How good is the Bounded Degree Sum-of-Squares Hierarchy of Lasserre, Toh, and Yang?
A numerical evaluation on the pooling problem

Ahmadreza Marandi, Joachim Dahl, Etienne de Klerk

Received: May 11, 2016/ Accepted: date

Abstract The bounded degree sum-of-squares (BSOS) hierarchy of Lasserre, Toh, and Yang [EURO J. Comput. Optim., to appear] constructs lower bounds for a general polynomial optimization problem with compact feasible set, by solving a sequence of semi-definite programming (SDP) problems. Lasserre, Toh, and Yang prove that these lower bounds converge to the optimal value of the original problem, under some assumptions. In this paper, we analyze the BSOS hierarchy and study its numerical performance on a specific class of bilinear programming problems, called pooling problems, that arise in the refinery and chemical process industries.

Keywords sum-of-squares hierarchy, Bilinear optimization, Pooling problem, Semidefinite programming

1 Introduction

Polynomial programming is the class of nonlinear optimization problems involving polynomials only:

---

A. Marandi
Tilburg School of Economics and Management, Tilburg University, The Netherlands
E-mail: a.marandi@uvt.nl
The research of this author is supported by EU Marie Curie Initial Training Network number 316647 (“Mixed Integer Nonlinear Optimization (MINO)”).

J. Dahl
MOSEK ApS, Copenhagen O, Denmark
E-mail: joachim.dahl@mosek.com

E. de Klerk
Tilburg School of Economics and Management, Tilburg University, The Netherlands
Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands
E-mail: E.deKlerk@uvt.nl
\[ f^* = \inf_{x \in \mathbb{R}^n} f(x) \]

\[ \text{s.t. } g_j(x) \geq 0, \quad j = 1, ..., m, \]

where \( f \) and all \( g_j \) are \( n \)-variate polynomials. We will assume throughout that

- the feasible set \( F = \{ x \in \mathbb{R}^n \mid g_j(x) \geq 0, \quad j = 1, ..., m \} \) is compact;
- for all \( x \in F \) one has \( g_j(x) < 1, \quad j = 1, ..., m \).

The second condition is theoretically without loss of generality (by scaling the \( g_j \)).

In general, these problems are \( \mathcal{NP} \)-hard, since they contain problems like the maximum cut problem as special cases; see e.g. [13]. In 2015, Lasserre, Toh and Yang [12] introduced the so-called bounded degree sum-of-squares (BSOS) hierarchy to obtain a nondecreasing sequence of lower bounds on the optimal value of problem (1) when the feasible set is compact. Each lower bound in the sequence is the optimal value of a semidefinite programming (SDP) problem. Moreover, the authors of [12] showed that, under some assumptions, this sequence converges to the optimal value of problem (1). From their numerical experiments, they concluded that the BSOS hierarchy was efficient for quadratic problems.

In this paper, we analyze the BSOS hierarchy in more detail. We also study variants of the BSOS hierarchy where the number of variables is reduced.

The numerical results in this paper are on pooling problems, that belong to the class of problems with bilinear functions. The pooling problem is well-studied in the chemical process and petroleum industries. It has also been generalised for application to wastewater networks; see e.g. [8]. It is a generalization of a minimum cost network flow problem where products possess different specifications. There are many equivalent mathematical models for a pooling problem and all of them include bilinear functions in their constraints. Haverly [7] described the so-called P-formulation, and afterwards many researchers used this model, e.g. [1], [3] and [4]. Also, there are Q-, PQ-, and TP-formulations; in this paper, we use the P-formulation and point the reader to the survey by Gupte et al. [6] where all the formulations are described, as well as the PhD thesis by Alfaki [2].

One way of getting a lower bound for a pooling problem is using convex relaxation, as done e.g. by Foulds et al. [4]. Similarly, Adhya et al. [1] introduced a Lagrangian approach to get tighter lower bounds for pooling problems. Also, there are many other papers studying duality [3], piecewise linear approximation [16], etc. A relatively recent survey on solution techniques is [15].

In a seminal paper in 2000, Lasserre [10] first introduced a hierarchy of lower bounds for polynomial optimization using SDP relaxations. Frimannslund et al. [5] tried to solve pooling problems with the LMI relaxations obtained by this hierarchy. They found that, due to the growth of the SDP problem sizes in the hierarchy, this method is not effective for the pooling problems. In this paper, we therefore consider the BSOS hierarchy as an alternative, since it is not so computationally intensive.
The structure of this paper is as follows: We describe the BSOS hierarchy in Section 2. In Section 3 the pooling problem is defined, and we review a mathematical model for it, namely the P-formulation. Also, we solve some pooling problems by the BSOS hierarchy in this section. Section 4 contains the numerical results after a reduction in the number of linear variables and constraints in each iteration of the BSOS hierarchy.

2 The bounded degree SOS (BSOS) hierarchy for polynomial optimization

In this section, we briefly review the background of the BSOS hierarchy from [12]. For easy reference, we will use the same notation as in [12].

In what follows \( N^k \) will denote all \( k \)-tuples of nonnegative integers, and we define

\[
N_d^k = \left\{ \alpha \in N^k : k \sum_{i=1}^k \alpha_i \leq d \right\}.
\]

The space of \( n \times n \) symmetric matrices will be denoted by \( S^n \), and its subset of positive semidefinite matrices by \( S^n_+ \).

Consider the general nonlinear optimization problem (1). For fixed \( d \geq 1 \), the following problem is clearly equivalent to (1):

\[
\begin{align*}
\min_x & \quad f(x) \\
\text{s.t.} & \quad \prod_{j=1}^m g_j(x)^{\alpha_j}(1 - g_j(x))^{\beta_j} \geq 0, \quad \forall (\alpha, \beta) \in N_d^{2m}.
\end{align*}
\]

The underlying idea of the BSOS hierarchy is to rewrite problem (1) as

\[
f^* = \sup_t \{ t : f(x) - t \geq 0 \ \forall x \in F \}.
\]

The next step is to use the following positivstellensatz by Krivine [9] to remove the quantifier “\( \forall x \in F \)”.

**Theorem 1** ([9], see also §3.6.4 in [13]) Assume that \( g_j(x) \leq 1 \) for all \( x \in F \) and \( j = 1, \ldots, m \), and \( \{1, g_1, \ldots, g_m\} \) generates the ring of polynomials. If a polynomial \( g \) is positive on \( F \) then

\[
g(x) = \sum_{(\alpha, \beta) \in N^{2m}} \lambda_{\alpha, \beta} \prod_{j=1}^m g_j(x)^{\alpha_j}(1 - g_j(x))^{\beta_j}
\]

for finitely many \( \lambda_{\alpha, \beta} > 0 \).

Defining

\[
h_{\alpha, \beta}(x) := \prod_{j=1}^m g_j(x)^{\alpha_j}(1 - g_j(x))^{\beta_j}, \quad x \in \mathbb{R}^n, \ \alpha, \beta \in N^m,
\]
we arrive at the following sequence of lower bounds (indexed by $d$) for problem (1):

$$f^* \geq \sup_t \left\{ t : f(x) - t = \sum_{(\alpha, \beta) \in \mathbb{N}_2} \lambda_{\alpha\beta} h_{\alpha\beta}(x) \right\}.$$  

For a given integer $d > 0$ the right-hand-side is a linear programming (LP) problem, and the lower bounds converge to $f^*$ in the limit as $d \to \infty$, by Krivine’s positivstellensatz. This hierarchy of LP bounds was introduced by Lasserre [11].

