_{ik} y_{ij} + \sum_{\substack{l\in\mathcal{L}:\\(l,j)\in\mathcal{A}}} p_{lk} y_{lj} \ge \mu_{jk}^{min} \sum_{\substack{i\in\mathcal{I}\cup\mathcal{L}:\\(i,j)\in\mathcal{A}}} y_{ij}, \quad j\in\mathcal{J},\ k\in\mathcal{K}. \tag{1i}$$

Here is a summarized interpretation of the constraints:

(1b): keeping the balance between the total incoming and outgoing flow-rates for each pool,

(1c),(1d),(1e): restricting the units' capacity,

(1f): restricting the capacity of the links between the units,

(1g): keeping the balance of the concentration of each specification in each pool for incoming and outgoing flows,

(1h),(1i): restricting the concentration of each specification in each output by the lower and upper bounds.

In this paper, we use a sparsity pattern and the sparse version of the bounded degree sum-of-squares hierarchy [23] to solve the P-formulation (1).

3. Sparsity pattern in a polynomial problem

In this section, we briefly describe the sparse bounded degree sum-of-squares (sparse-BSOS) hierarchy in polynomial optimization introduced, by Weisser, Lasserre and Toh. [23]. In what follows, for an integer $m \geq 0$,

$$[m] := \begin{cases} \{1, ..., m\} & \text{if } m > 0, \\ \emptyset & \text{if } m = 0. \end{cases}$$

Also, we assume that $x \in \mathbb{R}^n$, and for a given $\mathcal{D} \subseteq [n]$, we denote by $\Sigma[x; \mathcal{D}]_\kappa$, the cone of sum-of-squares polynomials of degree at most κ, and by $\mathbb{R}[x; \mathcal{D}]$ the ring of all polynomials in the variables $\{x_i : i \in \mathcal{D}\}$.

For a general polynomial optimization problem

$$\begin{aligned} f^* = \min_x \quad & f(x) \\ \text{s.t.} \quad & g_j(x) \geq 0, \quad j = 1, ..., m, \end{aligned} \tag{2}$$

where $x \in \mathbb{R}^n$, $n, m \in \mathbb{N}$ and all $f(x)$ and $g_j(x)$, $j = 1, ..., m$ are n-variate polynomials, the *running intersection property* is defined as follows.

Definition 1. *Problem (2) satisfies the running intersection property if there exists $q \in \mathbb{N}$, $\mathcal{D}_\ell \subseteq [n]$ and $\mathcal{C}_\ell \subseteq [m]$ for all $\ell \in [q]$ such that*

- *$f = \sum_{\ell=1}^q f^\ell$, where $f^\ell \in \mathbb{R}[x; \mathcal{D}_\ell]$, for all $\ell \in [q]$,*
- *$g_j \in \mathbb{R}[x; \mathcal{D}_\ell]$, for all $j \in \mathcal{C}_\ell$, and $\ell \in [q]$,*
- *$\bigcup_{\ell=1}^q \mathcal{D}_\ell = [n]$, and $\bigcup_{\ell=1}^q \mathcal{C}_\ell = [m]$,*
- *for all $\ell \in [q-1]$, there is an $s \leq \ell$ such that $\left(\mathcal{D}_{\ell+1} \cap \bigcup_{r=1}^{\ell} \mathcal{D}_\ell\right) \subseteq \mathcal{D}_s$.* □

Assume that the *running intersection property* holds for (2). Let

$$\mathbb{N}^\ell_d := \left\{ (\alpha, \beta) \in \mathbb{N}^{2m} : \quad \alpha_j = \beta_j = 0 \text{ if } j \notin \mathcal{C}_\ell, \ \sum_{j \in [m]} \alpha_j + \beta_j \leq d \right\},$$

and

$$h^\ell_{\alpha\beta} := \prod_{j \in [m]} g_j^{\alpha_j} (1 - g_j)^{\beta_j} \in \mathbb{R}[x, \mathcal{D}_\ell], \ (\alpha, \beta) \in \mathbb{N}^\ell_d.$$

Then, we have the following result from [23].

Theorem 1. *[23, Theorem 2] Consider the general polynomial optimization problem (2). Suppose that it satisfies the running intersection property, and $g_j(x) \leq 1$ for any feasible solution x, $j \in [m]$. Also, assume that for all $\ell \in [q]$, the ring of $\mathbb{R}[x; \mathcal{D}_\ell]$ is generated by $\{1, (g_j)_{j \in \mathcal{C}_\ell}\}$, and there exists $M_\ell > 0$ and $j \in \mathcal{C}_\ell$ such that $g_j = M_\ell - \sum_{i \in \mathcal{D}_\ell} x_i^2$. Then, for a fixed $\kappa \in \mathbb{N}$, $\{q_d^\kappa\}$ is a non-decreasing sequence and $q_d^\kappa \to f^*$ as $d \to +\infty$, where*

$$q_d^\kappa := \sup \left\{ t : \begin{array}{c} f^\ell - \sum_{(\alpha,\beta) \in \mathbb{N}_d^\ell} \lambda_{\alpha\beta}^\ell h_{\alpha\beta}^\ell \in \Sigma[x; \mathcal{D}_\ell]_\kappa, \qquad \ell \in [q] \\ \\ f - t = \sum_{\ell \in [q]} f^\ell, \;\; \lambda^\ell \geq 0, \; t \in \mathbb{R}, \quad f^\ell \in R[x; \mathcal{D}_\ell], \; \ell \in [q] \end{array} \right\}. \tag{3}$$

□

Theorem 1 introduces a non-decreasing sequence that converges to the optimal value of (2) under some assumptions. Instead of (3), we consider the following equivalent problem where the f^ℓ, $\ell = 1, ..., q$, have been eliminated,

$$q_d^\kappa = \sup \left\{ t : \begin{array}{c} f - t = \sum_{\ell \in [q]} \sum_{(\alpha,\beta) \in \mathbb{N}_d^\ell} \lambda_{\alpha\beta}^\ell h_{\alpha\beta}^\ell + \sum_{\ell \in [q]} \sigma_\ell \\ \\ \sigma_\ell \in \Sigma[x; \mathcal{D}_\ell]_\kappa, \;\; \lambda^\ell \geq 0, \; t \in \mathbb{R}, \; \ell \in [q] \end{array} \right\}. \tag{4}$$

The number of scalar variables in each level of (4) is smaller than the one in the same level of the BSOS hierarchy [15, equation (7)]. In the next proposition we show that all constraints in (4) are linearly independent, when the degree of the sums of squares equals to the degree of the whole equality constraint, but before it, we need to emphasize on the following remark.

