

A numerical evaluation of the bounded degree sum-of-squares hierarchy of Lasserre, Toh, and Yang on the pooling problem

Marandi, Ahmadreza; Dahl, Joachim; De Klerk, Etienne

DOI

[10.1007/s10479-017-2407-5](https://doi.org/10.1007/s10479-017-2407-5)

Publication date

2018

Document Version

Final published version

Published in

Annals of Operations Research

Citation (APA)

Marandi, A., Dahl, J., & De Klerk, E. (2018). A numerical evaluation of the bounded degree sum-of-squares hierarchy of Lasserre, Toh, and Yang on the pooling problem. *Annals of Operations Research*, 265, 67-92. <https://doi.org/10.1007/s10479-017-2407-5>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

A numerical evaluation of the bounded degree sum-of-squares hierarchy of Lasserre, Toh, and Yang on the pooling problem

Ahmadreza Marandi¹ · Joachim Dahl² ·
Etienne de Klerk^{1,3} 

Published online: 3 February 2017

© The Author(s) 2017. This article is published with open access at Springerlink.com

Abstract The bounded degree sum-of-squares (BSOS) hierarchy of Lasserre et al. (EURO J Comput Optim 1–31, 2015) 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:

Ahmadreza Marandi: The research of this author was supported by the EU Marie Curie Initial Training Network number 316647 (“Mixed Integer Nonlinear Optimization (MINO)”).

✉ Etienne de Klerk
e.deklerk@uvt.nl

Ahmadreza Marandi
a.marandi@uvt.nl

Joachim Dahl
joachim.dahl@mosek.com

¹ Tilburg School of Economics and Management, Tilburg University, Tilburg, The Netherlands

² MOSEK ApS, Copenhagen O, Denmark

³ Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands

$$\begin{aligned} f^* &= \inf_{x \in \mathbb{R}^n} f(x) \\ \text{s.t. } g_j(x) &\geq 0, \quad j = 1, \dots, m, \end{aligned} \quad (1)$$

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

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

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. [Laurent \(2009\)](#). In [2015](#), Lasserre et al. 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 [Lasserre et al. \(2015\)](#) 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. [Karuppiiah and Grossmann \(2006\)](#). 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 \(1978\)](#) described the so-called P-formulation, and afterwards many researchers used this model (e.g. [Adhya et al. 1999](#); [Ben-Tal et al. 1994](#); [Foulds et al. 1992](#)). Also, there are Q-, PQ-, and TP-formulations; in this paper, we use the P- and PQ-formulations and point the reader to the survey by [Gupte et al. \(2017\)](#) where all the formulations are described, as well as the PhD thesis by [Alfaki \(2012\)](#).

One way of getting a lower bound for a pooling problem is using convex relaxation, as done e.g. by [Foulds et al. \(1992\)](#). Similarly, [Adhya et al. \(1999\)](#) introduced a Lagrangian approach to get tighter lower bounds for pooling problems. Also, there are many other papers studying duality ([Ben-Tal et al. 1994](#)), piecewise linear approximation ([Misener et al. 2011](#)), heuristics for finding a good feasible solution ([Alfaki and Haugland 2014](#)), etc. A relatively recent survey on solution techniques is [Misener and Floudas \(2009\)](#).

In a seminal paper in 2000, [Lasserre \(2001\)](#) first introduced a hierarchy of lower bounds for polynomial optimization using SDP relaxations. [Frimannslund et al. \(2010\)](#) 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 Sect. 2. In Sect. 3 the pooling problem is defined, and we review three mathematical models for it, namely the P-, Q- and PQ-formulations. 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 [Lasserre et al. \(2015\)](#). For easy reference, we will use the same notation as in [Lasserre et al. \(2015\)](#).

In what follows \mathbb{N}^k will denote all k -tuples of nonnegative integers, and we define

$$\mathbb{N}_d^k = \left\{ \alpha \in \mathbb{N}^k : \sum_{i=1}^k \alpha_i \leq d \right\}.$$

The space of $n \times n$ symmetric matrices will be denoted by \mathcal{S}^n , and its subset of positive semidefinite matrices by \mathcal{S}_+^n .

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

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & \prod_{j=1}^m g_j(x)^{\alpha_j} (1 - g_j(x))^{\beta_j} \geq 0, \quad \forall (\alpha, \beta) \in \mathbb{N}_d^{2m}. \end{aligned} \quad (2)$$

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

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

The next step is to use the following *positivstellensatz* by Krivine (1964) to remove the quantifier ‘ $\forall x \in F$ ’.

Theorem 1 ([14], see also §3.6.4 in [Laurent 2009](#)) *Assume that $g_j(x) \leq 1$ for all $x \in F$ and $j = 1, \dots, m$, and $\{1, g_1, \dots, g_m\}$ generates the ring of polynomials. If a polynomial g is positive on F then*

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

for finitely many $\lambda_{\alpha\beta} > 0$.

Defining

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

we arrive at the following sequence of lower bounds (indexed by d) for problem (1):

$$f^* \geq \sup_t \left\{ t : f(x) - t = \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) \right\}. \quad (3)$$

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 $d \rightarrow \infty$, by Krivine’s *positivstellensatz*. This hierarchy of LP bounds was introduced by [Lasserre \(2005\)](#).

A subsequent idea, from [Lasserre et al. \(2015\)](#) was to strengthen the LP bounds by enlarging its feasible set as follow: If we fix $\kappa \in \mathbb{N}$, and denote by $\Sigma[x]_\kappa$ the space of sums of squares polynomials of degree at most 2κ , then we may define the bounds

$$q_d^\kappa := \sup_{t, \lambda} \left\{ t : f(x) - t - \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) \in \Sigma[x]_\kappa \right\}.$$

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

For fixed κ and for each d , one has

$$\begin{aligned} q_d^\kappa &= \sup_{t, \lambda, Q} t \\ \text{s.t. } & f(x) - \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) - t = \text{trace} \left(Q v_\kappa(x) v_\kappa(x)^T \right), \\ & Q \in S_+^{s(\kappa)}, \quad \lambda \geq 0, \end{aligned} \quad (4)$$

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

Letting $\tau = \max\{\deg(f), 2\kappa, d \max_j \deg(g_j)\}$, 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 (4), 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 τ 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 [Lofberg and Parrilo \(2004\)](#). It was also used for the numerical BSOS hierarchy calculations in [Lasserre et al. \(2015\)](#), with a set of $s(\tau)$ randomly generated points in \mathbb{R}^n .

We will instead use the points

$$\Delta(n, \tau) = \left\{ x \in \mathbb{R}^n \mid \tau x \in \mathbb{N}^n, \sum_{i=1}^n x_i \leq 1 \right\}.$$

Thus we obtain the following SDP reformulation of (4):

$$\begin{aligned} q_d^\kappa &= \sup_{t, \lambda, Q} t \\ & f(x) - \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) - t = \text{trace} \left(Q v_\kappa(x) v_\kappa(x)^T \right), \quad \forall x \in \Delta(n, \tau) \\ & Q \in S_+^{s(\kappa)}, \quad \lambda \geq 0. \end{aligned} \quad (5)$$

The following theorem, proved in [Lasserre et al. \(2015\)](#), gives some information on feasibility and duality issues for the BSOS relaxation.

Theorem 2 (Lasserre et al. 2015) *If problem (1) is Slater feasible, then so is the dual SDP problem of (5). Thus (by the conic duality theorem), if the SDP problem (5) has a feasible solution, it has an optimal solution as well.*

Note that problem (5) may be infeasible for given d and κ . One only knows that it will be feasible, and therefore q_d^κ will be defined, for sufficiently large d .