A subsequent idea, from [12] was to strengthen the LP bounds by enlarging its feasible set as follow: If we fixed $k \in \mathbb{N}$, and denote by $\sum[x]_k$ the space of sums of squares polynomials of degree at most $2k$, then we may define the bounds

$$q^k_d := \sup_{t, \lambda} \left\{ t : f(x) - t - \sum_{(\alpha, \beta) \in \mathbb{N}_2^m} \lambda_{\alpha\beta} h_{\alpha\beta}(x) \in \Sigma[x]_k \right\}.$$  

The resulting problem is a semidefinite programming (SDP) problem, and the size of the positive semidefinite matrix variable is determined by the parameter $k$, hence the name bounded-degree sum-of-squares (BSOS) hierarchy. By fixing $k$ to a small value, the resulting SDP problem is not much harder to solve that the preceding LP problem, but potentially yields a better bound for given $d$.

For fixed $k$ and for each $d$, one has

$$q^k_d = \sup_{t, \lambda, Q} t \left\{ f(x) - \sum_{(\alpha, \beta) \in \mathbb{N}_2^m} \lambda_{\alpha\beta} h_{\alpha\beta}(x) - t = \text{trace} \left( Q v_k(x) v_k(x)^T \right), \right.$$  

$$Q \in S_{s(k)}^o, \quad \lambda \geq 0,$$

where $s(k) = \binom{n+k}{k}$, and $v_k(x)$ is a vector with a basis for the $n$-variate polynomials up to degree $k$.

We may eliminate the variables $x$ in two ways to get an SDP problem:

- Equate the coefficients of the polynomials on both sides of the equality in (3), i.e. use the fact that two polynomials are identical if they have the same coefficients in some basis.
- Use the fact that two $n$-variate polynomials of degree $\tau$ are identical if their function values coincide on a finite set of $s(\tau) = \binom{n+\tau}{\tau}$ points in general position.

The second way of obtaining an SDP problem is called the ‘sampling formulation’, and was first studied in [14]. It was also used for the numerical BSOS hierarchy calculations in [12], with a set of $s(\tau)$ randomly generated points in $\mathbb{R}^n$. 
We will instead use the points
\[
\Delta(n, \tau) = \left\{ x \in \mathbb{R}^n \middle| \tau x \in \mathbb{N}^n, \sum_{i=1}^n x_i \leq 1 \right\},
\]
where \( \tau = \max \{\text{deg}(f), 2k, d \max_j \text{deg}(g_j)\} \).

Thus we obtain the following SDP reformulation of (3):
\[
q^k_d = \sup_{t, \lambda, Q} t 
\quad f(x) - \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) - t = \text{trace} (Qv_k(x)v_k(x)^T), \forall x \in \Delta(n, \tau)
\quad Q \in S_+^{\alpha(k)}, \lambda \geq 0.
\]

(4)

The following theorem, proved in [12], gives some information on feasibility and duality issues for the BSOS relaxation.

**Theorem 2 ([12])** If problem (1) is Slater feasible, then so is the dual SDP problem of (4). Thus (by the conic duality theorem), if the SDP problem (4) has a feasible solution, it has an optimal solution as well.

Note that problem (4) may be infeasible for given \( d \) and \( k \). One only knows that it will be feasible, and therefore \( q^k_d \) will be defined, for sufficiently large \( d \).

**Remark 1** Assume that at the \( d \)-th level of the hierarchy we have \( q^k_d = f^* \), i.e. finite convergence of the BSOS hierarchy, then
\[
f(x) - f^* = \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) + v_k(x)^TQv_k(x) \quad \forall x \in \mathbb{R}.
\]

(5)

Let \( x^* \in F \) be an optimal solution \( f(x^*) = f^* \), then it is clear from (5) that
\[
0 = \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x^*) + v_k(x^*)^TQv_k(x^*) \quad \forall x \in \mathbb{R},
\]
and due to the fact that \( Q \) is positive semi-definite, then
\[
\lambda_{\alpha\beta} h_{\alpha\beta}(x^*) = 0 \quad \forall (\alpha, \beta) \in \mathbb{N}_d^{2m}.
\]

(6)

Hence, for an \((\alpha, \beta) \in \mathbb{N}_d^{2m}\), if \( h_{\alpha\beta}(x) \) is not binding at an optimal solution, then \( \lambda_{\alpha\beta} = 0 \). We will use this observation to reduce the number of variables later on. \( \square \)
3 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 [6]. To define the pooling problem, consider an acyclic directed graph $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. This graph defines a pooling problem if:

i) the set $\mathcal{N}$ can be partitioned into three subsets $\mathcal{I}, \mathcal{L}$ and $\mathcal{J}$, where $\mathcal{I}$ is the set of inputs with $I$ members, $\mathcal{L}$ is the set of pools with $L$ members and $\mathcal{J}$ is the set of outputs with $J$ members.

ii) $\mathcal{A} \subseteq (\mathcal{I} \times \mathcal{L}) \cup (\mathcal{I} \times \mathcal{J}) \cup (\mathcal{L} \times \mathcal{L}) \cup (\mathcal{L} \times \mathcal{J})$; see Figure 1.

In this paper, we consider cases where $\mathcal{A} \cap \mathcal{L} \times \mathcal{L} = \emptyset$, which is called standard pooling problem because there is no arc between the pools.

For each arc $(i, j) \in \mathcal{A}$, let $c_{ij}$ be the cost of sending a unit flow on this arc. For each node, there is possibly a capacity restriction, which is a limit.
for sum of the incoming (outgoing) flows to a node. The capacity restriction is denoted by $C_i$ for each $i \in N$. Also, there are some specifications for the inputs, e.g. the sulfur concentrations in them, which are indexed by $k$ in a set of specifications $K$ with $K$ members. By letting $y_{ij}$ be the flow from node $i$ to node $j$, $u_{ij}$ the restriction on $y_{ij}$ that can be carried from $i$ to $j$, and $p_{lk}$ the concentration value of $k$th specification in the pool $l$, the pooling problem can be written as the following optimization model:

$$\min_{y, p} \sum_{(i,j) \in A} c_{ij} y_{ij}$$

s.t.

$$\sum_{i \in I} y_{il} = \sum_{j \in J} y_{lj}, \quad l \in L$$ (8)

$$\sum_{j \in J \cup L} y_{ij} \leq C_i, \quad i \in I$$ (9)

$$\sum_{j \in J} y_{lj} \leq C_l, \quad l \in L$$ (10)

$$\sum_{j \in J} y_{ij} \leq C_j, \quad j \in J$$ (11)

$$0 \leq y_{ij} \leq u_{ij}, \quad (i,j) \in A$$ (12)

$$\sum_{l \in L} \lambda_{ik} y_{il} = p_{lk} \sum_{j \in J} y_{lj}, \quad l \in L, k \in K$$ (13)

$$\sum_{l \in L} \lambda_{lk} y_{ij} + \sum_{l \in L} p_{lk} y_{lj} \leq \mu_{jk}^{\text{max}} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K$$ (14)

$$\sum_{l \in L} \lambda_{lk} y_{ij} + \sum_{l \in L} p_{lk} y_{lj} \geq \mu_{jk}^{\text{min}} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K$$ (15)

where $\mu_{jk}^{\text{max}}$ and $\mu_{jk}^{\text{min}}$ are the upper and lower bound of the $k$th specification in output $j \in J$, and $\lambda_{ik}$ is the concentration of $k$th specification in the input $i$. Here is a short interpretation of the constraints:

(8): volume balance between the incoming and outgoing flows in each pool.
(9): capacity restriction for each input.
(10): capacity restriction for each pool.
(11): capacity restriction for each output.
(12): limitation on each flow.
(13): specification balance between the incoming and outgoing flows in each pool.
(14): upper bound of the output specification.
(15): lower bound of the output specification.