Remark 1. *Let $x \in \mathbb{R}^n$ and $p(x) = \sum_{\alpha \in \bar{\mathbb{N}}_{2\omega}^{[n]}} p_\alpha x^\alpha$, where*

$$\bar{\mathbb{N}}_\kappa^{\mathcal{D}} := \left\{ \alpha \in \mathbb{N}^n : \; \alpha_i = 0 \text{ if } i \notin \mathcal{D}, \; \sum_{i \in [n]} \alpha_i \leq \kappa \right\},$$

for a set $\mathcal{D} \subseteq [n]$. If $M \in R^{\binom{n+\omega}{\omega} \times \binom{n+\omega}{\omega}}$ is a symmetric matrix variable whose rows (columns) are corresponding to the members of $\bar{\mathbb{N}}_\omega^{[n]}$, then linear constraints

$$p_\alpha = \sum_{\substack{\beta, \gamma \in \bar{\mathbb{N}}_\omega^{[n]} \\ \beta + \gamma = \alpha}} M_{\beta\gamma}, \;\; \forall \alpha \in \bar{\mathbb{N}}_{2\omega}^{[n]}, \tag{5}$$

are linearly independent. This is because all the constraints in (5) involve different variables, i.e., no variable appears in two constraints in (5). To see this, let $\beta, \gamma \in \bar{\mathbb{N}}_\omega^{[n]}$ be fixed. Due to the construction of the constraints in (5), the variable $M_{\beta\gamma}$ appears only in the constraint corresponding to $\alpha = \beta + \gamma$, and no other constraints. □

Proposition 1. *Consider problem (2). Let d be such that*

$$2\kappa = \max\{d \max_{j=1,\dots,m} (deg(g_j)), deg(f), 2\kappa\}.$$

Then, all equality constraints in (4) are linearly independent, if the polynomial equality is modeled by equating the monomials coefficients.

Proof. For each $\ell \in [q]$, set $v^\ell = \left(x^\beta\right)_{\beta \in \bar{\mathbb{N}}^{\mathcal{D}_\ell}_\kappa}$. Also, let $\sigma_\ell = {v^\ell}^T W^\ell v^\ell$, for each $\ell \in [q]$, where $W^\ell \in \mathbb{R}^{\binom{n_\ell+\kappa}{\kappa}\times\binom{n_\ell+\kappa}{\kappa}}$ is a positive semi-definite matrix variable, and $n_\ell = |\mathcal{D}_\ell|$. So, Remark 1 implies that the equality constraints in (4) are linearly independent, if the polynomial equality is modeled by equating the monomials coefficients. □

According to the proof of Proposition 1, if $2\kappa \neq \max\{d \max_{j=1,\dots,m} (deg(g_j)), deg(f), 2\kappa\}$, still the constraints corresponding to the monomials up to degree 2κ are linearly independent. If $deg(f) > \max\{d \max_{j=1,\dots,m} (deg(g_j)), 2\kappa\}$, then $deg(f) = \max\{d \max_{j=1,\dots,m} (deg(g_j)), deg(f), 2\kappa\}$, and clearly the dth iteration of the hierarchy is infeasible, because there is no monomial with degree $deg(f)$ in $\sum_{\ell\in[q]} \sum_{(\alpha,\beta)\in\mathbb{N}^\ell_d} \lambda^\ell_{\alpha\beta} h^\ell_{\alpha\beta} + \sum_{\ell\in[q]} \sigma_\ell$.

The main assumption in Theorem 1 is the existence of a splitting that satisfies the *running intersection property*. So, the question is how to exploit such a sparsity for a polynomial optimization problem. In the next section we answer this question.

4. Polynomial optimization and chordal graphs

In this section, we study the relation between polynomial optimization and graph theory. Specifically, we mention some results on chordal graphs and use them to exploit sparsity for a polynomial optimization problem that satisfies the *running intersection property*.

4.1. Chordal graph and maximal cliques

In this subsection we recall some well-known results on chordal graphs and maximal cliques. The notations we are using in this section is the same as in [5].

Definition 2. *Consider an undirected graph $G = (V, E)$, where V and E denote the sets of vertices and edges, respectively. A chord of a cycle is any edge joining two nonconsecutive vertices of the cycle. A graph G is called chordal, if every cycle of length greater than 3 has a chord.*

Let $adj(v)$ denote the set of adjacent vertices to a vertex v. A vertex ordering ϕ of graph $G = (V, E)$ with n vertices is a bijection $\phi : V \to [n]$, and it can be denoted by indexing the vertex set, such that $V = \{v_1, ..., v_n\}$, and $\phi(v_i) = i$, for $i \in [n]$. Let $v_1, ..., v_n$ be a vertex ordering of G and set

$$\mathcal{L}_i := \{v_i, ..., v_n\}, \quad i \in [n].$$

Definition 3. *A vertex ordering ϕ is a perfect elimination ordering if for any $i \in [n]$ the subgraph of G induced by $\mathcal{L}_i \cap adj(v_i)$ is complete.*

Theorem 2. *(see, e.g., [5, Theorem 2.2]) A graph G is chordal if and only if G has a perfect elimination ordering.*

Definition 4. *Let $G = (V, E)$ be any graph. A clique of G is any subset of V for which the induced graph is complete in G. A maximal clique is a clique that is not properly contained in another clique. We denote by $\mathcal{K}_G$ the set of all maximal cliques of G.*

Theorem 3. *For a chordal graph G, assume that $|\mathcal{K}_G| = q$. Then, there is a labeling $\mathcal{K}_G = \{\mathcal{K}_1, ..., \mathcal{K}_q\}$ such that*

$$\forall \ell \in [q-1],\ \exists s \leq \ell :\ \left(\mathcal{K}_{\ell+1} \cap \bigcup_{r=1}^{\ell} \mathcal{K}_\ell \right) \subseteq \mathcal{K}_s.$$

Proof. By invoking [5, Theorem 3.1] and [5, Corollary 1], one can deduce the theorem. □

For any vertex ordering ϕ of graph $G = (V, E)$, let us add extra edges to G in order to make all $\mathcal{L}_i \cap adj(v_i)$ complete, $i \in [n]$, and denote by E^*_ϕ the union of E with the extra edges. Then clearly $G^* = (V, E^*_\phi)$ is chordal, because ϕ is a *perfect elimination ordering* for G^*.

It is well-known that the Laplacian matrix of a graph $G = (V, E)$, denoted by L, is positive semi-definite. We denote the permuted Laplacian matrix of a graph G according to a vertex ordering ϕ by L_ϕ. As it is proved in [21, Section 9.1], using the Cholesky factorization $L_\phi = R^T R$, the nonzero components of $R + R^T$ correspond to the edges in E^*_ϕ.

A vertex ordering that minimizes $|E^*_\phi \backslash E|$ over all possible vertex orderings of G is called a minimum ordering. Finding a minimum ordering is known to be NP-complete [21, Section 6.6]. There are many polynomial-time algorithms to find a "good" ordering, see [21, Section 6.6]. *Minimum degree* ordering is such an algorithm, which finds the vertex v with the least degree, set $\phi(v) = i$ and delete v from the graph G in the ith iteration. Finding the vertex with the least degree in each

iteration is time-consuming. Therefore, in our numerical results we use the *approximate minimum degree* ordering (AMD) introduced in [4], which finds upper bounds of the vertices' degree in each iteration and use them to select the vertex v.