Remark 1 Assume that at the d -th level of the hierarchy we have $q_d^\kappa = f^*$, i.e. finite convergence of the BSOS hierarchy, then

$$f(x) - f^* = \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) + v_\kappa(x)^T Q v_\kappa(x) \quad \forall x \in \mathbb{R}. \quad (6)$$

Let $x^* \in F$ be an optimal solution ($f(x^*) = f^*$), then it is clear from (6) that

$$0 = \sum_{(\alpha, \beta) \in \mathbb{N}_d^{2m}} \lambda_{\alpha\beta} h_{\alpha\beta}(x^*) + v_\kappa(x^*)^T Q v_\kappa(x^*) \quad \forall x \in \mathbb{R},$$

and due to the fact that Q is positive semidefinite, then

$$\lambda_{\alpha\beta} h_{\alpha\beta}(x^*) = 0 \quad \forall (\alpha, \beta) \in \mathbb{N}_d^{2m}. \quad (7)$$

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-, Q- and PQ-formulations of the pooling problem

In this section, we describe the P-, Q- and PQ-formulations of the pooling problem. The notation we are using is the same as in Gupte et al. (2017). 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 Fig. 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 $i \in \mathcal{N}$. 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 \mathcal{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:

$$\min_{y, p} \sum_{(i, j) \in \mathcal{A}} c_{ij} y_{ij} \quad (8)$$

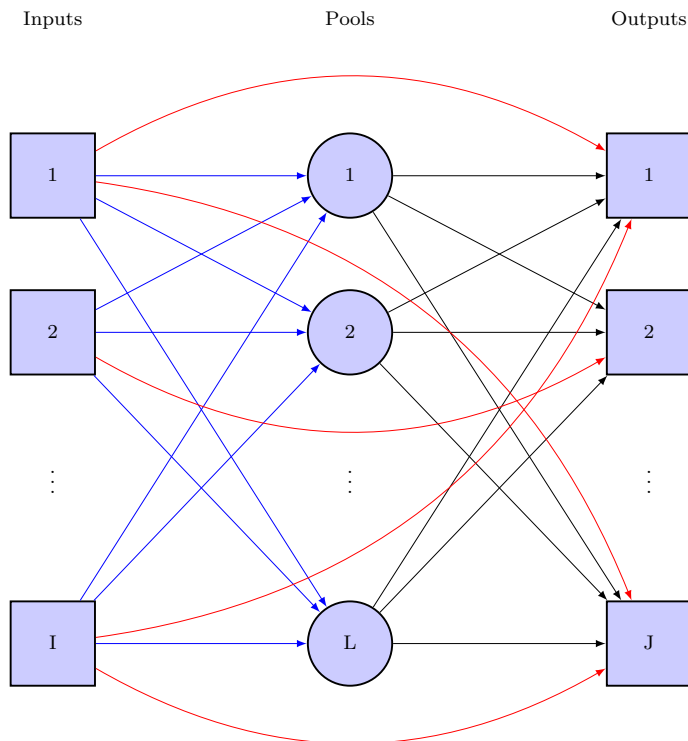


Fig. 1 An example of a standard pooling problem with I inputs, L pools and J outputs

s.t.

$$\sum_{\substack{i \in \mathcal{I}: \\ (i,l) \in \mathcal{A}}} y_{il} = \sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj}, \quad l \in \mathcal{L} \quad (9)$$

$$\sum_{\substack{j \in \mathcal{L} \cup \mathcal{J}: \\ (i,j) \in \mathcal{A}}} y_{ij} \leq C_i, \quad i \in \mathcal{I} \quad (10)$$

$$\sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj} \leq C_l, \quad l \in \mathcal{L} \quad (11)$$

$$\sum_{\substack{i \in \mathcal{I} \cup \mathcal{L}: \\ (i,j) \in \mathcal{A}}} y_{ij} \leq C_j, \quad j \in \mathcal{J} \quad (12)$$

$$0 \leq y_{ij} \leq u_{ij}, \quad (i,j) \in \mathcal{A} \quad (13)$$

$$\sum_{\substack{i \in \mathcal{I}: \\ (i,l) \in \mathcal{A}}} \lambda_{ik} y_{il} = p_{lk} \sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj}, \quad l \in \mathcal{L}, k \in \mathcal{K} \quad (14)$$

$$\sum_{\substack{i \in \mathcal{I}: \\ (i,j) \in \mathcal{A}}} \lambda_{ik} y_{ij} + \sum_{\substack{l \in \mathcal{L}: \\ (l,j) \in \mathcal{A}}} p_{lk} y_{lj} \leq \mu_{jk}^{\max} \sum_{\substack{i \in \mathcal{I} \cup \mathcal{L}: \\ (i,j) \in \mathcal{A}}} y_{ij}, \quad j \in \mathcal{J}, k \in \mathcal{K} \quad (15)$$

$$\sum_{\substack{i \in \mathcal{I}: \\ (i,j) \in \mathcal{A}}} \lambda_{ik} y_{ij} + \sum_{\substack{l \in \mathcal{L}: \\ (l,j) \in \mathcal{A}}} p_{lk} y_{lj} \geq \mu_{jk}^{\min} \sum_{\substack{i \in \mathcal{I} \cup \mathcal{L}: \\ (i,j) \in \mathcal{A}}} y_{ij}, \quad j \in \mathcal{J}, k \in \mathcal{K} \quad (16)$$

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

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

For a general pooling problem, the aforementioned model is called the P-formulation. Consider a pool $l \in \mathcal{L}$ and the arc incident to it from input $i \in \mathcal{I}$. Let us denote by q_{il} the ratio between the flow in this arc and the total incoming flow to this pool. So, $y_{il} = q_{il} \sum_{j \in \mathcal{J}} y_{lj}$, and $p_{lk} = \sum_{i \in \mathcal{I}} \lambda_{ik} q_{il}$ for any $k \in \mathcal{K}$. Applying these to the P-formulation yields the following problem called the Q-formulation:

$$\begin{aligned} & \min_{y,p} \sum_{(i,j) \in \mathcal{A}} c_{ij} y_{ij} \\ & \text{s.t.} \\ & (10) - (13) \\ & \sum_{\substack{i \in \mathcal{I}: \\ (i,l) \in \mathcal{A}}} q_{il} = 1, \quad q_{il} \geq 0, \quad l \in \mathcal{L}, i \in \mathcal{I}, (i,l) \in \mathcal{A} \\ & y_{il} = q_{il} \sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj}, \quad l \in \mathcal{L}, i \in \mathcal{I}, (i,l) \in \mathcal{A} \\ & \sum_{\substack{i \in \mathcal{I}: \\ (i,j) \in \mathcal{A}}} \lambda_{ik} y_{ij} + \sum_{\substack{l \in \mathcal{L}: \\ (l,j) \in \mathcal{A}}} \sum_{\substack{i \in \mathcal{I}: \\ (i,l) \in \mathcal{A}}} \lambda_{ik} q_{il} y_{lj} \leq \mu_{jk}^{\max} \sum_{\substack{i \in \mathcal{I} \cup \mathcal{L}: \\ (i,j) \in \mathcal{A}}} y_{ij}, \quad j \in \mathcal{J}, k \in \mathcal{K} \\ & \sum_{\substack{i \in \mathcal{I}: \\ (i,j) \in \mathcal{A}}} \lambda_{ik} y_{ij} + \sum_{\substack{l \in \mathcal{L}: \\ (l,j) \in \mathcal{A}}} \sum_{\substack{i \in \mathcal{I}: \\ (i,l) \in \mathcal{A}}} \lambda_{ik} q_{il} y_{lj} \geq \mu_{jk}^{\min} \sum_{\substack{i \in \mathcal{I} \cup \mathcal{L}: \\ (i,j) \in \mathcal{A}}} y_{ij}, \quad j \in \mathcal{J}, k \in \mathcal{K}. \end{aligned} \quad (17)$$

Adding two sets of redundant constraints

$$y_{lj} \sum_{\substack{i \in \mathcal{I}: \\ (i,l) \in \mathcal{A}}} q_{il} = y_{lj}, \quad l \in \mathcal{L}, j \in \mathcal{J}, (l,j) \in \mathcal{A}, \quad (18a)$$

$$q_{il} \sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj} \leq C_l q_{il}, \quad i \in \mathcal{I}, l \in \mathcal{L}, (i,l) \in \mathcal{A}, \quad (18b)$$

gives an equivalent problem, called the PQ-formulation. It is clear that all formulations are nonconvex quadratic optimization problems which are not easy to solve (Haugland 2016).

3.1 McCormick relaxation and the pooling problem

Assume that x and y are variables with given lower and upper bounds

$$\ell_x \leq x \leq u_x, \quad \ell_y \leq y \leq u_y.$$

Then, the following inequalities are implied when $\chi = xy$:

$$\chi \geq \ell_x y + \ell_y x - \ell_x \ell_y, \quad (19a)$$

$$\chi \geq u_x y + u_y x - u_x u_y, \quad (19b)$$

$$\chi \leq \ell_x y + u_y x - \ell_x u_y, \quad (19c)$$

$$\chi \leq u_x y + \ell_y x - u_x \ell_y. \quad (19d)$$

It is known that the convex hull of

$$\mathcal{B} := \{(x, y, \chi) \mid \chi = xy, \ell_x \leq x \leq u_x, \ell_y \leq y \leq u_y\},$$

which is called the McCormick relaxation (Gupte et al. 2017), is exactly the set of (x, y, χ) that satisfies the inequalities (19).