For a general pooling problem, the aforementioned model is called the P-formulation. It is clear that the P-formulation is a nonconvex quadratic optimization problem which is not easy to solve. In this paper, we are going to
use the BSOS hierarchy to find a sequence of a lower bounds that converges to the optimal value of the pooling problem.

The BSOS hierarchy is only defined for problems without equality constraints and the P-formulation has \((K+1)L\) equality constraints. The simplest way of eliminating equality constraints, is to replace each equality constraint by two inequalities; however, this process increases the number of constraints which is not favorable for the BSOS hierarchy.

In the following section, we discuss another way of eliminating equality constraints.

### 3.1 Solving pooling problems with the BSOS hierarchy

To solve the pooling problem with the BSOS hierarchy, we first need to find an equivalent model in the form (1). One way of doing so is eliminating the equality constraints (8) and (13), if possible.

#### 3.1.1 Eliminating equality constraints

In this part, we study a possible way of eliminating equality constraints (8) and (13), which can be written as follows:

\[
\begin{pmatrix}
1 & 1 & \ldots & 1 \\
\lambda_{11} & \lambda_{21} & \ldots & \lambda_{I1} \\
\lambda_{12} & \lambda_{22} & \ldots & \lambda_{I2} \\
\vdots & \vdots & \ddots & \vdots \\
\lambda_{1K} & \lambda_{2K} & \ldots & \lambda_{IK}
\end{pmatrix}
\begin{bmatrix}
y_{l1} \\
y_{l2} \\
\vdots \\
y_{lI}
\end{bmatrix}
= \sum_{j \in J} y_{lj}
\begin{bmatrix}
1 \\
p_{l1} \\
p_{l2} \\
\vdots \\
p_{lK}
\end{bmatrix}
\quad l \in \mathcal{L}.
\] (16)

Taking the QR decomposition of the coefficient matrix, we can write (16) as

\[
R
\begin{bmatrix}
y_{l1} \\
y_{l2} \\
\vdots \\
y_{lI}
\end{bmatrix}
= \left( \sum_{j \in J} y_{lj} \right) Q^T
\begin{bmatrix}
1 \\
p_{l1} \\
p_{l2} \\
\vdots \\
p_{lK}
\end{bmatrix}
\quad l \in \mathcal{L}.
\] (17)

Given a pool \(l\), if it has at least \(K+1\) incoming arcs, then the number of rows in \(R\) is fewer than the number of columns, so we can split the vector \(y_{jl}\) into two blocks, the first block contains \(K+1\) variables corresponding to the first \(K+1\) incoming arcs to the pool \(l\), and the second block contains the rest of the variables. Without loss of generality, we assume that the first block and the second block are \([y_{l1}, \ldots, y_{l(K+1)}]^T\) and \([y_{l(K+2)}, \ldots, y_{lI}]^T\), respectively.
Therefore, (17) can be written as

\[
\begin{pmatrix} R_1 & R_2 \end{pmatrix} \begin{bmatrix} y_{11} \\ \vdots \\ y_{(K+1)}l \\ \vdots \\ y_{ll} \end{bmatrix} = \left( \sum_{j \in J} y_{lj} \right) Q^T \begin{bmatrix} p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix},
\]

which is equivalent to

\[
R_1 \begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{ll} \end{bmatrix} = \left( \sum_{j \in J} y_{lj} \right) Q^T \begin{bmatrix} p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix} - R_2 \begin{bmatrix} y_{(K+2)}l \\ \vdots \\ y_{ll} \end{bmatrix}.
\]  

Due to the fact that $R_1$ is an upper triangular square matrix, (18) is a simple system of linear equalities and by solving it, we can replace $y_{il}, i = 1, \ldots, K + 1$ in the inequality constraints with the quadratic functions.

Up to this point, we have only considered a pool with at least $K + 1$ incoming arcs. Now, assume that for a pool $l$, the number of incoming arcs is at most $K$. In this case, the number of the columns in $R$ is smaller than the number of the rows. Without loss of generality, assume that the first $t$ inputs feed the pool $l$, and denote by $R_1$ and $R_2$ the first $t$ rows of $R$ and the remaining rows, respectively. This partitioning of $R$ corresponds to a partitioning of $Q$’s columns, denoted by $Q_1$ and $Q_2$. Because $R$ is an upper triangular matrix, $R_2 = 0$. It means that (17) can be rewritten as the following two equality systems:

\[
R_1 \begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{tl} \end{bmatrix} = \left( \sum_{j \in J} y_{lj} \right) Q^T_1 \begin{bmatrix} p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix},
\]

\[
0 = \left( \sum_{j \in J} y_{lj} \right) Q^T_2 \begin{bmatrix} p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix}.
\]

As $R_1$ is an upper triangular square matrix, it is easy to solve (19) and find the quadratic function that equals $[y_{1l}, y_{2l}, \ldots, y_{tl}]$. If, for a feasible solution, $\sum_{j \in J} y_{lj} = 0$ then it means that there is no outflow from pool $l$, which implies by (19) that there is no input to it and $p_{lk}$,
\[ k = 1, \ldots, K \] can attain any real values. So, among all of the possible values for \( p_{l_k} \) we choose one satisfying

\[
0 = Q_2^T \begin{bmatrix}
1 \\
p_{l1} \\
p_{l2} \\
\vdots \\
p_{lK}
\end{bmatrix},
\tag{21}
\]

which is a system of \( K \) variables and \( K - t + 1 \) linear equalities with \( t \geq 2 \).

Moreover, a feasible solution with the property \( \sum_{j \in J} y_{lj} \neq 0 \) definitely satisfies (21). So, instead of (20), we may solve (21), which may be done using the QR decomposition.

By solving (21), we may write \( [p_{lt}, \ldots, p_{lK}] \) as a linear function of \( [p_{l1}, \ldots, p_{l(t-1)}] \), and by substitution in (19), we find the quadratic function corresponding to \( [y_{1l}, y_{2l}, \ldots, y_{tl}] \).

**Remark 2** We emphasize that after these substitutions, the equivalent mathematical model to the pooling problem is still nonconvex quadratic problem.

### 3.1.2 First numerical Results

In this section, we study convergence of the BSOS hierarchy of lower bounds \( q_d^1 \) \( (d = 1, 2, \ldots) \) for pooling problems. First, it is worth pointing out the number of variables and constraints needed to compute \( q_d^1 \). The number of constraints, as it is mentioned in the previous section, is \( \binom{n+2d}{2d} \). Also, the number of linear variables is one more than the size of \( N_d^{2m} \), namely \( \binom{2m+d}{d} + 1 \).

Table 1 gives some information of the standard pooling problems we use in this paper. The GAMS files of the pooling problem instances that we use in this paper can be found on the website [http://www.ii.uib.no/~mohammeda/spooling/](http://www.ii.uib.no/~mohammeda/spooling/).

**Example 1** By way of example, we give the details for the first instance in Table 1, called *Haverly1*. Its optimal solution is shown in Figure 2, and the optimal value is \(-400 [1]\). The optimal flow from node \( i \) to node \( j \) is denoted by \( y_{ij}^* \) in Figure 2.