4.2. Exploiting sparsity in a polynomial optimization using chordal graphs

In this subsection, we construct a graph corresponding to problem (2) and use the results mentioned in Section 4.1 to exploit sparsity that satisfies the *running intersection property*. The graph is essentially the same as the one constructed in [22].

Consider a general polynomial problem (2). A graph $G = (V, E)$ associated to this problem can be constructed as follows:

- the vertex set $V := \{x_1, ..., x_n\}$,
- $E_j := \{(x_i, x_k) : \text{variables } x_i, x_k \text{ are present in the definition of } g_j(x)\}$, $j \in [m]$,
- $E_0 := \{(x_i, x_k) : \text{product } x_i x_k \text{ is present in the definition of } f(x)\}$,
- the edge set $E := \bigcup_{j=0}^{m} E_j$.

Let L be the Laplacian matrix of $G = (V, E)$. Using the results in Section 4.1, in order to find the sparsity pattern for the polynomial optimization that satisfies the *running intersection property*, one can use Algorithm 1.

Algorithm 1 Exploit sparsity in the polynomial optimization (2) using chordal graphs

$\phi \leftarrow$ get the ordering from AMD algorithm on graph G
$L_\phi \leftarrow$ reorder Laplacian matrix L according to ϕ
use Cholesky factorization of L_ϕ to construct a chordal extension G^* of G
$\{\mathcal{D}_1, ..., \mathcal{D}_q\} \leftarrow$ the maximal cliques of G^*
for all $l \in [q]$ **do**
  $\mathcal{C}_l \leftarrow$ all g_j in $\mathbb{R}[x, \mathcal{D}_l]$
end for

Theorem 4. *There is a bijection $\Gamma : [q] \to [q]$, such that the index blocks $\mathcal{D}_{\Gamma(\ell)}$ and constraint blocks $\mathcal{C}_{\Gamma(\ell)}$, $\ell \in [q]$, constructed by Algorithm 1 satisfy the running intersection property.*

Proof. By Theorem 3, there is a bijection $\Gamma : [q] \to [q]$ such that

$$\forall \Gamma(\ell) \in [q-1],\ \exists \Gamma(s) \leq \Gamma(\ell) :\ \left(\mathcal{D}_{\Gamma(\ell)+1} \cap \bigcup_{r=1}^{\ell} \mathcal{D}_{\Gamma(\ell)} \right) \subseteq \mathcal{D}_{\Gamma(s)}.$$

Now, we show that $f = \sum_{\ell=1}^{q} f^\ell$, where $f^\ell \in \mathbb{R}[x; \mathcal{D}_{\Gamma(\ell)}]$, for all $\ell \in [q]$. Because f is a polynomial, it is sufficient to show that for each monomial in the definition of f there is an ℓ such that the monomial belongs to $\mathbb{R}[x; \mathcal{D}_\ell]$. Let x^β be a monomial in the definition of f, where $\beta \in \mathbb{N}^n$. Due to the structure of E_0, the graph induced by the vertices corresponding to x^β is complete and hence contained in one of the maximal cliques of G^*. Therefore, there is an $\ell \in [q]$ such that $x^\beta \in \mathbb{R}[x; \mathcal{D}_\ell]$.

By the construction of $\mathcal{C}_l$, it is clear that $g_j \in \mathbb{R}[x, \mathcal{D}_\ell]$ for all $j \in \mathcal{C}_l$, $l \in [q]$, and $\bigcup_{\ell=1}^{q} \mathcal{C}_\ell = [m]$. Also, $\bigcup_{\ell=1}^{q} \mathcal{D}_\ell = [n]$, because $\{\mathcal{D}_1, ..., \mathcal{D}_q\}$ is the set of all maximal cliques of G^*. □

In Theorem 4, we showed that there is an ordering of $[q]$ with which the blocks $\mathcal{D}_\ell$ and $\mathcal{C}_\ell$, $\ell \in [q]$ satisfy the *running intersection property*. In the following lemma, we show that we do not need to know the ordering to solve problem (4).

Lemma 1. *Let $\Gamma : [q] \to [q]$ be a bijection for some $q \in \mathbb{N}$, $\mathcal{D}_\ell \subseteq [n]$ and $\mathcal{C}_\ell \subseteq [m]$ for all $\ell \in [q]$. Then for each $\kappa, d \in \mathbb{N}$, q_d^κ in (4) is given by*

$$q_d^\kappa = \sup \left\{ t : \begin{array}{c} f - t = \sum_{\ell \in [q]} \sum_{(\alpha,\beta) \in \mathbb{N}_d^{\Gamma(\ell)}} \lambda_{\alpha\beta}^{\Gamma(\ell)} h_{\alpha\beta}^{\Gamma(\ell)} + \sum_{\ell \in [q]} \sigma_{\Gamma(\ell)} \\ \sigma_\ell \in \Sigma[x; \mathcal{D}_\ell]_\kappa,\ \ \lambda^\ell \geq 0,\ t \in \mathbb{R},\ \ell \in [q] \end{array} \right\}. \tag{6}$$

Proof. The summations in (6) are over $\ell \in [q]$, and may change the order of summations. In other words:

$$\sum_{\ell \in [q]} \sum_{(\alpha,\beta) \in \mathbb{N}_d^{\Gamma(\ell)}} \lambda_{\alpha\beta}^{\Gamma(\ell)} h_{\alpha\beta}^{\Gamma(\ell)} = \sum_{\ell \in [q]} \sum_{(\alpha,\beta) \in \mathbb{N}_d^{\ell}} \lambda_{\alpha\beta}^{\ell} h_{\alpha\beta}^{\ell},$$

$$\sum_{\ell \in [q]} \sigma_{\Gamma(\ell)} = \sum_{\ell \in [q]} \sigma_\ell.$$

□

Theorems 1, 4 and Lemma 1 show that if:

- $g_j(x) \leq 1$ for any feasible solution x, $j \in [m]$,
- for all $\ell \in [q]$, the ring of $\mathbb{R}[x; \mathcal{D}_\ell]$ is generated by $\{1, (g_j)_{j \in \mathcal{C}_\ell}\}$,
- there exists $M_\ell > 0$ and $j \in \mathcal{C}_\ell$ such that $g_j = M_\ell - \sum_{i \in \mathcal{D}_\ell} x_i^2$,

then for a fixed $\kappa \in \mathbb{N}$, $\{q_d^\kappa\}$ is a non-decreasing sequence that converges to the optimal value of (2), when $\mathcal{D}_\ell$ and $\mathcal{C}_\ell$ are the outputs of Algorithm 1.

The result of this section can be applied to the P-formulation (1) by elimination of equality constraints proposed in [16]. The following example shows how Algorithm 1 works for "Haverly1" and "Adhya1", two pooling problem instances, after elimination of the equality constraints.