In the pooling problem, the following lower and upper bounds on the variables are implied:

$$\begin{aligned} m_\lambda &:= \min_{i \in \mathcal{I}} \lambda_{ik} \leq p_{lk} \leq M_\lambda := \max_{i \in \mathcal{I}} \lambda_{ik}, & \forall l \in \mathcal{L}, k \in \mathcal{K}, \\ 0 &\leq y_{lj} \leq \min\{C_j, u_{lj}\}, & \forall j \in \mathcal{J}, l \in \mathcal{L}. \end{aligned}$$

So, one can get a lower bound by using the McCormick relaxation of each bilinear term in the P- or PQ-formulations.

The redundant constraints (18) guarantee that the relaxation obtained by using the McCormick relaxation for the PQ-formulation is stronger than that for the P-formulation (Gupte et al. 2017).

In this paper, we are going to use the BSOS hierarchy to find a sequence of lower bounds that converges to the optimal value of the pooling problem. First we analyze the P-formulation and in Sect. 4.3 we compare the results by using the PQ-formulation.

3.2 Solving pooling problems with the BSOS hierarchy

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 dealing with 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. Another way of doing so is eliminating the equality constraints (9) and (14), if possible.

3.2.1 Eliminating equality constraints

Let $l \in \mathcal{L}$. We assume without loss of generality that the first t inputs feed the pool l . Therefore, equality constraints (9) and (14) can be written as follows:

$$\underbrace{\begin{pmatrix} 1 & 1 & \dots & 1 \\ \lambda_{11} & \lambda_{21} & \dots & \lambda_{t1} \\ \lambda_{12} & \lambda_{22} & \dots & \lambda_{t2} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{1K} & \lambda_{2K} & \dots & \lambda_{tK} \end{pmatrix}}_{A:=} \begin{bmatrix} y_{1l} \\ y_{2l} \\ \vdots \\ y_{tl} \end{bmatrix} = \sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj} \begin{bmatrix} 1 \\ p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix} \quad l \in \mathcal{L}. \quad (20)$$

Let $\text{rank}(A) = r$. Applying a singular value decomposition (see, e.g. [Boyd and Vandenberghe 2004](#), Appendix A.5.4), there are matrices $U = [U_1, U_2] \in \mathbb{R}^{(K+1) \times (K+1)}$, $V = [V_1, V_2] \in \mathbb{R}^{t \times t}$, $\Sigma \in \mathbb{R}^{r \times r}$ such that

$$A = U \begin{bmatrix} \Sigma & 0_1 \\ 0_2 & 0_3 \end{bmatrix} V^T, U^T U = I, V^T V = I, \\ \Sigma = \text{diag}(\sigma_1, \dots, \sigma_r), \sigma_1 > \sigma_2 > \dots > \sigma_r > 0,$$

where $V_1 \in \mathbb{R}^{t \times r}$, $V_2 \in \mathbb{R}^{t \times (t-r)}$, $U_1 \in \mathbb{R}^{(K+1) \times r}$, $U_2 \in \mathbb{R}^{(K+1) \times (K+1-r)}$, $0_1 \in \mathbb{R}^{r \times (t-r)}$, $0_2 \in \mathbb{R}^{(K+1-r) \times r}$, $0_3 \in \mathbb{R}^{(K+1-r) \times (t-r)}$. Therefore, (20) can be written as

$$V_1^T \begin{bmatrix} y_{1l} \\ y_{2l} \\ \vdots \\ y_{tl} \end{bmatrix} = \left(\sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj} \right) \Sigma^{-1} U_1^T \begin{bmatrix} 1 \\ p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix} \quad l \in \mathcal{L}, \quad (21)$$

$$0 = \left(\sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj} \right) U_2^T \begin{bmatrix} 1 \\ p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix} \quad l \in \mathcal{L}. \quad (22)$$

The fact that $V^T V = I$, implies that all columns in V , and hence in V_1 are linearly independent. Therefore, taking the QR decomposition of V_1^T , i.e., $V_1^T = Q[R_1, R_2]$, where $R_1 \in \mathbb{R}^{r \times r}$ is upper triangular and invertible, $R_2 \in \mathbb{R}^{r \times (t-r)}$, and $Q \in \mathbb{R}^{r \times r}$ is orthonormal ($Q^T Q = Q Q^T = I$), (21) is equivalent to

$$\begin{bmatrix} y_{1l} \\ y_{2l} \\ \vdots \\ y_{tl} \end{bmatrix} = R_1^{-1} \left(\left(\sum_{\substack{j \in \mathcal{J}: \\ (l,j) \in \mathcal{A}}} y_{lj} \right) Q^T \Sigma^{-1} U_1^T \begin{bmatrix} 1 \\ p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix} - R_2 \begin{bmatrix} y_{(r+1)l} \\ y_{2l} \\ \vdots \\ y_{tl} \end{bmatrix} \right), \quad l \in \mathcal{L}. \quad (23)$$

Note that one may use Eq. (23) to eliminate the variables $y_{1l}, y_{2l}, \dots, y_{tl}$.

Concerning (22), if, for a feasible solution of (8)–(16), $\sum_{j \in \mathcal{J}} y_{lj} = 0$, then any other choice of the values $p_{lk}, k = 1, \dots, K$, gives another feasible solution. So in this case one may choose values that satisfy

$$0 = U_2^T \begin{bmatrix} 1 \\ p_{l1} \\ p_{l2} \\ \vdots \\ p_{lK} \end{bmatrix}, \quad (24)$$

which is a system of K variables and $K - r + 1$ linearly independent equalities with $r \geq 1$. Conversely, a feasible solution with the property $\sum_{j \in \mathcal{J}} y_{lj} \neq 0$ definitely satisfies (24). So, instead of (22), we may solve (24), which may be done using the QR decomposition. Thus we may write p_{lr}, \dots, p_{lK} as a linear function of $p_{l1}, \dots, p_{l(r-1)}$, and subsequently eliminate p_{lr}, \dots, p_{lK} as well.

Table 1 Details for some well-known pooling problem instances

	Optimal value	PQ-linear relaxation value (Alfaki and Haugland 2013; Dey and Gupte 2015)	I	J	L	K	# var.	# const.
Haverly1	− 400.00	−500.00	3	2	1	1	5	11
Haverly2	−600.00	−1000.00	3	2	1	1	5	11
Haverly3	−750.00	−800.00	3	2	1	1	5	11
Ben-Tal4	−450.00	−550.00	4	2	1	1	6	13
Ben-Tal5	−3,500.00	−3500.00	5	5	3	2	29	54
DeyGupte4	−1.00	[−4, −3]	2	4	2	2	10	52
Foulds2	−1100.00	−1100.00	6	4	2	1	18	38
Foulds3	−8.00	−8.00	11	16	8	1	152	219
Foulds4	−8.00	−8.00	11	16	8	1	152	219
Adhya1	−549.80	−840.27	5	4	2	4	11	41
Adhya2	−549.80	−574.78	5	4	2	6	11	53
Adhya3	−561.05	−574.78	8	4	3	6	17	66
Adhya4	−877.6.	−961.93	8	5	2	4	16	51
RT2	−4391.83	−6034.87	3	3	2	8	14	67
sppA0	Unknown*	−37,772.75	20	15	10	24	161	816

* The optimal value for this instance is not known exactly, but known to lie in the interval $[-36233.40, -35812.33]$

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

Remark 3 The interpretation of eliminating equality constraints is as follows when the matrix A is full rank ($\text{rank}(A) = \min\{K + 1, t\}$): For pools with exactly $K + 1$ entering arcs, the entering flow values are given by the total leaving flow and the concentrations in the pool. With more than $K + 1$ arcs, say t , $t - K - 1$ flow values can be chosen freely and the remaining $K + 1$ determined by total leaving flow and concentrations. When $t < K + 1$, a basis of t concentration values define the $K + 1 - t$ remaining ones.

3.2.2 First numerical results

In this section, we study convergence of the BSOS hierarchy of lower bounds q_d^1 ($d = 1, 2, \dots$) for pooling problems ($\kappa = 1$). 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 \mathbb{N}_d^{2m} , namely $\binom{2m+d}{d} + 1$.