Note that the third input has only flows to the outputs. Thus, this problem has three inputs, one pool, two outputs and one specification. The mathemati-
How good is the BSOS hierarchy?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Optimal Value</th>
<th># Inputs</th>
<th># Output</th>
<th># Pools</th>
<th># Spec.</th>
<th># Var.</th>
<th># Const.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haverly1</td>
<td>-400</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>Haverly2</td>
<td>-600</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>Haverly3</td>
<td>-750</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>Ben-Tal4</td>
<td>-450</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>Ben-Tal5</td>
<td>-3500</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>29</td>
<td>54</td>
</tr>
<tr>
<td>Foulds2</td>
<td>-1100</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>18</td>
<td>38</td>
</tr>
<tr>
<td>Foulds3</td>
<td>-38</td>
<td>11</td>
<td>16</td>
<td>8</td>
<td>1</td>
<td>152</td>
<td>204</td>
</tr>
<tr>
<td>Foulds4</td>
<td>-38</td>
<td>11</td>
<td>16</td>
<td>8</td>
<td>1</td>
<td>152</td>
<td>204</td>
</tr>
<tr>
<td>Adhya1</td>
<td>-350.8</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>41</td>
<td>51</td>
</tr>
<tr>
<td>Adhya2</td>
<td>-350.8</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>15</td>
<td>66</td>
</tr>
<tr>
<td>Adhya3</td>
<td>-561.05</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>36</td>
<td>53</td>
</tr>
<tr>
<td>Adhya4</td>
<td>-877.6</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>51</td>
</tr>
<tr>
<td>RT2</td>
<td>-4391.83</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td>38</td>
</tr>
<tr>
<td>sppA0</td>
<td>Unknown *</td>
<td>20</td>
<td>15</td>
<td>10</td>
<td>24</td>
<td>161</td>
<td>816</td>
</tr>
</tbody>
</table>

Table 1: Some well known pooling problems and some of their details. The number of variables and constraints mentioned here are after elimination of equality constraints.

* There is no exact optimal value for this problem and the best bound that has been found is $[-36233.4, -35811.5]$.

![Diagram](image)

Fig. 2: Optimal solution for Haverly1
The reformulated model of this problem using the elimination process and scaling $x_1 := \frac{y_1}{200}$, $x_2 := \frac{y_2}{200}$, $x_3 := \frac{y_3}{200}$, $x_4 := \frac{y_4}{200}$, $x_5 := \frac{y_5}{200}$, is
\[
\begin{align*}
\text{min} & \quad -200x_2(15x_1 - 12) - 200x_3(15x_1 - 6) + 200x_4 - 1000x_5 \\
\text{s.t.} & \quad 1 \geq \frac{3}{4}(x_1 - 1)(x_2 + x_3) \\
& \quad 1 \geq \frac{1}{4}(3x_1 - 1)(x_2 + x_3) \\
& \quad 1 \geq 1 - 2(x_2 + x_4) \\
& \quad 1 \geq 1 - (x_3 + x_5) \\
& \quad 1 \geq 0.5(x_4 + x_2) - 0.4x_4 - 0.6x_1x_2 \geq 0 \\
& \quad 1 \geq 0.5(x_5 + x_3) - \frac{2}{3}x_5 - x_1x_3 \geq 0 \\
& \quad 1 \geq x_i \geq 0, \quad i = 1, \ldots, 5,
\end{align*}
\] (22)

where the leftmost inequalities are redundant.

The last step is to multiply the constraint functions by a factor 0.9 (any value in (0, 1) will do, but we used 0.9 for our computations), to ensure that the ‘$\leq$’ conditions hold with strict inequality on the feasible set. Thus, we define $q_1(x) = -0.9 \cdot \frac{3}{4}(x_1 - 1)(x_2 + x_3)$, etc.

We will use the BSOS hierarchy to find the optimal value of this example (Table 2 below).

The results for Haverly1 and the other pooling problems are listed in Table 2. All computations in this paper were carried out with MOSEK 7 on an Intel i7-4790 3.60GHz Windows computer with 16GB of RAM.

As it is clear, in order to compute $q^d_1$ we can have a large number of linear variables and constraints (depending of $d$), which affects the speed and the time we need to solve (4). In the coming section, we describe how one can reduce the number of linear variables and constraints at each level of the BSOS hierarchy significantly.
### 4 Reduction in the number of linear variables and constraints

In this section, we propose a method to reduce the number of linear variables and an upper bound for the number of linearly independent constraints in each iteration.

#### 4.1 Reduction in the number of variables

As it is mentioned in Remark 1, if we can identify constraints that are not binding at optimality, then we can reduce the number of variables.

Table 2: Results for computing the lower bounds $q_d^1$ for various pooling problem instances.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>d=1</th>
<th>d=2</th>
<th>d=3</th>
<th>$# \text{lin. var.}$</th>
<th>$\text{size of SDP var.}$</th>
<th>$# \text{const.}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haverly</td>
<td>0.00s</td>
<td>-600</td>
<td></td>
<td>23</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>0.03s</td>
<td>-417.2043</td>
<td></td>
<td>276</td>
<td>6</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>0.47s</td>
<td>-400.000</td>
<td></td>
<td>2,300</td>
<td>6</td>
<td>462</td>
</tr>
<tr>
<td>Haverly2</td>
<td>0.00s</td>
<td>-1,200</td>
<td></td>
<td>23</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>0.03s</td>
<td>-601.6680</td>
<td></td>
<td>276</td>
<td>6</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>0.50s</td>
<td>-600.000</td>
<td></td>
<td>2300</td>
<td>6</td>
<td>462</td>
</tr>
<tr>
<td>Haverly3</td>
<td>0.00s</td>
<td>-875</td>
<td></td>
<td>23</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>0.03s</td>
<td>-750.000</td>
<td></td>
<td>276</td>
<td>6</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>2.14s</td>
<td>-450.000</td>
<td></td>
<td>3,654</td>
<td>7</td>
<td>924</td>
</tr>
<tr>
<td>Ben-Tal4</td>
<td>0.00s</td>
<td>-650</td>
<td></td>
<td>27</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>0.03s</td>
<td>-467.2043</td>
<td></td>
<td>378</td>
<td>7</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td>2.14s</td>
<td>-450.000</td>
<td></td>
<td>3,654</td>
<td>7</td>
<td>924</td>
</tr>
<tr>
<td>Ben-Tal5</td>
<td>0.17s</td>
<td>-3500</td>
<td></td>
<td>169</td>
<td>30</td>
<td>465</td>
</tr>
<tr>
<td>Foulds2</td>
<td>0.03s</td>
<td>-1,200</td>
<td></td>
<td>77</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td></td>
<td>391.55s</td>
<td>-1,191.3</td>
<td></td>
<td>3,903</td>
<td>19</td>
<td>7,315</td>
</tr>
<tr>
<td></td>
<td>79,079</td>
<td>19</td>
<td>154,596</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Foulds3</td>
<td>444.75s</td>
<td>-64,000</td>
<td></td>
<td>409</td>
<td>153</td>
<td>11,781</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>83,845</td>
<td>153</td>
<td>23,738,715</td>
<td></td>
</tr>
<tr>
<td>Foulds4</td>
<td>298.59s</td>
<td>-64,000</td>
<td></td>
<td>409</td>
<td>153</td>
<td>11,781</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>83,845</td>
<td>153</td>
<td>23,738,715</td>
<td></td>
</tr>
<tr>
<td>Adhya1</td>
<td>0.02s</td>
<td>-999.3182</td>
<td></td>
<td>83</td>
<td>12</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>7.61s</td>
<td>-723.947</td>
<td></td>
<td>3,486</td>
<td>12</td>
<td>1,365</td>
</tr>
<tr>
<td></td>
<td>98,770</td>
<td>12</td>
<td>12,376</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adhya2</td>
<td>0.01s</td>
<td>-798.2904</td>
<td></td>
<td>107</td>
<td>12</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>15.88s</td>
<td>-576.827</td>
<td></td>
<td>5,778</td>
<td>12</td>
<td>1,365</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>209,934</td>
<td>12</td>
<td>12,376</td>
<td></td>
</tr>
<tr>
<td>Adhya3</td>
<td>0.03s</td>
<td>-882.844</td>
<td></td>
<td>133</td>
<td>18</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>466.58s</td>
<td>Numerical Prob.</td>
<td>8,911</td>
<td>18</td>
<td>5,985</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>400,995</td>
<td>18</td>
<td>100,947</td>
<td></td>
</tr>
<tr>
<td>Adhya4</td>
<td>0.01s</td>
<td>-1055</td>
<td></td>
<td>103</td>
<td>17</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>304.65s</td>
<td>-1034.9</td>
<td></td>
<td>5,356</td>
<td>17</td>
<td>4,845</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>187,460</td>
<td>17</td>
<td>74,613</td>
<td></td>
</tr>
<tr>
<td>RT2</td>
<td>0.03s</td>
<td>-45,420.2015</td>
<td></td>
<td>135</td>
<td>15</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>17.47s</td>
<td>Numerical Prob.</td>
<td>9,180</td>
<td>15</td>
<td>3,060</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>419,220</td>
<td>15</td>
<td>38,760</td>
<td></td>
</tr>
<tr>
<td>sppA0</td>
<td>602.48s</td>
<td>-47,674.946</td>
<td></td>
<td>1633</td>
<td>162</td>
<td>13,203</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>1,334,161</td>
<td>162</td>
<td>29,772,765</td>
<td></td>
</tr>
</tbody>
</table>
In particular, by construction the constraints \( g_j(x) \leq 1 \) will never be binding at optimality. Recalling that the variable \( \lambda_{\alpha\beta} \) corresponds to