Example 1. *"Haverly1" is a pooling problem instance with 3 inputs, 2 outputs and 1 pool, where the inputs are characterized with only 1 specification. The formulation of this instance after elimination of the equality constraints, proposed in [16, Section 3.2], is*

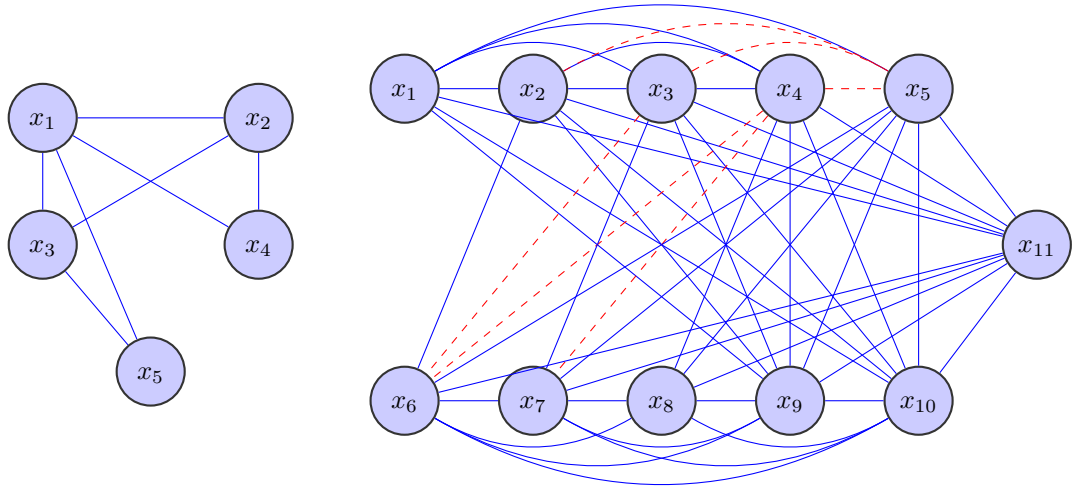
$$
\begin{aligned}
\min \quad & -200x_2(15x_1 - 12) - 200x_3(15x_1 - 6) + 200x_4 - 1000x_5 \\
s.t. \quad & 1 \geq -\frac{3}{4}(x_1 - 1)(x_2 + x_3) \geq 0 && \text{(7a)} \\
& 1 \geq \frac{1}{4}(3x_1 - 1)(x_2 + x_3) \geq 0 && \text{(7b)} \\
& 1 \geq 1 - 2(x_2 + x_4) \geq 0 && \text{(7c)} \\
& 1 \geq 1 - (x_3 + x_5) \geq 0 && \text{(7d)} \\
& 1 \geq \frac{1}{2}(x_4 + x_2) - \frac{2}{5}x_4 - \frac{3}{5}x_1x_2 \geq 0 && \text{(7e)} \\
& 1 \geq \frac{1}{2}(x_5 + x_3) - \frac{2}{3}x_5 - x_1x_3 \geq 0 && \text{(7f)} \\
& 1 \geq x_i \geq 0, \quad i = 1, ..., 5. && \text{(7g)}
\end{aligned}
$$

For this problem, the Laplacian matrix corresponding to its graph G (Figure 2a) is

$$
L = \begin{bmatrix}
4 & -1 & -1 & -1 & -1 \\
-1 & 3 & -1 & -1 & 0 \\
-1 & -1 & 3 & 0 & -1 \\
-1 & -1 & 0 & 2 & 0 \\
-1 & 0 & -1 & 0 & 2
\end{bmatrix}.
$$

The output of the AMD algorithm is $\phi(\{x_1, x_2, x_3, x_4, x_5\}) = [5, 3, 1, 4, 2]$. Using the Cholesky factorization, one finds out that there is no need to add any extra edge, so G is chordal with the maximal cliques $\mathcal{D}_1 = \{x_1, x_2, x_3\}$, $\mathcal{D}_2 = \{x_1, x_2, x_4\}$, and $\mathcal{D}_3 = \{x_1, x_3, x_5\}$. Hence,

$$
\begin{aligned}
\mathcal{C}_1 &= \{(7a), (7b), (7g)_1, (7g)_2, (7g)_3\}, \\
\mathcal{C}_2 &= \{(7c), (7e), (7g)_1, (7g)_2, (7g)_4\}, \\
\mathcal{C}_3 &= \{(7d), (7f), (7g)_1, (7g)_3, (7g)_5\},
\end{aligned}
$$



(a) Haverly1 (b) Adhya1

Figure 2: The graphs corresponding to "Haverly1" and "Adhya1" after elimination of the equality constraints. The red dashed arcs are added to make the graph chordal.

where $(7g)_i$ *is the constraint* $(7g)$ *for* x_i, $i = 1, ..., 5$.

"Adhya1" is a pooling problem instance that has 5 inputs, 4 outputs, 2 pools where the inputs are characterized with 4 specifications. After elimination of the equality constraints, the problem contains 11 variables and 41 constraints. The graph in Figure 2b shows G *where the red dashed arcs are corresponding to the components of* $R + R^T$ *that are zeros in* L_ϕ*. This means that the red dashed arcs are added to make the graph chordal. For* G^**, the maximal cliques are*

$$\begin{aligned}
\mathcal{D}_1 &= \{x_1, x_2, x_3, x_4, x_5, x_9, x_{10}, x_{11}\},\\
\mathcal{D}_2 &= \{x_2, x_3, x_4, x_5, x_6, x_9, x_{10}, x_{11}\},\\
\mathcal{D}_3 &= \{x_3, x_4, x_5, x_6, x_7, x_9, x_{10}, x_{11}\}.
\end{aligned}$$

□

Elimination of the equality constraints may destroy the sparsity pattern. So, in the following section we study algebraic sets with equality constraints and proof a *positivstellensatz* that deals with equality constraints.

5. Problems with equality constraints

Consider the following algebraic set:

$$\mathcal{F} := \{x \in \mathbb{R}^n : \ \ e_t(x) = 0,\ t \in [T],\ \ g_i(x) \geq 0,\ i \in [m]\}\,, \tag{8}$$

where $e_t, g_i \in \mathbb{R}[x, [n]]$, $t \in [T]$, $i \in [m]$. In the next theorem we show that one can slightly change the Krivine's *positivstellensatz* [12] in order to handle algebraic set $\mathcal{F}$.