Table 1 gives some information of the standard pooling problem instances we use in this paper. The GAMS files of the pooling problem instances that we use in this paper, except DeyGupte4, can be found on the website <http://www.ii.uib.no/~mohammeda/spooling/>. DeyGupte4 is constructed in this paper (Appendix) by using the results of Dey and Gupte (2015). In Table 1, we recall in column “PQ-linear relaxation value” the lower bound proposed in Alfaki and Haugland (2013) of each instance. This lower bound is the optimal

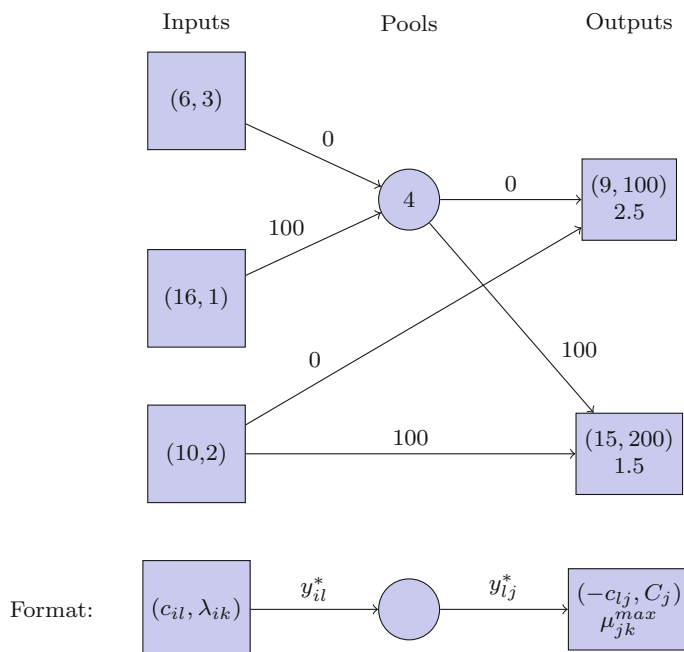


Fig. 2 Optimal solution for Haverly1

value of the PQ-formulation after applying McCormick relaxation for each bilinear term. It is proved in [Dey and Gupte \(2015\)](#) that any optimal value of a piecewise linear approximation of the PQ-formulation (for a precise definition see Appendix) for DeyGupte4, has optimal value in $[-4, -3]$. Also, columns “# var.” and “# const.” contain the number of variables and constraints in the P-formulation after eliminating equality constraints, respectively.

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 Fig. 2, and the optimal value is -400 ([Adhya et al. 1999](#)). The optimal flow from node i to node j is denoted by y_{ij}^* in Fig. 2.

This instance has three inputs (denoted by 1, 2, 3), one pool (denoted by 4), two outputs (denoted by 5, 6), and one specification. The mathematical model for this instance is as follows:

$$\begin{aligned}
 \min \quad & 6y_{14} + 16y_{24} + 10[y_{35} + y_{36}] - 9[y_{45} + y_{35}] - 15[y_{46} + y_{36}] \\
 \text{s.t.} \quad & y_{14} + y_{24} = y_{45} + y_{46}, \\
 & 0 \leq y_{45} + y_{35} \leq 100, \quad (25a) \\
 & 0 \leq y_{46} + y_{36} \leq 200, \quad (25b) \\
 & 3y_{14} + y_{24} = p_1[y_{45} + y_{46}], \\
 & 2y_{35} + p_1y_{45} \leq 2.5[y_{35} + y_{45}], \quad (25c) \\
 & 3y_{36} + p_1y_{46} \leq 1.5[y_{36} + y_{46}], \\
 & y_{ij} \geq 0, \quad p_1 \geq 0. \quad (25d)
 \end{aligned}$$

So, we can use the elimination method described in the previous section, which implies that $y_{14} = \frac{1}{2}(y_{45} + y_{46})(p_1 - 1)$, $y_{24} = \frac{1}{2}(y_{45} + y_{46})(3 - p_1)$. Therefore, the reformulated model

of this instance using scaling $x_1 := \frac{p_1}{3}$, $x_2 := \frac{y_{45}}{200}$, $x_3 := \frac{y_{46}}{200}$, $x_4 := \frac{y_{35}}{200}$, $x_5 := \frac{y_{36}}{200}$, is

$$\begin{aligned} \min \quad & -200x_2(15x_1 - 12) - 200x_3(15x_1 - 6) + 200x_4 - 1000x_5 \\ \text{s.t.} \quad & 1 \geq -\frac{3}{4}(x_1 - 1)(x_2 + x_3) \geq 0 \end{aligned} \quad (26a)$$

$$1 \geq \frac{1}{4}(3x_1 - 1)(x_2 + x_3) \geq 0 \quad (26b)$$

$$1 \geq 1 - 2(x_2 + x_4) \geq 0 \quad (26c)$$

$$1 \geq 1 - (x_3 + x_5) \geq 0 \quad (26d)$$

$$1 \geq \frac{1}{2}(x_4 + x_2) - \frac{2}{5}x_4 - \frac{3}{5}x_1x_2 \geq 0 \quad (26e)$$

$$\begin{aligned} 1 &\geq \frac{1}{2}(x_5 + x_3) - \frac{2}{3}x_5 - x_1x_3 \geq 0 \\ 1 &\geq x_i \geq 0, \quad i = 1, \dots, 5, \end{aligned} \quad (26f)$$

where the leftmost inequalities are redundant, (26a) and (26b) are from the sign constraints after the elimination, (26c), (26d), (26e), and (26f) are from (25a), (25b), (25c) and (25d), respectively.

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 ‘ ≤ 1 ’ conditions hold with strict inequality on the feasible set. Thus, we define $g_1(x) = -0.9 \cdot \frac{3}{4}(x_1 - 1)(x_2 + x_3)$, etc.

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

The results for *Haverly1* and the other pooling problem instances are listed in Table 2. All computations in this paper were carried out with MOSEK 8 on an Intel i7-4790 3.60 GHz Windows computer with 16 GB of RAM. “Numerical Prob.” and “ \approx ” in the tables mean the solver reported a numerical problem, and only obtained an approximate optimal value, respectively. In all the tables from now on, columns “# lin. var.”, “size of SDP var.” and “# const.” present the number of linear variables, the size of the semidefinite matrix variable and the number of constraints in the hierarchy.

As it is clear, in order to compute q_d^K 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 (5). 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.

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

	Iteration	Time	Solution	# Lin. var.	Size of SDP var.	# Const.
Haverly1	d = 1	0.01s	−600.00	23	6	21
	d = 2	0.03s	−417.20	276	6	126
	d = 3	0.47s	−400.00	2300	6	462
Haverly2	d = 1	0.01s	−1200	23	6	21
	d = 2	0.03s	−601.67	276	6	126
	d = 3	0.39s	−600.00	2300	6	462
Haverly3	d = 1	0.02s	−875.00	23	6	21
	d = 2	0.03s	−750.00	276	6	126
Ben-Tal4	d = 1	0.02s	−650.00	27	7	28
	d = 2	0.03s	−467.20	378	7	210
	d = 3	1.44s	−450.00	3654	7	924
Ben-Tal5	d = 1	0.06s	−3500.00	109	30	465
DeyGupte4	d = 1	0.02s	−4.00	105	11	66
	d = 2	4.60s	−3.86	5565	11	1001
	d = 3	–	–	198,485	11	8008
Foulds2	d = 1	0.01s	−1200.00	77	19	190
	d = 2	109.20s	−1191.30	3003	19	7315
	d = 3	–	–	79,079	19	134,596
Foulds3	d = 1	90.84s	−8.00	439	153	11,781
Foulds4	d = 1	92.85s	−8.00	439	153	11,781
Adhya1	d = 1	0.02s	−999.32	83	12	78
	d = 2	4.26s	≈−723.94	3486	12	1365
	d = 3	–	–	98,770	12	12,376
Adhya2	d = 1	0.02s	−798.29	107	12	78
	d = 2	11.51s	≈−576.82	5778	12	1365
	d = 3	–	–	209,934	12	12,376
Adhya3	d = 1	0.03s	−882.84	133	18	171
	d = 2	135.39s	≈−802.89	8911	18	5985
	d = 3	–	–	400,995	18	100,947
Adhya4	d = 1	0.02s	−1055.00	103	17	153
	d = 2	52.59s	≈−1035.00	5356	17	4845
	d = 3	–	–	187,460	17	74,613
RT2	d = 1	0.02s	−45,420.50	135	15	120
	d = 2	30.84s	−36,542.19	9180	15	3060
	d = 3	–	–	419,220	15	38,760
sppA0	d = 1	273.00s	−47,675.00	1633	162	13,203
	d = 2	–	–	1,334,161	162	29,772,765

Bold faced entries correspond to the optimal solution

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.