\[
h_{\alpha\beta}(x) := \prod_{j=1}^{m} g_j(x)^{\alpha_j}(1 - g_j(x))^{\beta_j}, \quad x \in \mathbb{R}^n,
\]

we know from Remark 1 that, in case of finite convergence, we will have \( \lambda_{\alpha\beta} = 0 \) whenever \( \alpha = 0 \).

Hence, instead of solving (4) to compute \( q_d^k \), we may compute the following (weaker) bound more efficiently:

\[
\hat{q}_d^k := \max_{t, \lambda, Q} f(x) - \lambda_{\alpha\beta} h_{\alpha\beta}(x) - t \cdot \text{trace}(Q v_k(x) v_k(x)^T), \quad \forall x \in \Delta(n, \tau),
\]

\[
Q \in S^n_{+}, \quad \lambda \geq 0.
\]

The advantage of (23) is that it has \( \binom{m+d}{d} \) fewer nonnegative variables than (4). We emphasize that problem (23) is not equivalent to (4), i.e. the lower bounds \( q_d^k \) and \( \hat{q}_d^k \) are not equal in general — the bound \( \hat{q}_d^k \) is weaker, and may be strictly weaker.

The precise relation of the bounds \( q_d^k \) and \( \hat{q}_d^k \) is spelled out in the next theorem, which follows from the argument in Remark 1.

**Theorem 3** If, for given \( d \) and \( k \), \( q_d^k \) and \( \hat{q}_d^k \) are both finite, then \( \hat{q}_d^k \leq q_d^k \).

Moreover, if the sequence \( q_d^k \) (\( d = 1, 2, \ldots \)) from (4) converges finitely to \( f^* \), then so does \( \hat{q}_d^k \) (\( d = 1, 2, \ldots \)) from (23). More precisely, if \( q_d^k = f^* \) for some \( d^* \in \mathbb{N} \), then \( \hat{q}_d^k = f^* \).

It is important to note that finite convergence of the sequence \( q_d^k \) (\( d = 1, 2, \ldots \)) is not guaranteed in general. Sufficient conditions for finite convergence are described in [12].

The numerical results for using (23) for the pooling problem instances is demonstrated in Table 3. Note that there is a significant reduction in computational times when compared to Table 2.

4.2 Reduction in the number of constraints

As it was mentioned, the number of constraints in each level of the BSOS hierarchy is \( \binom{n+2d}{2d} \), where \( n \) is the number of variables in the original problem (1) and \( d \) is the level of the BSOS hierarchy. So, the number of constraints increases quickly with \( d \). In this subsection, we discuss the redundancy of constraints and how we can eliminate linearly dependent constraints.
<table>
<thead>
<tr>
<th>iteration</th>
<th>time</th>
<th>solution</th>
<th># lin. var.</th>
<th>size of SDP var.</th>
<th># const.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haverly1</td>
<td>d=1</td>
<td>0.00s</td>
<td>-600</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-417.2043</td>
<td>198</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.31s</td>
<td>-400.0007</td>
<td>1,936</td>
<td>6</td>
</tr>
<tr>
<td>Haverly2</td>
<td>d=1</td>
<td>0.02s</td>
<td>-1200</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-601.6680</td>
<td>198</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.39s</td>
<td>-600.002</td>
<td>1,936</td>
<td>6</td>
</tr>
<tr>
<td>Haverly3</td>
<td>d=1</td>
<td>0.02s</td>
<td>-875</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-750.0001</td>
<td>198</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>1.90s</td>
<td>-450.001</td>
<td>3,095</td>
<td>7</td>
</tr>
<tr>
<td>Ben-Tal4</td>
<td>d=1</td>
<td>0.02s</td>
<td>-1200</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-601.6680</td>
<td>198</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.39s</td>
<td>-600.002</td>
<td>1,936</td>
<td>6</td>
</tr>
<tr>
<td>Ben-Tal5</td>
<td>d=1</td>
<td>0.02s</td>
<td>-875</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-750.0001</td>
<td>198</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>1.90s</td>
<td>-450.001</td>
<td>3,095</td>
<td>7</td>
</tr>
<tr>
<td>Foulds2</td>
<td>d=1</td>
<td>0.02s</td>
<td>-1200</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-1191.3</td>
<td>2,224</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>66,419</td>
<td>19</td>
<td>134,596</td>
</tr>
<tr>
<td>Foulds3</td>
<td>d=1</td>
<td>0.02s</td>
<td>-1200</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-601.6680</td>
<td>198</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.39s</td>
<td>-600.002</td>
<td>1,936</td>
<td>6</td>
</tr>
<tr>
<td>Foulds4</td>
<td>d=1</td>
<td>0.02s</td>
<td>-1200</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.03s</td>
<td>-601.6680</td>
<td>198</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.39s</td>
<td>-600.002</td>
<td>1,936</td>
<td>6</td>
</tr>
<tr>
<td>Adhya1</td>
<td>d=1</td>
<td>0.05s</td>
<td>-999.3182</td>
<td>42</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>5.27s</td>
<td>-723.94</td>
<td>2,583</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>85,526</td>
<td>12</td>
<td>12,376</td>
</tr>
<tr>
<td>Adhya2</td>
<td>d=1</td>
<td>0.02s</td>
<td>-798.2905</td>
<td>54</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>11.23s</td>
<td>-576.82</td>
<td>4,293</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>182,214</td>
<td>12</td>
<td>12,376</td>
</tr>
<tr>
<td>Adhya3</td>
<td>d=1</td>
<td>0.05s</td>
<td>-882.8436</td>
<td>67</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>862.69s</td>
<td>-882.8436</td>
<td>6.634</td>
<td>18</td>
</tr>
<tr>
<td>Adhya4</td>
<td>d=1</td>
<td>0.03s</td>
<td>-1035</td>
<td>52</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>263.75s</td>
<td>-1034.9</td>
<td>3,979</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>162,657</td>
<td>17</td>
<td>74,613</td>
</tr>
<tr>
<td>RT2</td>
<td>d=1</td>
<td>0.03s</td>
<td>-45,420.2015</td>
<td>68</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>37.4s</td>
<td>-45,420.2015</td>
<td>6,836</td>
<td>15</td>
</tr>
<tr>
<td>sppA0</td>
<td>d=1</td>
<td>394.93s</td>
<td>-47,674.805</td>
<td>817</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>-</td>
<td>1,000,008</td>
<td>162</td>
<td>29,772,765</td>
</tr>
</tbody>
</table>

Table 3: Results for computing the lower bounds $\hat{q}_d$ for pooling problems using (23).