Theorem 5. *Assume that $e_t(x) \leq 1$ and $g_i(x) \leq 1$, for all $x \in \mathcal{F}$, $i \in [m]$, and $t \in [T]$. If a polynomial $f(x)$ is positive on $\mathcal{F}$ and the ring of polynomials $\mathbb{R}[x, [n]]$ is generated by $\{1, e_t(x), g_i(x)\}_{\substack{i\in[m]\\ t\in[T]}}$, then there is an integer d such that*

$$f(x) = \sum_{(\gamma,\alpha,\theta,\beta)\in\mathbb{N}_d^{2m+2T}} \lambda_{\gamma,\alpha,\theta,\beta} \prod_{t\in[T]} e_t(x)^{\gamma_t} \left(1 - e_t(x)\right)^{\theta_t} \prod_{i\in[m]} g_i(x)^{\alpha_i} \left(1 - g_i(x)\right)^{\beta_i},$$

for some λ that

$$\begin{cases} \lambda_{\gamma,\alpha,\theta,\beta} \geq 0 & \text{if } \gamma = 0 \text{ or } T = 0, \\ \lambda_{\gamma,\alpha,\theta,\beta} \in \mathbb{R} & \text{otherwise.} \end{cases}$$

Proof. We prove this theorem by induction on T, the number of equality constraints. If $T = 0$, there is no equality constraints in $\mathcal{F}$. So, the result follows directly from Krivine's *positivstellensatz* [12]. Now, assume that the result holds for all sets in the form of (8) with T equality constraints and we prove it for a set $\mathcal{F}$ with $T + 1$ equality constraints.

Setting $g(x) := e_{T+1}(x)$, we can write $\mathcal{F}$ as follows:

$$\{x : \ \ g(x) \geq 0,\ \ -g(x) \geq 0,\ \ g_i(x) \geq 0,\ i = 1, ..., m,\ \ e_t(x) = 0,\ t = 1, ..., T\}\,.$$

So, by the induction hypothesis, there is an integer d such that

$$f(x) = \sum_{\substack{\bar{\alpha} = (\alpha_0,\, \alpha_{00},\, \alpha,\, \gamma)\\ \bar{\beta} = (\beta_0,\, \beta_{00},\, \beta,\, \theta)\\ \lambda_{\bar{\alpha},\bar{\beta}} \in \mathbb{N}_d^{2m+2T+4}}} \lambda_{\bar{\alpha},\bar{\beta}}\, g(x)^{\alpha_0} \left(-g(x)\right)^{\alpha_{00}} \left(1 + g(x)\right)^{\beta_{00}} h_{\beta_0\alpha\beta\gamma\theta}(x),$$

where,

$$h_{\beta_0\alpha\beta\gamma\theta}(x) = \left(1 - g(x)\right)^{\beta_0} \prod_{i=1}^{m} g_i(x)^{\alpha_i} \left(1 - g_i(x)\right)^{\beta_i} \prod_{t=1}^{T} e_t(x)^{\gamma_t} \left(1 - e_t(x)\right)^{\theta_t},$$

and

$$\begin{cases} \lambda_{\bar{\alpha},\bar{\beta}} \geq 0 & \gamma = 0 \text{ or } T = 0, \\ \lambda_{\bar{\alpha},\bar{\beta}} \in \mathbb{R} & \text{otherwise.} \end{cases}$$

So, $f(x)$ can be written as

$$\sum_{\substack{\bar{\alpha} = (\alpha_0,\, \alpha_{00},\, \alpha,\, \gamma) \\ \bar{\beta} = (\beta_0,\, \beta_{00},\, \beta,\, \theta) \\ \lambda_{\bar{\alpha},\bar{\beta}} \in \mathbb{N}_d^{2m+2T+4}}} (-1)^{\alpha_{00}} \lambda_{\bar{\alpha},\bar{\beta}}\, g(x)^{\alpha_0+\alpha_{00}}\, (1+g(x))^{\beta_{00}}\, h_{\beta_0\alpha\beta\gamma\theta}(x).$$

By using binomial theorem for $(1+g(x))^{\beta_{00}}$, we have

$$f(x) = \sum_{\substack{\bar{\alpha} = (\alpha_0,\, \alpha_{00},\, \alpha,\, \gamma) \\ \bar{\beta} = (\beta_0,\, \beta_{00},\, \beta,\, \theta) \\ \lambda_{\bar{\alpha},\bar{\beta}} \in \mathbb{N}_d^{2m+2T+4}}} (-1)^{\alpha_{00}} \lambda_{\bar{\alpha},\bar{\beta}}\, g(x)^{\alpha_0+\alpha_{00}} \left(\sum_{j=0}^{\beta_{00}} a_j g(x)^j \right) h_{\beta_0\alpha\beta\gamma\theta}(x),$$

where a_j, $j = 0, ..., \beta_{00}$, are the binomial coefficients and therefore positive. This means that

$$f(x) = \sum_{\substack{\bar{\alpha} = (\alpha_0,\, \alpha_{00},\, \alpha,\, \gamma) \\ \bar{\beta} = (\beta_0,\, \beta_{00},\, \beta,\, \theta) \\ \lambda_{\bar{\alpha},\bar{\beta}} \in \mathbb{N}_d^{2m+2T+4}}} \sum_{j=0}^{\beta_{00}} (-1)^{\alpha_{00}} a_j \lambda_{\bar{\alpha},\bar{\beta}}\, g(x)^{\alpha_0+\alpha_{00}+j} h_{\beta_0\alpha\beta\gamma\theta}(x).$$

Let us fix $\beta_0, \alpha, \beta, \gamma, \theta$ such that $k := \beta_0 + |\alpha| + |\beta| + |\gamma| + |\theta| \leq d$, and set

$$\chi_{\beta_0,\alpha,\beta,\gamma,\theta}(x) := \sum_{(\alpha_0,\alpha_{00},\beta_{00}) \in \mathbb{N}_{d-k}^3} \sum_{j=0}^{\beta_{00}} (-1)^{\alpha_{00}} a_j \lambda_{\bar{\alpha},\bar{\beta}}\, g(x)^{\alpha_0+\alpha_{00}+j}.$$

220 The coefficient of $g(x)^l$, $l = 0, ..., d-k$, in $\chi_{\beta_0,\alpha,\beta,\gamma,\theta}(x)$ is the summation of some $a_j \lambda_{\bar{\alpha},\bar{\beta}}$ and $-a_j \lambda_{\bar{\alpha},\bar{\beta}}$ corresponding to different $\bar{\beta}$ and j. If $l = 0$ and $\gamma = 0$, or $l = 0$ and $T = 0$, then, $\alpha_0 = \alpha_{00} = j = 0$, which means the coefficient of $g(x)^0$ is nonnegative. Hence,

$$\begin{aligned} f(x) &= \sum_{\substack{(\beta_0,\alpha,\beta,\gamma,\theta) \\ k \leq d}} \chi_{\beta_0,\alpha,\beta,\gamma,\theta}(x) h_{\beta_0,\alpha,\beta,\gamma,\theta}(x) \\ &= \sum_{\substack{(\beta_0,\alpha,\beta,\gamma,\theta) \\ k \leq d}} \sum_{l=0}^{d-k} \bar{\lambda}_{l,\beta_0,\alpha,\beta,\gamma,\theta}\, g(x)^l h_{\beta_0,\alpha,\beta,\gamma,\theta}(x), \end{aligned}$$

for some $\bar{\lambda}$ with real components such that $\bar{\lambda}_{l,\beta_0,\alpha,\beta,\gamma,\theta}$ is nonnegative if $l = 0$ and $\gamma = 0$ or $l = 0$ and $T = 0$. So, combining the two summations completes the proof. □

Theorem 5 asserts that the coefficients corresponding to the polynomial-multiplications

$$\prod_{t\in[T]} e_t(x)^{\gamma_t}\left(1-e_t(x)\right)^{\theta_t}\prod_{i\in[m]} g_i(x)^{\alpha_i}\left(1-g_i(x)\right)^{\beta_i},$$

with $\gamma \neq 0$, are unrestricted.