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 (5) to compute q_d^κ , we may compute the following (weaker) bound more efficiently:

$$\begin{aligned} \hat{q}_d^\kappa &:= \sup_{t, \lambda, Q} t \\ f(x) - \sum_{\substack{(\alpha, \beta) \in \mathbb{N}_d^{2m} \\ \alpha \neq 0}} \lambda_{\alpha\beta} h_{\alpha\beta}(x) - t &= \text{trace} \left(Q v_\kappa(x) v_\kappa(x)^T \right), \quad \forall x \in \Delta(n, \tau), \\ Q &\in S_+^{s(\kappa)}, \quad \lambda \geq 0. \end{aligned} \quad (27)$$

The advantage of (27) is that it has $\binom{m+d}{d}$ fewer nonnegative variables than (5). We emphasize that problem (27) is not equivalent to (5), i.e. the lower bounds q_d^κ and \hat{q}_d^κ are not equal in general — the bound \hat{q}_d^κ is weaker, and may be strictly weaker.

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

Theorem 3 *If, for given d and κ , q_d^κ and \hat{q}_d^κ are both finite, then $\hat{q}_d^\kappa \leq q_d^\kappa$. Moreover, if the sequence q_d^κ ($d = 1, 2, \dots$) from (5) converges finitely to f^* , then so does \hat{q}_d^κ ($d = 1, 2, \dots$) from (27). More precisely, if $q_{d^*}^\kappa = f^*$ for some $d^* \in \mathbb{N}$, then $\hat{q}_{d^*}^\kappa = f^*$.*

It is important to note that finite convergence of the sequence q_d^κ ($d = 1, 2, \dots$) is not guaranteed in general. Sufficient conditions for finite convergence are described in Lasserre et al. (2015).

The numerical results for using (27) for the pooling problem instances is demonstrated in Table 3. The “rel. time” column from this table onward gives the solution time for each level of the hierarchy as a ratio of that in Table 2, which shows that there is a significant reduction in computational times when compared to Table 2.

4.2 Reduction in the number of constraints

From now on we fix $\kappa = 1$ and $v_1(x) = (1, x_1, \dots, x_n)$. 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.

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

$$\text{svec}(X) = \left[X_{11}, \sqrt{2}X_{12}, X_{22}, \dots, \sqrt{2}X_{n(n+1)}, X_{(n+1)(n+1)} \right], \quad \forall X \in S^{n+1}.$$

Table 3 Results for computing the lower bounds \hat{q}_d^1 for pooling problem instances using (27)

	Iteration	Rel. time	Solution	# Lin. var.	Size of SDP var.	# Const.
Haverly1	d = 1	1	−600.00	11	6	21
	d = 2	1	−417.20	198	6	126
	d = 3	0.60	−400.00	1936	6	462
Haverly2	d = 1	1	−1200.00	11	6	21
	d = 2	1	−601.67	198	6	126
	d = 3	0.79	−600.00	1936	6	462
Haverly3	d = 1	1	−875.00	11	6	21
	d = 2	1	−750.00	198	6	126
Ben-Tal4	d = 1	1	−650.00	14	7	28
	d = 2	1	−467.20	274	7	210
	d = 3	0.73	−450.00	3095	7	924
Ben-Tal5	d = 1	0.83	−3500.00	55	30	465
DeyGupte4	d = 1	1	−4.00	53	11	66
	d = 2	0.62	−3.86	4135	11	1001
	d = 3	–	–	172,250	11	8008
Foulds2	d = 1	1	−1200.00	39	19	190
	d = 2	0.85	−1191.29	2224	19	7315
	d = 3	–	–	66,419	19	134,596
Foulds3	d = 1	0.94	−8.00	220	153	11,781
Foulds4	d = 1	0.92	−8.00	220	153	11,781
Adhya1	d = 1	1	−999.32	42	12	78
	d = 2	0.95	≈ −723.94	2583	12	1365
	d = 3	–	–	85,526	12	12,376
Adhya2	d = 1	1	−798.29	54	12	78
	d = 2	0.55	≈ −576.82	4293	12	1365
	d = 3	–	–	182,214	12	12,376
Adhya3	d = 1	1	−882.84	67	18	171
	d = 2	0.69	≈ −802.82	6634	18	5985
	d = 2	–	–	348,601	18	100,947
Adhya4	d = 1	1	−1055.00	52	17	153
	d = 2	0.71	≈ −1035.10	3979	17	4845
	d = 3	–	–	162,657	17	74,613
RT2	d = 1	1	−45,420.48	68	15	120
	d = 2	0.65	−36,542.06	6836	15	3060
	d = 3	–	–	419,220	15	38,760
sppA0	d = 1	0.99	−47,6750.00	817	162	13,203
	d = 2	–	–	1,000,008	162	29,772,765

Bold faced entries correspond to the optimal solution

It will also be convenient to number the elements of $\Delta(n, \tau)$ as x^1, \dots, x^L where $L = s(\tau)$. Finally, we will use the notation $|\beta| = \sum_i \beta_i$.

So, for $d \geq 1$ and $\kappa = 1$ we may write the linear equality constraints in (5) as $H_d y_d = b_d$, where

$$H_d = \begin{bmatrix} 1 & (h_{\alpha\beta}(x^1))_{(\alpha,\beta) \in \mathbb{N}_d^{2m}} & \text{svec}(v_1(x^1)v_1(x^1)^T) \\ \vdots & \vdots & \vdots \\ 1 & (h_{\alpha\beta}(x^L))_{(\alpha,\beta) \in \mathbb{N}_d^{2m}} & \text{svec}(v_1(x^L)v_1(x^L)^T) \end{bmatrix},$$

$$b_d = \begin{bmatrix} f(x^1) \\ \vdots \\ f(x^L) \end{bmatrix}, \quad y_d = \begin{bmatrix} t \\ (\lambda_{\alpha\beta})_{(\alpha,\beta) \in \mathbb{N}_d^{2m}} \\ \text{svec}(Q)^T \end{bmatrix},$$

and $L = \binom{n+2d}{2d}$. It is 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 $g_j(x)$, $j = 1, \dots, m$, 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 (5) are linearly independent.*

Proof Fix $d = 1$, which implies $\tau = 2$ and $L = \binom{n+2}{2}$ in (5). Then,

$$H_1 = \begin{bmatrix} 1 & g_1(x^1) & \dots & g_m(x^1) & 1 - g_1(x^1) & \dots & 1 - g_m(x^1) & \text{svec}(v_1(x^1)v_1(x^1)^T) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & g_1(x^L) & \dots & g_m(x^L) & 1 - g_1(x^L) & \dots & 1 - g_m(x^L) & \text{svec}(v_1(x^L)v_1(x^L)^T) \end{bmatrix},$$

and,

$$b_1 = \begin{bmatrix} f(x^1) \\ \vdots \\ f(x^L) \end{bmatrix}, \quad y_1 = \begin{bmatrix} t \\ (\lambda_{\alpha\beta})_{(\alpha,\beta) \in \mathbb{N}_1^{2m}} \\ \text{svec}(Q)^T \end{bmatrix},$$

for $x^1, \dots, x^L \in \Delta(n, 2)$, defined in (5). To show that all of the rows in H_1 are linearly independent, we prove that the submatrix

$$V_n^1 = \begin{bmatrix} \text{svec}(v_1(x^1)v_1(x^1)^T) \\ \vdots \\ \text{svec}(v_1(x^L)v_1(x^L)^T) \end{bmatrix} \in \mathbb{R}^{\binom{n+2}{2} \times \binom{n+2}{2}} = \mathbb{R}^{L \times L},$$

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 the rank of the matrix $V_1^1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & \frac{\sqrt{2}}{2} & \frac{1}{4} \\ 1 & \sqrt{2} & 1 \end{bmatrix}$, is 3, which

means that V_1^1 is a full rank matrix.