Let $svec$ denote the map from the $n \times n$ symmetric matrix space $S^{n+1}$ to $\mathbb{R}^1 \times \mathbb{R}^{n^2}$ given by

$$svec(X) = \begin{bmatrix} X_{11}, \sqrt{2}X_{12}, \ldots, \sqrt{2}X_{1(n+1)}, X_{(n+1)(n+1)} \end{bmatrix}, \quad \forall X \in S^{n+1}.$$

It will also be convenient to number the elements of $\Delta(n, \tau)$ as $x_1, \ldots, x_L$ where $L = s(\tau)$. Finally, we will use the notation $|\beta| = \sum |\beta_i|$.

So, for $d \geq 1$ and $k = 1$ we may write the linear equality constraints in (4) as $H_d y_d = b_d$, where

$$H_d = \begin{bmatrix} 1 \left( h_{\alpha \beta}(x^1) \right)_{(\alpha, \beta) \in \mathbb{N}^2_d} & svec(v_1(x^1)v_1(x^1)^T) \\ \vdots & \vdots \\ 1 \left( h_{\alpha \beta}(x^L) \right)_{(\alpha, \beta) \in \mathbb{N}^2_d} & svec(v_1(x^L)v_1(x^L)^T) \end{bmatrix}.$$
\[ b_d = \begin{bmatrix} f(x^1) \\ \vdots \\ f(x^L) \end{bmatrix}, \quad y_d = \begin{bmatrix} t \\ (\lambda_{\alpha\beta})_{(\alpha,\beta)\in\mathbb{N}^2} \end{bmatrix}. \]

and \( L = \binom{n+2d}{2d} \). It clear that \( H_d \in \mathbb{R}^{L \times \left( \binom{2m+d}{d} + L + 1 \right)} \).

In the following theorem we prove that all the constraints are linearly independent when \( d = 1 \).

**Theorem 4** For the general problem (1) with quadratic functions \( f(x) \) and \( g_j(x) \), \( j = 1, \ldots, m \), all of the constraints in the first iteration of the BSOS hierarchy are linearly independent, i.e. if \( d = 1 \), all of the constraints of (4) are linearly independent.

**Proof** Fix \( d = 1 \), which implies \( \tau = 2 \) and \( L = \binom{n+2}{2} \) in (4). Then,

\[
H_1 = \begin{bmatrix}
1 & g_1(x^1) & \cdots & g_m(x^1) & 1 - g_1(x^1) & \cdots & 1 - g_m(x^1) & \text{svec} \left( v_1(x^1)v_1(x^1)^T \right) \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & g_1(x^L) & \cdots & g_m(x^L) & 1 - g_1(x^L) & \cdots & 1 - g_m(x^L) & \text{svec} \left( v_1(x^L)v_1(x^L)^T \right)
\end{bmatrix},
\]

and,

\[
b_1 = \begin{bmatrix} f(x^1) \\ \vdots \\ f(x^L) \end{bmatrix}, \quad y_1 = \begin{bmatrix} t \\ (\lambda_{\alpha\beta})_{(\alpha,\beta)\in\mathbb{N}^2} \end{bmatrix},
\]

for \( x^1, \ldots, x^L \in \Delta(n, 2) \), defined in (4). To show that all of the rows in \( H_1 \) are linearly independent, we prove that the submatrix

\[
V_1^n = \begin{bmatrix} \text{svec} \left( v_1(x^1)v_1(x^1)^T \right) \\ \vdots \\ \text{svec} \left( v_1(x^L)v_1(x^L)^T \right) \end{bmatrix} \in \mathbb{R}^{\left( \binom{n+2}{2} \right) \times \left( \binom{n+2}{2} \right) = \mathbb{R}^{L \times L},
\]

is a full rank matrix by induction over \( n \), the dimension of \( x \). Assume that \( n = 1 \). Because \( \Delta(1, 2) = \{0, \frac{1}{2}, 1\} \), it is clear that rank of the matrix \( V_1^1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & \frac{\sqrt{2}}{2} & 1 \\ 1 & 1 & 1 \end{bmatrix} \), is 3, which means that \( V_1^1 \) is a full rank matrix.

Now, suppose that \( V_1^n \) is a full rank matrix, and let us show it is full rank for \( n + 1 \). When \( x \in \mathbb{R}^{n+1} \), we can partition the points in \( \Delta(n+1, 2) \) into three cases:

I) points with \( x_{n+1} = 0 \). These points can be generated by considering all of the points in \( \Delta(n, 2) \), and adding a 0 as their last component.

II) points with \( x_{n+1} = \frac{1}{2} \). The points in this class can be sub-partitioned into two groups:

i) points with one nonzero component.

ii) points with two nonzero components.
III) points with $x_{n+1} = 1$. Clearly, there is only one point in this class.

According to the definition of $\text{svec}(v_1(x)v_1(x)^T)$, each of $V_{n+1}^1$’s column is related to $x^\gamma$, where $\gamma \in \mathbb{N}_2^{n+1}$. Let us order the columns of $V_{n+1}^1$ as follows: first we put all of the columns related to $x^\alpha$, where $(\alpha,0) \in \mathbb{N}_2^{n+1}$, after that the columns related to $x_{n+1}, x_{n+1}^2, x_{n+1}x_i, i = 1, ..., n$. So, because each row of $V_{n+1}^1$ is related to a point in $\Delta(n+1,2)$, after ordering its rows, the matrix looks like this:

$$V_{n+1} = \begin{pmatrix}
    x^\alpha_{(\alpha,0) \in \mathbb{N}_2^{n+1}} & x_{n+1}^2 & x_{n+1}x_i \\
    V_{n+1}^1 & 0_{L \times 1} & 0_{L \times 1} & 0_{L \times n} \\
    a_1 & \frac{\sqrt{2}}{2} & \frac{1}{4} & \frac{\sqrt{2}}{2}I_n \\
    a_2 & \frac{\sqrt{2}}{2} & \frac{1}{4} & 0_{1 \times n} \\
    a_3 & \sqrt{2} & 1 & 0_{1 \times n}
\end{pmatrix},$$

for some $a_1 \in \mathbb{R}^{n \times L}$, and $a_2, a_3 \in \mathbb{R}^{1 \times L}$. Due to the induction assumption, $V_{n+1}^1$ is a full rank matrix, which implies that $V_{n+1}^1$ is a full rank matrix. Therefore, the constraints in the first iteration of the BSOS hierarchy are linearly independent.

In Theorem 4, we prove that if $d = 1$, then all of the constraints in (4) are linearly independent. In the next theorem, we prove that for $d \geq 2$, if we rewrite $H_d$ as $[\bar{H}_d, V_n^d]$, where

$$V_n^d = \begin{pmatrix}
    \text{svec}(v_1(x)v_1(x)^T) \\
    \vdots \\
    \text{svec}(v_1(x^L)v_1(x^L)^T)
\end{pmatrix} \in \mathbb{R}^{L \times L},$$

then $\text{Rank}(H_d) = \text{Rank}(\bar{H}_d)$.