Remark 2. *Applying Theorem 5 to [23, Theorem 1], if a polynomial optimization problem with feasible region (8) satisfies the assumptions of Theorem 1, then the part of linear variable λ^l in (3) associated with the polynomial-multiplications containing equality constraints are unrestricted, and all of the convergence results in [23] are valid.*

Remark 3. *Considering Theorem 5 and Remark 2, one can easily construct the corresponding graph to any polynomial optimization problem with some equality constraints, and exploit the sparsity that satisfies the running intersection property, as described in Section 4.*

For a pooling problem, let denote by G the graph of the P-formulation (1) constructed with the procedure in Section 4.2. All nodes in G are corresponding to a variable in the P-formulation (1). Because of the constraint (1c), nodes corresponding to y_{il}, y_{ij}, $(i,l),(i,j) \in \mathcal{A}$ are connected in G, for each $i \in \mathcal{I}$. We denote by $\mathcal{K}_i$, $i \in \mathcal{I}$, this type of cliques. The nodes corresponding to y_{ij}, y_{lj}, $(i,j),(l,j) \in \mathcal{A}$, for each $j \in \mathcal{J}$ are connected because of (1e), and we denote the cliques by $\mathcal{K}_j$, $j \in \mathcal{J}$. In the same way because of (1d), the nodes y_{il}, y_{lj}, $(i,l),(l,j) \in \mathcal{A}$, makes the cliques $\mathcal{K}_l$ for each $l \in \mathcal{L}$. If there is an arc in $\mathcal{A}$ between two units, then their corresponding cliques have a node in common. This means that the overlaps between the cliques in G is related to the arcs in $\mathcal{A}$. Let denote by $\bar{G}$ the network (Figure 1) of the pooling problem. The latter discussion shows that the more sparse is $\bar{G}$, the fewer overlaps are between the cliques in G. This means that if $\bar{G}$ is sparse then the possibility that in the sparse-BSOS hierarchy the matrix variables have fewer overlaps are high and therefore in this case each level of the sparse-BSOS hierarchy can be solved faster than the same level of the BSOS one.

6. Numerical result

In this section, we present the numerical evaluation of the sparse-BSOS hierarchy on the pooling problem instances and compare it with the BSOS hierarchy [15]. In the implementation of the BSOS hierarchy, we model the polynomial equality by equating the monomials coefficients. The detailed information of the instances is presented in Table 1.

	optimal value	I	J	L	K	# var.	# const.
Haverly1	-400.00	3	2	1	1	7	13
Haverly2	-600.00	3	2	1	1	7	13
Haverly3	-750.00	3	2	1	1	7	13
Ben-Tal4	-450.00	4	2	1	1	8	15
Ben-Tal5	-3,500.00	5	5	3	2	38	63
DeyGupte4	-1.00	2	4	2	2	16	58
Foulds2	-1,100.00	6	4	2	1	22	42
Foulds3	-8.00	11	16	8	1	168	235
Foulds4	-8.00	11	16	8	1	168	235
Adhya1	-549.80	5	4	2	4	21	51
Adhya2	-549.80	5	4	2	6	25	67
Adhya3	-561.05	8	4	3	6	38	87
Adhya4	-877.6.	8	5	2	4	26	61
RT2	-4,391.83	3	3	2	8	32	85
sppA0	Unknown *	20	15	10	24	411	1,066

Table 1: Detailed information of some pooling problem instances. # var. and # const. present the number of variables and constraints in the P-formulation (1), respectively.

* For this instance the exact optimal value is unknown, but the interval on which it lies is known to be $[-36233.40, -35812.33]$.

The results in Sections 4 and 5 have been implemented in a Julia 0.5 package called "*Polyopt*", available on `https://github.com/MOSEK/Polyopt.jl`. The splitting can be specified by the user or by using the function "*Polyopt.chordal_embedding()*", which is the implementation of the results in Section 4.2.

The GAMS files of the instances except DeyGupte4 can be found in the website `http://www.ii.uib.no/~mohammeda/spooling/`. The instance DeyGupte4 is constructed in [16], and the details can be found in that paper. All computations in this paper were carried out with Julia 0.5 on an Intel i7-4790 3.60GHz Windows computer with 16GB of RAM.

In our numerical evaluations, due to the suggestion in [23], we consider $\kappa = 2$ in (4) for small and moderate-sized instances (Haverly1-3, Ben-Tal4, DeyGupte4, Foulds2 and Adhya1). However, for the large-sized instances (Ben-Tal 5, Foulds 3,4, Adhya2-4, RT2, sppA0) we consider $\kappa = 1$. Also, to construct problems that satisfies the assumptions of Theorem 1, we add the constraints $M_\ell - \sum_{i\in\mathcal{D}_\ell} x_i^2 \geq 0$, $\ell \in [q]$, to the problem.

Tables 2 and 3 compare the results of solving different pooling problem instances with the sparse-BSOS hierarchy (4) and BSOS hierarchy [15]. In each cell of the tables the following information is presented: $\boxed{\begin{smallmatrix} q_d^\kappa() \\ [\text{time}] \end{smallmatrix}}$, where the number of blocks is presented between parentheses, and the time contains the time for constructing the level of the hierarchy and solving it by Mosek 8 [18]. The comparison has been made in two ways: the columns that are denoted by "with elimination" contain the result of applying the corresponding hierarchy to the pooling problem instances using the elimination method proposed in [16]. In the other columns, we use Theorem 5 to handle the equality constraints directly. For the instances Foulds 3-4 and sppA0, the time that is mentioned in Table 3 is larger for the sparse-BSOS hierarchy than the one for the BSOS hierarchy. This is due to the overlap of the matrix variables in the sparse-BSOS hierarchy. The dash "-" in Table 3 means we cannot solve the corresponding level of the hierarchy, due to the size of the problem.

After elimination of equality constraints, the constraints and variables are reduced. This means that in this case, applying Theorem 5 is not worthwhile with respect to the time, because the solver needs to solve a larger problem. Comparing the columns in Tables 2 and 3 shows that using Theorem 5 does not necessarily result in better or worse lower bounds. For Adhya4, applying this theorem results in better lower bounds both in sparse-BSOS and BSOS hierarchies, but this is not the case for Adhya1.