Now, suppose that V_n^1 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^γ , 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^α , where $(\alpha, 0) \in \mathbb{N}_2^{n+1}$, after that the columns related to x_{n+1} , x_{n+1}^2 , $x_{n+1}x_i$, $i = 1, \dots, n$. 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:

$$V_{n+1} = \begin{matrix} & \begin{matrix} x^\alpha \\ (\alpha, 0) \in \mathbb{N}_2^{n+1} \\ V_n^1 \\ a_1 \\ a_2 \\ a_3 \end{matrix} & \begin{matrix} x_{n+1} \\ 0_{L \times 1} \\ \frac{\sqrt{2}}{2} \mathbb{1}_{n \times 1} \\ \frac{\sqrt{2}}{2} \\ \sqrt{2} \end{matrix} & \begin{matrix} x_{n+1}^2 \\ 0_{L \times 1} \\ \frac{1}{4} \mathbb{1}_{n \times 1} \\ \frac{1}{4} \\ 1 \end{matrix} & \begin{matrix} x_{n+1}x_i \\ i = 1, \dots, n \\ 0_{L \times n} \\ \frac{\sqrt{2}}{4} I_n \\ 0_{1 \times n} \\ 0_{1 \times n} \end{matrix} \end{matrix} \quad \text{Case I} \quad \text{Case II}_{ii} \quad \text{Case II}_i \quad \text{Case III}$$

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 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. \square

In Theorem 4, we prove that if $d = 1$, then all of the constraints in (5) are linearly independent. In the next theorem, we prove that for $d \geq 2$, if we rewrite H_d as $[\tilde{H}_d, V_n^d]$, where

$$V_n^d = \begin{bmatrix} \text{svec}(v_1(x^1)v_1(x^1)^T) \\ \vdots \\ \text{svec}(v_1(x^L)v_1(x^L)^T) \end{bmatrix} \in \mathbb{R}^{L \times L},$$

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

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

Proof The ‘if’ part is trivial.

To prove the ‘only if’ part, without loss of generality we assume that x^p , $p = 1, \dots, t$ 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 \tilde{H}_d , for $d \geq 2$. This means that V_n^d is a submatrix of \tilde{H}_d , which implies that the first t rows in \tilde{H}_d are linearly independent. \square

After elimination of the equality constraints in pooling problem (8), 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 \tilde{H}_d .

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

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 $L = \binom{n+2d}{2d}$, $g(x) = (g_1(x), \dots, g_m(x))$, for each $\alpha \in \mathbb{N}_d^m$, $g(x^p)^\alpha = \prod_{j=1}^m g_j(x)^{\alpha_j}$, and $(g(x^p)^\alpha)_{\alpha \in \mathbb{N}_d^m} \in \mathbb{R}^{1 \times \binom{m+d}{d}}$, $p = 1, \dots, L$.

Because the columns of \hat{H}_d are a subset of the columns of \bar{H}_d , so $\text{Range}(\hat{H}_d) \subseteq \text{Range}(\bar{H}_d)$. To prove the other containment, we show that all columns of \bar{H}_d are linear combinations of \hat{H}_d 's columns. Each column of \bar{H}_d is related to a function $h_{\alpha\beta}(x)$ for some $(\alpha, \beta) \in \mathbb{N}_d^{2m}$. If $\beta = 0$ for a column of \bar{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 $\gamma_i \in \mathbb{N}_{|\beta|}^m$, $a_i \in \mathbb{R}$, $i = 1, \dots, w$, 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 \hat{H}_d , for each $i = 1, \dots, w$. This means that any column of \bar{H}_d is a linear combination of the columns in \hat{H}_d . \square

By Theorem 6, to find the number of linearly independent constraints in (5), we only need to check the columns related to $h_{\alpha\beta}(x)$ with $\beta = 0$.

It is clear that the results in this paper, except Theorem 4, can be modified for the LP bounds (3). In fact, Theorems 5 and 6 are true in each level, even $d = 1$. In Table 4, the results of solving the pooling problem instances in Table 1 are shown after eliminating the linearly dependent constraints using (3) and \hat{q}_d^1 in (27). Note that the computational times at the $d = 2$ and $d = 3$ levels are greatly reduced when compared to the times in Table 3. For some instances because of the large number of constraints in the last level of the hierarchy, we could not find the number of linearly independent constraints and we put “-” as in Table 4. Also in this table we show how much stronger the BSOS hierarchy is compared to the LP bounds (3) after reducing the number of variables and deleting the linear dependent constraints. As one can see, the main difference between the BSOS hierarchy and (3) is in the first level, in which the number of independent constraints in (3) is much smaller than the BSOS hierarchy. If there is a difference between two hierarchies, it is presented in Table 4 with “()”, in which the value corresponds to the LP bounds (3). It can be seen that there is a pay-off between using (3) and the BSOS hierarchy. By using the LP bounds you may solve each level faster (4 cases) but the lower bound can be strictly weaker than the one from the BSOS hierarchy (2 cases).

4.3 Lower bounds using PQ-formulation

Up to now, we evaluated the BSOS hierarchy on the P-formulation. Since the McCormick relaxation (Sect. 3.1) of the PQ-formulation is stronger than that of the P-formulation Gupte et al. (2017), it is worthwhile to evaluate the BSOS hierarchy using the PQ-formulation. In Table 5 we present these results for the PQ-formulation. To eliminate the equality constraints, we replace them by two inequalities.

Table 4 Results for computing the bounds from (3) and \hat{q}_d^1 in (27) after elimination of linearly dependent constraints

	Iteration	Rel.time	solution	# Lin. var.	Size of SDP var.	# Const.
Haverly1	d = 1	1	−600.00	12	6	21(8)
	d = 2	1	−417.20	199	6	33
	d = 3	0.11	− 400.00	1937	6	98
Haverly2	d = 1	1	−1200.00	12	6	21(8)
	d = 2	1	−601.67(−640.00)	199	6	33
	d = 3	0.13	− 600.00	1937	6	98
Haverly3	d = 1	1	−875.00	12	6	21(8)
	d = 2	1	− 750.00	199	6	33
Ben-Tal4	d = 1	1	−650.00	14	7	28(9)
	d = 2	1	−467.20	274	7	42
	d = 3	0.08	− 450.00	3095	7	140
Ben-Tal5	d = 1	1(0.11)	− 3500.00	55	30	465(44)
DeyGupte4	d = 1	1	−4.00	53	11	66(16)
	d = 2	0.03	−3.86	4135	11	131
	d = 3	–	–	172,250	11	–
Foulds2	d = 1	1	−1200.00	39	19	190(24)
	d = 2	0.002	−1191.30	2224	19	295
	d = 3	–	–	49,385	19	–
Foulds3	d = 1	0.94(10 ^{−4})	− 8.00	220	153	11,781(176)
Foulds4	d = 1	0.92(10 ^{−4})	− 8.00	220	153	11,781(176)
Adhya1	d = 1	1	−999.32	42	12	78(24)
	d = 2	0.06(0.5)	≈−723.95	2583	12	260
	d = 3	–	–	85,526	12	–
Adhya2	d = 1	1	−798.29	54	12	78(24)
	d = 2	0.12(0.3)	−576.83	4293	12	260
	d = 3	–	–	182,214	12	–
Adhya3	d = 1	1	−882.84	67	18	171(38)
	d = 2	0.02	≈−802.88(−806.64)	6634	18	671
	d = 3	–	–	348,602	18	–
Adhya4	d = 1	1	−1055.00	52	17	153(39)
	d = 2	0.03	≈−1035.54	3979	18	732
	d = 3	–	–	162,657	17	–
RT2	d = 1	1	−45,420.48	68	15	120(23)
	d = 2	0.02	−36,541.89	6836	15	266
	d = 3	–	–	364,480	15	–
sppA0	d = 1	0.99(10 ^{−4})	−47,675.00	817	162	13,203(372)
	d = 2	–	–	1,000,008	162	–

The values in “()” are corresponding to the LP bounds (3) if they are different than those from \hat{q}_d^1

Table 5 Results for computing the bounds \hat{q}_d^1 in (27) after elimination of linearly dependent constraints on the PQ-formulation