**Theorem 5** Suppose that $f$ is quadratic, $d \geq 2$, and $\Theta \subseteq \Delta(n,2d)$. The equality constraints in (4) corresponding to the points in $\Theta$ applied to the general problem (1) with sign constraints over all of the variables, are linearly independent if and only if rows in $\bar{H}_d$ corresponding to the points in $\Theta$ are linearly independent.

**Proof** The ‘if’ part is trivial. To prove the ‘only if’ part, without loss of generality we assume that $x^p$, $p = 1, ..., t$ generate linearly independent constraints, which means that the first $t$ rows of $H_d$ are linearly independent. Since the objective function $f$ is quadratic, $b_d$ is a linear combination of the columns of $V_n^d$. Because of the sign constraints for all of the variables, each column of $V_n^d$ is also a column in $\bar{H}_d$, for $d \geq 2$. This means that $V_n^d$ is a submatrix of $\bar{H}_d$, which implies that the first $t$ rows in $\bar{H}_d$ are linearly independent.

After elimination of the equality constraints in pooling problem (7), we rewrite the model with sign constraints over all of the remaining variables. So, when using Theorem 5 to find the linearly independent constraints, we only need to check $\bar{H}_d$. 

$\square$
Theorem 6 Fix \(d \geq 2\). Consider \(\hat{H}_d\), which is a matrix with all of the columns of \(\hat{H}_d\) related to \((\alpha, \beta)\) with \(\beta = 0\). Then \(\text{Range}(H_d) = \text{Range}(\hat{H}_d)\).

Proof Since we consider \(\beta = 0\), we can write \(\hat{H}_d\) as follows:

\[
\hat{H}_d = \begin{bmatrix}
(g(x^1)\alpha)_{\alpha \in \mathbb{N}_d^m} \\
\vdots \\
(g(x^L)\alpha)_{\alpha \in \mathbb{N}_d^m}
\end{bmatrix},
\]

where \(L = \left\lceil \frac{n+2d}{2d} \right\rceil\), \(g(x) = (g_1(x), \ldots, g_m(x))\), for each \(\alpha \in \mathbb{N}_d^m\), \(g(x)^\alpha = \prod_{j=1}^m g_j(x)^{\alpha_j}\), and \((g(x)^\alpha)_{\alpha \in \mathbb{N}_d^m} \in \mathbb{R}^{1 \times (\sum_{j=1}^m d^j)}\), \(p = 1, \ldots, L\).

Because the columns of \(\hat{H}_d\) are a subset of the columns of \(\hat{H}_d\), so \(\text{Range}(\hat{H}_d) \subseteq \text{Range}(H_d)\). To prove the other side, we show that all columns of \(\hat{H}_d\) are linear combinations of \(\hat{H}_d\)'s columns. Each column of \(\hat{H}_d\) is related to a function \(h_{\alpha,\beta}(x)\) for some \((\alpha, \beta) \in \mathbb{N}_d^m\). If \(\beta = 0\) for a column of \(\hat{H}_d\), then it is a column of \(\hat{H}_d\). Now consider a column with \(\beta \neq 0\). Therefore, \(h_{\alpha,\beta}(x)\) related to this column is equal to

\[
\prod_{j=1}^m g_j(x)^{\alpha_j} \prod_{j=1}^t (1 - g_j(x))^{\beta_j} = g(x)^\alpha \left(\sum_{i=1}^w a_i g(x)^\gamma_i\right),
\]

for some \(\gamma_i \in \mathbb{N}_d^m\), \(a_i \in \mathbb{R}\), \(i = 1, \ldots, w\), and \(w \geq 0\). Hence, \(h_{\alpha,\beta}(x) = \sum_{i=1}^w a_i g(x)^{\gamma_i+\alpha}\). Because \(\gamma_i + \alpha \in \mathbb{N}_d^m\), \(g(x)^{\gamma_i+\alpha}\) is related to a column of \(H_d\), for each \(i = 1, \ldots, w\). This means that any column of \(\hat{H}_d\) is a linear combination of the columns in \(H_d\). \(\square\)

By Theorem 6, to find the number of linearly independent constraints in (4), we only need to check the columns related to \(h_{\alpha,\beta}(x)\) with \(\beta = 0\). In Table 4, the results of solving the pooling problems in Table 1 are shown after eliminating the linearly dependent constraints. Note that the computational times at the \(d = 2\) and \(d = 3\) levels are greatly reduced when compared to the times in Table 3.

4.3 Upper bound for the number of linearly independent constraints

According to Theorem 6, to find the number of linearly independent columns of \(H_d\), for \(d \geq 2\) we only need to find the rank of the linear space, say \(N_d\), spanned by \(\{g(x)^\alpha\}_{\alpha \in \mathbb{N}_d^m}\). Hence, the dimension of \(N_d\) is an upper bound on the number of linearly independent constraints. In this part we give an upper bound for the dimension of \(N_d\), which is an upper bound for the number of linearly independent constraints in (4).

It is clear that \(N_d\) is a subspace of the linear space \(M_d\) spanned by \(\{w(x)^\alpha\}_{\alpha \in \mathbb{N}_d^m}\), where \(w(x)\) is a vector containing all of the monomial existing in (1), and \(\omega\) in the size of \(w(x)\). Therefore, \(\text{rank}(M_d)\) is an upper bound for \(\text{rank}(N_d)\), and
<table>
<thead>
<tr>
<th>Method</th>
<th>iteration</th>
<th>time</th>
<th>solution</th>
<th># lin. var.</th>
<th>size of SDP var.</th>
<th># const.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haverly1</td>
<td>d=1</td>
<td>0.01s</td>
<td>-690</td>
<td>11</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.00s</td>
<td>-417,2043</td>
<td>199</td>
<td>6</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.06s</td>
<td>-400,000</td>
<td>1937</td>
<td>6</td>
<td>98</td>
</tr>
<tr>
<td>Haverly2</td>
<td>d=1</td>
<td>0.00s</td>
<td>-1200</td>
<td>11</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.00s</td>
<td>-601,6680</td>
<td>199</td>
<td>6</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.06s</td>
<td>-600,002</td>
<td>1937</td>
<td>6</td>
<td>98</td>
</tr>
<tr>
<td>Haverly3</td>
<td>d=1</td>
<td>0.02s</td>
<td>-875</td>
<td>11</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.02s</td>
<td>-750,001</td>
<td>199</td>
<td>6</td>
<td>33</td>
</tr>
<tr>
<td>Ben-Tal4</td>
<td>d=1</td>
<td>0.01s</td>
<td>-650</td>
<td>14</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.02s</td>
<td>-467,2043</td>
<td>274</td>
<td>7</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>0.11s</td>
<td>-450,00</td>
<td>3,095</td>
<td>7</td>
<td>140</td>
</tr>
<tr>
<td>Ben-Tal5</td>
<td>d=1</td>
<td>0.09s</td>
<td>-3500</td>
<td>55</td>
<td>30</td>
<td>465</td>
</tr>
<tr>
<td>Foulds2</td>
<td>d=1</td>
<td>0.03s</td>
<td>-1200</td>
<td>39</td>
<td>19</td>
<td>190</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.75s</td>
<td>-1191.3</td>
<td>2,224</td>
<td>19</td>
<td>295</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>-</td>
<td>49,385</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>Foulds3</td>
<td>d=1</td>
<td>222,33s</td>
<td>-64</td>
<td>205</td>
<td>153</td>
<td>11,781</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>-</td>
<td>-</td>
<td>62,730</td>
<td>153</td>
<td>-</td>
</tr>
<tr>
<td>Foulds4</td>
<td>d=1</td>
<td>228,42s</td>
<td>-64</td>
<td>205</td>
<td>153</td>
<td>11,781</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>-</td>
<td>-</td>
<td>62,730</td>
<td>153</td>
<td>-</td>
</tr>
<tr>
<td>Adhya1</td>
<td>d=1</td>
<td>0.06s</td>
<td>-999,3182</td>
<td>42</td>
<td>12</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>0.51s</td>
<td>-723.94</td>
<td>2,583</td>
<td>12</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>-</td>
<td>85,526</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>Adhya2</td>
<td>d=1</td>
<td>0.02s</td>
<td>-798,2905</td>
<td>54</td>
<td>12</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>1.08s</td>
<td>-576.8</td>
<td>4,293</td>
<td>12</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>-</td>
<td>182,214</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>Adhya3</td>
<td>d=1</td>
<td>0.05s</td>
<td>-882,8436</td>
<td>67</td>
<td>18</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>8.47s</td>
<td>-802.8</td>
<td>6,634</td>
<td>18</td>
<td>671</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>-</td>
<td>348,602</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>Adhya4</td>
<td>d=1</td>
<td>0.03s</td>
<td>-1055</td>
<td>52</td>
<td>17</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>4.91s</td>
<td>-1035.11</td>
<td>3,979</td>
<td>18</td>
<td>732</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>-</td>
<td>162,657</td>
<td>17</td>
<td>-</td>
</tr>
<tr>
<td>RT2</td>
<td>d=1</td>
<td>0.03s</td>
<td>-45,429.2015</td>
<td>68</td>
<td>15</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>1.34s</td>
<td>-36,542</td>
<td>6836</td>
<td>15</td>
<td>266</td>
</tr>
<tr>
<td></td>
<td>d=3</td>
<td>-</td>
<td>-</td>
<td>364,480</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>sppA0</td>
<td>d=1</td>
<td>104.6s</td>
<td>-286,040</td>
<td>817</td>
<td>162</td>
<td>13,203</td>
</tr>
<tr>
<td></td>
<td>d=2</td>
<td>-</td>
<td>-</td>
<td>1,000,008</td>
<td>162</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: Results for computing the bounds $q^1_d$ in (23) after elimination of linearly dependent constraints.