As it can be seen in Tables 2 and 3, the sparse-BSOS hierarchy may get worse lower bounds

	iteration	BSOS with elimination	BSOS	SBSOS with elimination	SBSOS
Haverly1	d=1	−600.00 [0.01]	−600.00 [0.04]	−600.00(3) [0.01]	−600.00(3) [0.02]
	d=2	−417.20 [0.02]	−417.20 [0.05]	−481.66(3) [0.01]	−479.88 [0.03]
	d=3	**-400.00** [0.08]	**-400.00** [0.14]	**-400.00**(3) [0.03]	**-400.00**(3) [0.05]
Haverly2	d=1	−1200.00 [0.01]	−1200.00 [0.04]	−1200.00(3) [0.01]	−1200.00(3) [0.01]
	d=2	**-600.00** [0.02]	**-600.00** [0.08]	−773.53(3) [0.02]	−766.79(3) [0.02]
	d=3	**-600.00** [0.13]	**-600.00** [0.21]	−607.45(3) [0.05]	−603.24(3) [0.07]
	d=4	**-600.00** [2.46]	**-600.00** [3.53]	**-600.00**(3) [0.28]	**-600.00**(3) [0.38]
Haverly3	d=1	−875.00 [0.01]	−875.00 [0.05]	−875.00(3) [0.01]	−875.00(3) [0.03]
	d=2	**-750.00** [0.02]	**-750.00** [0.07]	−785.84(3) [0.02]	−785.84(3) [0.03]
	d=3	**-750.00** [0.07]	**-750.00** [0.20]	**-750.00**(3) [0.04]	**-750.00**(3) [0.05]
Ben-Tal4	d=1	−650.00 [0.02]	−650.00 [0.08]	−650.00(3) [0.01]	−650.00(3) [0.03]
	d=2	−467.20 [0.04]	−467.20 [0.12]	−535.84(3) [0.02]	−530.10(3) [0.04]
	d=3	**-450.00** [0.16]	**-450.00** [0.28]	**-450.00** (3) [0.05]	**-450.00** (3) [0.09]
DeyGupte4	d=1	−4.00 [0.32]	−1.33 [13.62]	−4.00(1) [0.32]	−1.33(6) [3.01]
	d=2	−3.33 [0.98]	−1.33 [26.12]	−3.33(1) [0.99]	−1.33(6) [5.82]
	d=3	≈**-0.98** [33.53]	**-1.00** [80.09]	≈**-0.98**(1) [27.89]	≈−1.24(6) [19.36]
Foulds2	d=1	−1,200.00 [25.04]	−1,200.00 [147.16]	−1,200.00(6) [2.24]	−1,200.00(8) [5.97]
	d=2	−1,191.28 [53.07]	−1,182.79 [380.90]	−1,193.86(6) [4.83]	−1,182.80(8) [6.05]
	d=3	−1,102.56 [126.04]	≈−1,101.74 [863.37]	−1,103.25(6) [12.14]	−1,103.52(8) [10.78]
Adhya1	d=1	−999.32 [0.59]	−999.32 [127.02]	−999.32(4) [0.26]	−999.32(4) [10.77]
	d=2	−716.84 [1.38]	−981.54 [326.31]	−868.05(4) [0.40]	≈−997.99(4) [24.84]
	d=3	−574.91 [23.10]	≈−613.32 [864.60]	−663.68(4) [4.11]	≈−836.23(4) [54.81]

Table 2: Comparing the BSOS and sparse-BSOS hierarchies with $\kappa = 2$. The columns denoting by "with elimination" apply the hierarchy after elimination of equality constraints. The number of blocks (q) in each level is presented between parentheses. The time of model construction and the solution time is presented between the brackets. Bold-faced entries indicate the (approximate) optimal value.

	iteration	BSOS with elimination	BSOS	SBSOS with elimination	SBSOS
Ben-Tal5	d=1	**-3500.00** [0.06]	**-3500.00** [0.16]	**-3500.00**(9) [0.04]	**-3500.00**(13) [0.07]
Foulds3	d=1	**-8.00** [58.53]	**-8** [115.90]	**-8**(20) [90.20]	**-8**(33) [93.32]
Foulds4	d=1	**-8.00** [57.67]	**-8** [130.71]	**-8**(22) [162.53]	**-8**(33) [92.99]
Adhya2	d=1	−798.29 [0.01]	−854.10 [0.05]	−798.29(4) [0.05]	−854.10(4) [0.09]
	d=2	−577.00 [0.20]	−853.74 [0.81]	−686.60(4) [0.11]	−854.10(4) [0.37]
	d=3	−566.27 [32.10]	≈−575.67 [988.90]	−573.(4) [6.98]	−749.43(4) [84.97]
Adhya3	d=1	−882.84 [0.02]	−882.84 [0.18]	−882.84(5) [0.02]	−882.84(7) [0.11]
	d=2	−805.07 [0.58]	−882.72 [8.65]	−870.54(5) [0.27]	−882.84(7) [3.07]
Adhya4	d=1	−1055.00 [0.02]	−1003.33 [0.04]	−1055.00(5) [0.02]	−1003.33(7) [0.04]
	d=2	−1,040.00 [0.30]	−1003.33 [0.81]	−1,040.00(5) [0.18]	−1003.33(7) [0.35]
	d=3	−907.96 [253.41]	≈−891.42 [1,140.48]	−1,012.06(5) [17.01]	−982.42(7) [39.29]
RT2	d=1	−45,420.50 [0.02]	−45,420.50 [0.10]	−45,420.50 [0.02]	−45,420.50(13) [0.09]
	d=2	≈−39,287.30 [0.50]	−44,953.90 [2.90]	≈−39,291.60(2) [0.52]	−44,953.90(13) [1.46]
sppA0	d=1	−47,674.90 [161.30]	–	−47,675.00(27) [389.56]	–

Table 3: Comparing the BSOS hierarchy and sparse-BSOS hierarchies with $\kappa = 1$ for large instances. The columns denoting by "with elimination" apply the hierarchy after elimination of equality constraints. The number of blocks (q) in each level is presented between parentheses. The time of the model construction and the solution time is presented between the brackets. Bold-faced entries indicate the (approximate) optimal value.

compared to the BSOS hierarchy, which can be seen on the second level of the sparse-BSOS hierarchy of Haverly1-3, Ben-Tal4, Adhya1-2. The advantage of the sparse-BSOS hierarchy is that each level of the hierarchy can be solved relatively faster than the BSOS one, if the problem is sparse. For Foulds2, the lower bounds from the BSOS and sparse-BSOS hierarchies are close but the time in the sparse-BSOS hierarchy is much less than in the BSOS one.