	Iteration	Rel. time	Solution	# Lin. var.	Size of SDP var.	# const.
Haverly1	d = 1	1	−600.00	25	9	45
	d = 2	1	−411.11	901	9	82
	d = 3	–	Numerical Prob.	17,901	9	354
Haverly2	d = 1	1	−1200.00	25	9	45
	d = 2	1	−600.00	901	9	82
Haverly3	d = 1	1	−875.00	25	9	45
	d = 2	1	−750.00	901	9	82
Ben-Tal4	d = 1	1	−650.00	30	11	66
	d = 2	1	−459.86	1306	11	124
	d = 3	–	–	31,031	11	–
Ben-Tal5	d = 1	4.17	−3500.00	127	45	1035
DeyGupte4	d = 1	1	−4.00	89	17	153
	d = 2	0.35	≈ -2.49	11,749	17	438
	d = 3	–	–	818,445	17	–
Foulds2	d = 1	1	−1200.00	77	25	325
	d = 2	–	–	8779	25	–
Foulds3	d = 1	2.26	−8.00	628	193	18,721
Foulds4	d = 1	2.32	−8.00	628	193	18,721
Adhya1	d = 1	1	−999.32	73	19	190
	d = 2	–	–	7885	19	–
Adhya2	d = 1	1	−798.29	81	19	190
	d = 2	–	–	9721	19	–
Adhya3	d = 1	2	−882.84	109	29	435
	d = 2	–	–	17,659	29	–
Adhya4	d = 1	2	−1,055.00	96	27	378
	d = 2	–	–	13,680	27	–
RT2	d = 1	1	−18,155.84	96	23	276
	d = 2	–	–	13,680	23	–
sppA0	d = 1	5.88	−47,675.00	1165	234	27,495
	d = 2	–	–	1,326,340	234	–

4.4 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 on the dimension of N_d , which is an upper bound on the number of linearly independent constraints in (5).

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 ω in the size of $w(x)$.

Therefore, $\text{rank}(M_d)$ is an upper bound on $\text{rank}(N_d)$, and 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 (8), the functions defining the inequality constraints can be partitioned into three classes:

- (I) bilinear functions,
- (II) $x_i, i = 1, \dots, n$,
- (III) some other affine functions.

The bilinear functions are those related to constraints (15) and (16), or those related to the constraints (13) 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,

$$\left\langle \left\{ \begin{pmatrix} 1, \{y_{ij}\}_{\substack{i \in \mathcal{I} \\ j \in \mathcal{J}'}} \\ \{y_{lj}\}_{\substack{l \in \mathcal{L} \\ j \in \mathcal{J}'}} \\ \{y_{il}\}_{(i,l) \in \tilde{\mathcal{I}}} \\ \{p_{lk}\}_{(l,k) \in \tilde{\mathcal{L}}} \\ \{p_{lk}y_{lj}\}_{(l,k,j) \in \tilde{\mathcal{J}}} \end{pmatrix}^{\alpha} \right\}_{\alpha \in \mathbb{N}_d^{\omega}} \right\rangle = M_d,$$

where $\tilde{\mathcal{I}}$, $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{J}}$ are respectively including (i, l) , (l, k) and (l, k, j) that y_{il} , p_{lk} and $p_{lk}y_{lj}$ appear in (8) after elimination of the equality constraints, and

$$\omega = 1 + I \times L + L \times J + |\tilde{\mathcal{I}}| + |\tilde{\mathcal{L}}| + |\tilde{\mathcal{J}}|.$$

Clearly the number of variables in the pooling problem (8) after elimination of equality constraints is $I \times L + L \times J + |\tilde{\mathcal{I}}| + |\tilde{\mathcal{L}}|$. For $d = 1$, we prove in Theorem 4 that all of the constraints in (5) 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 $k \in \mathcal{K}$ and $l \in \mathcal{L}$, 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 $\left[\binom{K \times L \times J}{2} + K \times L \times J \right]$, 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

$$\binom{n+2}{2} + K \times L \times \left[\binom{n+1}{2} - \binom{n-J+1}{2} \right] + \binom{K \times L \times J}{2} + K \times L \times J. \quad (28)$$

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

$$\begin{aligned}
 & \underbrace{\binom{n+3}{3}}_{\text{monomials up to degree 3}} + \underbrace{K \times L \times \left[\binom{n+2}{3} - \binom{n-J+2}{3} \right]}_{\text{monomials of degree 4}} \\
 & + \underbrace{K \times L \times \left[\binom{n+2}{3} - \binom{n-J+2}{3} - J \times \binom{n-J+1}{2} \right]}_{\text{monomials of degree 5}} \\
 & + \underbrace{\left[\binom{K \times L \times J}{3} + K \times L \times J + 2 \binom{K \times L \times J}{2} \right]}_{\text{monomials of degree 6}}. \tag{29}
 \end{aligned}$$

Example 2 Consider the example (26). The only bilinear terms in (26) are $y_1 y_2$ and $y_1 y_3$. So,

$$M_d = \left\{ (1, y_1, y_2, y_3, y_4, y_5, y_1 y_2, y_1 y_3)^\alpha \right\}_{\alpha \in \mathbb{N}_d^8}$$

Therefore, the number of linearly independent constraints is at most

$$\binom{7}{2} + \binom{6}{2} - \binom{4}{2} + \binom{2}{2} + 2 = 33,$$

if $d = 2$, and

$$\binom{8}{3} + 2 \times \binom{7}{3} - 2 \times \binom{5}{3} - 2 \times \binom{4}{2} + 2 \times \binom{2}{2} + 2 = 98,$$

if $d = 3$. □

4.5 Improving lower bounds by adding valid inequalities

Adding redundant constraints to the original problem (1) increases the number of linear variables in (4); this introduces some flexibility in each level of the hierarchy because of the new linear variables and may provide a stronger lower bound. As it was mentioned in Sect. 3.1, for each bilinear term in the P- or PQ-formulations there are four valid inequalities given by (19). So, in Table 6 we present the result of adding these valid inequalities to the P-formulation and using \hat{q}_d^1 in (27) to solve the problem. In each level of the hierarchy in this table, we use the upper bounds for the number of constraints proposed in Sect. 4.4. As Table 6 shows, this improvement helps to obtain the optimal values of Haverly1, Harverly2, Ben-Tal4, and DeyGupte4, and to get a good approximation of the optimal value of Foulds2 in the second level of the hierarchy. Also, for Adhya1,2,4 we obtained better lower bounds than the PQ-linear relaxation values in Table 1.

Table 6 Results for computing the lower bounds \hat{q}_d^1 for the P-formulation after adding valid inequalities and considering (28) and (29) as the upper bound on the number of linearly independent constraints

	Iteration	Rel. time	Solution	# Lin. var.	Size of SDP var.	# const.
Haverly1	d = 1	1	−600.00	18	6	21
	d = 2	1	− 400.00	460	6	33
Haverly2	d = 1	1	−1100.00	18	6	21
	d = 2	1	− 600.00	460	6	33
Haverly3	d = 1	1	−850.00	18	6	21
	d = 2	1	− 750.00	460	6	33
Ben-Tal4	d = 1	1	−650.00	20	7	28
	d = 2	1	− 450.00	571	7	42
Ben-Tal5	d = 1	1	− 3500.00	145	30	465
DeyGupte4	d = 1	1	−4.00	101	11	66
	d = 2	0.32	− 1.02	15,155	11	170
Foulds2	d = 1	1	−1200.00	63	19	190
	d = 2	0.02	−1101.83	5860	19	358
	d = 3	–	–	289,695	19	2850
Foulds3	d = 1	1.60	− 8.00	604	153	11,781
Foulds4	d = 1	1.53	− 8.00	604	153	11,781
Adhya1	d = 1	1	−960.37	138	12	78
	d = 2	1.34	−640.19	28,360	12	270
	d = 3	–	–	3,056,470	12	2860
Adhya2	d = 1	1	−777.63	198	12	78
	d = 2	1.13	−569.55	58,510	12	270
	d = 3	–	–	9,036,390	12	4108
Adhya3	d = 1	1	−879.02	283	18	171
	d = 2	1.06	−664.39	119,710	18	691
	d = 2	–	–	26,402,485	18	13,452
Adhya4	d = 1	1	−1032.50	172	17	153
	d = 2	1.58	−948.88	44,119	17	1038
	d = 3	–	–	5,921,616	17	7089
RT2	d = 1	1	−36,542.22	212	15	120
	d = 2	0.091	≈−32,739.03	67,099	15	354
	d = 3	–	–	11,093,536	15	6440
sppA0	d = 1	1.39	≈−46,636.57	4705	162	13,203
	d = 2	–	–	37,414,021	162	94,812

5 Conclusion

In this paper we analysed and evaluated the bounded degree sum-of-squares (BSOS) hierarchy of Lasserre et al. (2015) 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.