hence an upper bound of the number of linearly independent constraints in each iteration of the BSOS hierarchy.

In the rest of this part, we try to find $\text{rank}(M_d)$ for the pooling problems, and assume that the number of outgoing flows from each pool is equal to $J$. After elimination of equality constraints in the pooling problem (7), the functions defining the inequality constraints can be partitioned into three classes:

I) bilinear functions,
II) $x_i$, $i = 1, \ldots, n$,
III) some other affine functions.
The bilinear functions are those related to constraints (14) and (15), or those related to the constraints (12) after elimination of equality constraints. Hence, the only bilinear terms in the reformulated problem are $p_{lk}y_{lj}$, for each pool $l$ and specification $k$, where there is an outgoing flow from pool $l$ to output $j$. Therefore,

$$\left\langle \left\{ \begin{array}{l}
\{(1, \{y_{il}\}_{i \in I}, \{y_{lj}\}_{j \in J}, \{p_{lk}\}_{(l,k) \in \bar{L}}\} \right.
\end{array} \right\} : \alpha \in \mathbb{N} \right\rangle = M_d,$$

where $\bar{I}$, $\bar{L}$, and $\bar{J}$ are respectively including $(i, l)$, $(l, k)$ and $(l, k, j)$ that $y_{il}$, $p_{lk}$ and $p_{lk}y_{lj}$ appear in (7) after elimination of the equality constraints, and $\omega = 1 + I \times L + L \times J + |\bar{I}| + |\bar{L}| + |\bar{J}|$.

Clearly the number of variables in the pooling problem (7) after elimination of equality constraints is $I \times L + L \times J + |\bar{I}| + |\bar{L}| + |\bar{J}|$. For $d = 1$, we prove in Theorem 4 that all of the constraints in (4) are linearly independent, with the number of $\binom{n+2}{2}$. For $d \geq 2$, we are seeking for the monomials up to degree $2d$ that appear in $M_d$. If $d = 2$, the number of monomials with degree at most 2 is $\binom{n+2}{2}$. The number of monomials with degree 3 that appear in $M_d$ is at most $K \times L \times \left[ \binom{n+1}{2} - \binom{n-J+1}{2} \right]$, because for each $k \in K$ and $l \in L$, in this case the only way of having a monomial with degree 3 is by multiplying a monomial of degree 2 with a variable, which makes $\binom{n+1}{2} - \binom{n-J+1}{2}$ monomials of degree 3. And finally, the number of monomials of degree 4 that appear in $M_d$ is $\left[ K \times L \times J \right] + K \times L \times J$, because the only ways to make such monomials are by taking the square of a monomial with degree 2, or multiplying two degree 2 monomials. Therefore, the number of linearly independent constraints for $d = 2$ is at most

$$\binom{n+2}{2} + K \times L \times \left[ \binom{n+1}{2} - \binom{n-J+1}{2} \right] + \left( K \times L \times J \right) + K \times L \times J. \quad (24)$$

With the same line of reasoning as above, the number of monomials with degree at most 6 for $d = 3$ is less than or equal to

$$\binom{n+3}{3} + K \times L \times \left[ \binom{n+2}{3} - \binom{n-J+2}{3} \right]$$

$$+ K \times L \times \left[ \binom{n+2}{3} - \binom{n-J+2}{3} - J \times \binom{n-J+1}{2} \right]$$

$$+ \left[ K \times L \times J \right] + K \times L \times J + 2 \left( K \times L \times J \right). \quad (25)$$
Example 2 Consider the example (22). The only bilinear terms in (22) are $y_1y_2$ and $y_1y_3$. So,

$$M_d = \left\{ (1, y_1, y_2, y_3, y_4, y_1y_2, y_1y_3) \alpha \in \mathbb{N}_8^2 \right\}$$

Therefore, the number of linearly independent constraints is at most

$$\left(\begin{array}{c}
7 \\
2 
\end{array}\right) + \left(\begin{array}{c}
6 \\
2 
\end{array}\right) - \left(\begin{array}{c}
4 \\
2 
\end{array}\right) + 2 = 33,$$

if $d = 2$, and

$$\left(\begin{array}{c}
8 \\
3 
\end{array}\right) + 2 \times \left(\begin{array}{c}
7 \\
3 
\end{array}\right) - 2 \times \left(\begin{array}{c}
5 \\
3 
\end{array}\right) - 2 \times \left(\begin{array}{c}
4 \\
2 
\end{array}\right) + 2 \times \left(\begin{array}{c}
2 \\
2 
\end{array}\right) + 2 = 98,$$

if $d = 3$. $\square$

5 Conclusion

In this paper we analysed and evaluated the bounded degree sum-of-squares (BSOS) hierarchy of Lasserre, Toh and Yang [12] for a class of bilinear optimization problems, namely pooling problems. We showed that this approach is successful in obtaining the global optimal values for smaller instances, but scalability remains a problem for larger instances. In particular, the number of nonnegative variables and linear constraints grows quickly with the level of the hierarchy. We have showed that it is possible to eliminate some variables and redundant linear constraints in the hierarchy in a systematic way, and this goes some way in improving scalability. More ideas are needed, though, if this approach is to become competitive for medium to larger scale pooling problems.

References