The intuition behind the sparse-BSOS hierarchy is to split the variables into some blocks that contain only some (and not all) variables. If the blocks have many variables in common, then it does not matter much if we merge them together. In Tables 4 we present the results of solving the pooling problem instances when two blocks are merged if the number of variables in their intersection is greater than 75% of the size of the smallest one. For Haverly1-3, and Ben-Tal4, there is no block merging. For Ben-Tal5, DeyGupte4, Foulds3-4, Adhya2-3, RT2 and sppA0, all of the blocks get merged, which result in the BSOS hierarchy. Hence, we present the results for the rest of the instances in Table 4. To compare the results with Tables 2 and 3, we use $\kappa = 1$ for Adhya4 and $\kappa = 2$ for the rest.

According to the discussion in Section 5, the overlaps in the matrix variables of the sparse-BSOS hierarchy corresponding to the P-formulation (1) is related to the sparsity of the network of the pooling problem, Figure 1. As one can see, for the three instances in Table 4, the networks of the instances are highly sparse, and therefore the possibility that each level of the sparse-BSOS hierarchy can be solved faster than the same level in the BSOS hierarchy is high.

For Adhya2, the network is the same as Adhya1, but because the number of specifications are much higher in Adhya2, the overlaps of the matrix variables in the sparse-BSOS hierarchy is much higher and more than 75% of their sizes.

7. Conclusion

In this paper, we studied the sparse-BSOS hierarchy introduced in [23]. We first showed how to find a splitting of variables for a general polynomial optimization problem that satisfies the *running intersection property*. Then, we modified this hierarchy to handle the problems with equality constraints. The results in this paper has been implemented in a Julia 0.5 package "Polyopt" to solve a polynomial optimization problem.

In the numerical results we compared the sparse-BSOS hierarchy with the BSOS one. According to this comparison, the problems in each level of the sparse-BSOS hierarchy can be solved faster

	iteration	SBSOS with elimination	SBSOS
Foulds2	d=1	−1,200.00(1) [25.04]	−1,200.00(3) [28.44]
	d=2	−1,191.28(1) [53.07]	−1,182.79(3) [62.63]
	d=3	−1,102.56(1) [126.04]	−1,102.38(3) [127.87]
Adhya1	d=1	−999.32(1) [0.59]	−999.32(2) [35.81]
	d=2	−716.84(1) [1.38]	−989.20(2) [90.68]
	d=3	−574.91(1) [23.10]	≈−688.36(2) [241.31]
Adhya4	d=1	−1055.00(1) [0.02]	−1003.33(3) [0.07]
	d=2	−1,040.00(1) [0.30]	−1003.33(3) [0.29]
	d=3	−907.96(1) [253.41]	−974.54(3) [53.69]

Table 4: The result of solving the pooling problem instances with sparse-BSOS hierarchy when the blocks are merged if the intersection size is larger than 75% of the size of smallest block (the number in the parentheses shows the number of blocks).

than the BSOS one if the network of the pooling problem is sparse enough, like the instance Foulds2. The quality of the lower bounds we get from the sparse-BSOS hierarchy can sometimes be worse than the BSOS hierarchy. The modification we proposed to the BSOS and sparse-BSOS hierarchies to handle equality constraints was tested on the pooling problem instances, and it could sometimes yield much better lower bounds than the original hierarchies, like the first and second levels of the hierarchies in the DeyGupte4 instance.

References

[1] A. A. Ahmadi, A. Majumdar, Some applications of polynomial optimization in operations research and real-time decision making, Optimization Letters 10 (4) (2016) 709–729.

[2] M. Alfaki, Models and solution methods for the pooling problem, Ph.D. thesis, University of Bergen (2012).

[3] M. Alfaki, D. Haugland, Strong formulations for the pooling problem, Journal of Global Optimization 56 (3) (2013) 897–916.

[4] P. R. Amestoy, T. A. Davis, I. S. Duff, An approximate minimum degree ordering algorithm, SIAM Journal on Matrix Analysis and Applications 17 (4) (1996) 886–905.

[5] J. R. S. Blair, B. Peyton, An Introduction to Chordal Graphs and Clique Trees, Springer New York, New York, NY, (1993) 1–29. doi:10.1007/978-1-4613-8369-7_1.

[6] S. S. Dey, A. Gupte, Analysis of MILP techniques for the pooling problem, Operations Research 63 (2) (2015) 412–427.

[7] L. Frimannslund, M. El Ghami, M. Alfaki, D. Haugland, Solving the pooling problem with LMI relaxations, S. Cafieri, B.G.-Tóth, E. Hendrix, L. Liberti and F. Messine (Eds), Proceedings of the Toulous Global Optimization Workshop (TOGO) (2010) 51–54.

[8] M. Fukuda, M. Kojima, K. Murota, K. Nakata, Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework, SIAM Journal on Optimization 11 (3) (2001) 647–674.

[9] A. Gupte, S. Ahmed, S. S. Dey, M. S. Cheon, Relaxations and discretizations for the pooling problem, Journal of Global Optimization (2016) 1–39.

[10] C. A. Haverly, Studies of the behavior of recursion for the pooling problem, ACM SIGMAP Bulletin (25) (1978) 19–28.

[11] S. Kim, M. Kojima, H. Waki, Generalized Lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems, SIAM Journal on Optimization 15 (3) (2005) 697–719.

[12] J.-L. Krivine, Anneaux préordonnés, Journal d'analyse mathématique 12 (1) (1964) 307–326.

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

[14] J. B. Lasserre, Polynomial programming: LP-relaxations also converge, SIAM Journal on Optimization 15 (2) (2005) 383–393.

[15] J. B. Lasserre, K. Toh, S. Yang, A bounded degree SOS hierarchy for polynomial optimization, EURO Journal on Computational Optimization (2015) 1–31.

[16] A. Marandi, J. Dahl, E. de Klerk, A numerical evaluation of the bounded degree sum-of-squares hierarchy of Lasserre, Toh, and Yang on the pooling problem, Annals of Operations Research (2017) 1–26. doi:10.1007/s10479-017-2407-5.

[17] R. Misener, C. A. Floudas, Advances for the pooling problem: modeling, global optimization, and computational studies, Applied and Computational Mathematics 8 (1) (2009) 3–22.

[18] MOSEK ApS, The mosek optimization tools version 3.2 (revision 8) users manual and reference (2002).

[19] P. M. Pardalos, S. A. Vavasis, Quadratic programming with one negative eigenvalue is NP-hard, Journal of Global Optimization 1 (1) (1991) 15–22.

[20] N. V. Sahinidis, M. Tawarmalani, Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints, Journal of Global Optimization 32 (2) (2005) 259–280.

[21] L. Vandenberghe, M. S. Andersen, Chordal graphs and semidefinite optimization, Foundations and Trends in Optimization 1 (4) (2015) 241–433. doi:10.1561/2400000006.

[22] 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) (2006) 218–242.

[23] T. Weisser, J. B. Lasserre, K. Toh, A bounded degree SOS hierarchy for large scale polynomial optimization with sparsity, arXiv:1607.01151.