Acknowledgements The authors would like to thank Claudia D'Ambrosio and Ruth Misener for useful discussions and providing us some references. Also, we are grateful to the two anonymous reviewers for their helpful and constructive comments, in particular Remark 3 and Sect. 3.2.1, that greatly contributed to improving the paper.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix: A new pooling problem instance

In this section we use a result in Dey and Gupte (2015) to construct a pooling problem instance (DeyGupte4) for which piecewise linear approximations of the PQ-formulation fail.

Consider a standard pooling problem with $I = 2$ inputs, $L = 2$ pools and $J = 4$ outputs. Assume that both inputs are connected to the pools and both pools are connected to the outputs (Fig. 3). Let $K = 2$ and the concentration of specifications be $(1, 0)$ and $(0, 1)$ for the first and second input, respectively. We number the inputs by 1, 2, pools by 3, 4, and outputs by 5, 6, 7, 8. Let $\mu_{jk}^{max} = \mu_{jk}^{min}$ (given in Fig. 3), $u_{il} = 4$ and $u_{lj} = 1$, for $i = 1, 2, l = 3, 4, j = 5, 6, 7, 8$, and $k = 1, 2$. Set the capacity of inputs, pools and outputs $C_1 = C_2 = 8, C_3 = C_4 = 4$, and $C_5 = C_6 = C_7 = C_8 = 1$. Let

$$\delta := \min \left\{ \|\mu_{\hat{j}k}^{max} - \mu_{\bar{j}k}^{max}\|_2 : \hat{j} \neq \bar{j}, \hat{j}, \bar{j} = 5, 6, 7, 8, k = 1 \right\} \approx 0.014,$$

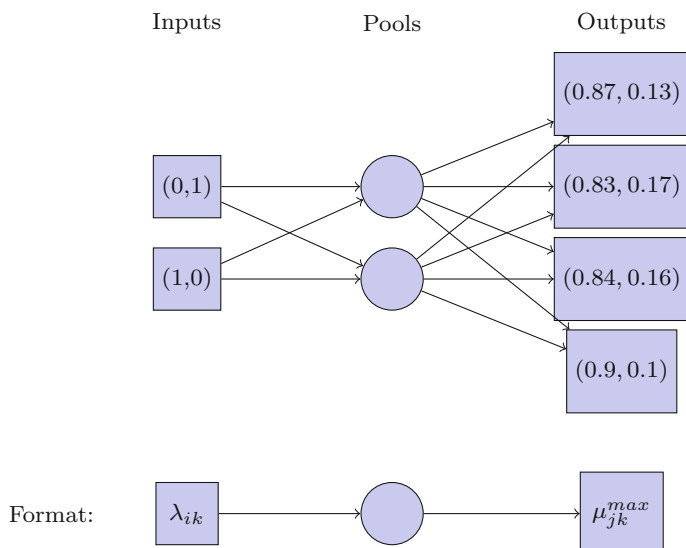


Fig. 3 Flowchart of DeyGupte4

$c_{il} = 0$, for $i = 1, 2$ and $l = 3, 4$. Set $c_{3j} = -1$, $c_{4j} = \frac{2}{8}$, for all $j = 5, 6, 7, 8$, and the rest of the costs as 0.

The optimal value of this problem is -1 with the optimal solution constructed by sending flows from inputs to the first pool, and from it to one of the outputs such that the restriction in the specification in it is satisfied (Dey and Gupte 2015).

Let $g, h : [0, 1] \times [0, 1] \rightarrow \mathbb{R}$ be piecewise linear functions such that

$$g(\alpha, \beta) \geq \alpha\beta \geq h(\alpha, \beta), \quad \forall \alpha, \beta \in [0, 1].$$

Assume that

$$S = \{(\alpha, \beta, \chi) \mid g(\alpha, \beta) \geq \chi \geq h(\alpha, \beta), \alpha, \beta \in [0, 1]\},$$

is the piecewise linear approximation of

$$B = \{(\alpha, \beta, \chi) \mid \chi = \alpha\beta, \alpha, \beta \in [0, 1]\},$$

such that for all $\alpha, \beta \in [0, 1]$ and $|e| \leq 0.05$, $(\alpha, \beta, \alpha\beta + e) \in S$. As it was mentioned in Sect. 3.1, in the pooling problem:

$$\begin{aligned} m_\lambda &\leq p_{lk} \leq M_\lambda, & \forall l \in \mathcal{L}, k \in \mathcal{K}, \\ 0 &\leq y_{lj} \leq \min\{C_j, u_{lj}\}, & \forall j \in \mathcal{J}, l \in \mathcal{L}. \end{aligned}$$

So, for any bilinear term $v_{ljk} = \frac{p_{lk} - m_\lambda}{M_\lambda - m_\lambda} \frac{y_{lj}}{\min\{C_j, u_{lj}\}}$, $l \in \mathcal{L}$, $j \in \mathcal{J}$, $k \in \mathcal{K}$, one can use S to find a lower bound for the optimal value of the pooling problem. It is proved in Dey and Gupte (2015) that applying this approximation to the PQ-formulation gives an objective value in $[-4, -3]$.

References

- Adhya, N., Tawarmalani, M., & Sahinidis, N. V. (1999). A Lagrangian approach to the pooling problem. *Industrial and Engineering Chemistry Research*, 38(5), 1956–1972.
- Alfaki, M. (2012). *Models and solution methods for the pooling problem*. Ph.D. thesis, University of Bergen
- Alfaki, M., & Haugland, D. (2013). Strong formulations for the pooling problem. *Journal of Global Optimization*, 56(3), 897–916.
- Alfaki, M., & Haugland, D. (2014). A cost minimization heuristic for the pooling problem. *Annals of Operations Research*, 222(1), 73–87.
- Ben-Tal, A., Eiger, G., & Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical Programming*, 63(1–3), 193–212.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.
- Dey, S. S., & Gupte, A. (2015). Analysis of MILP techniques for the pooling problem. *Operations Research*, 63(2), 412–427.
- Foulds, L. R., Haugland, D., & Jörnsten, K. (1992). A bilinear approach to the pooling problem. *Optimization*, 24(1–2), 165–180.
- Frimannslund, L., El Ghami, M., Alfaki, M., & Haugland, D. (2010). Solving the pooling problem with LMI relaxations. In: S. Cafieri, B.G.-Tóth, E. Hendrix, L. Liberti & F. Messine (Eds.), *Proceedings of the Toulous Global Optimization Workshop (TOGO)*, pp. 51–54.
- Gupte, A., Ahmed, S., Dey, S. S., & Cheon, M. S. (2017). Relaxations and discretizations for the pooling problem. *Journal of Global Optimization*, 67(3), 631–669. doi:10.1007/s10898-016-0434-4.
- Haugland, D. (2016). The computational complexity of the pooling problem. *Journal of Global Optimization*, 64(2), 199–215.
- Haverly, C. A. (1978). Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, 25, 19–28.
- Karupiah, R., & Grossmann, I. E. (2006). Global optimization for the synthesis of integrated water systems in chemical processes. *Computers and Chemical Engineering*, 30(4), 650–673.
- Krivine, J. L. (1964). Anneaux préordonnés. *Journal d'Analyse Mathématique*, 12(1), 307–326.

- Lasserre, J. B. (2001). Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3), 796–817.
- Lasserre, J. B. (2005). Polynomial programming: LP-relaxations also converge. *SIAM Journal on Optimization*, 15(2), 383–393.
- Lasserre, J. B., Toh, K., & Yang, S. (2015). A bounded degree SOS hierarchy for polynomial optimization. *EURO Journal on Computational Optimization*, 1–31. doi:[10.1007/s13675-015-0050-y](https://doi.org/10.1007/s13675-015-0050-y).
- Laurent, M. (2009). Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry* (pp. 157–270). Berlin: Springer.
- Lofberg, J., & Parrilo, P. A. (2004). From coefficients to samples: A new approach to SOS optimization. In 2004. CDC. 43rd IEEE Conference on Decision and Control, vol. 3, pp 3154–3159. IEEE.
- Misener, R., & Floudas, Ch. A. (2009). Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied Computational Mathematics*, 8(1), 3–22.
- Misener, R., Thompson, J. P., & Floudas, Ch. A. (2011). APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Computers and Chemical Engineering*, 35(5), 876–892.