A dynamic inequality generation scheme for polynomial programming

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

A Dynamic Inequality Generation Scheme for Polynomial Programming

Bissan Ghaddar · Juan C. Vera · Miguel F. Anjos

Received: date / Accepted: date

Abstract Hierarchies of semidefinite programs have been used to approximate or even solve polynomial programs. This approach rapidly becomes computationally expensive and is often tractable only for problems of small size. In this paper, we propose a dynamic inequality generation scheme to generate valid polynomial inequalities for general polynomial programs. When used iteratively, this scheme improves the bounds without incurring an exponential growth in the size of the relaxation. As a result, the proposed scheme is in principle scalable to large general polynomial programming problems. When all the variables of the problem are non-negative or when all the variables are binary, the general algorithm is specialized to a more efficient algorithm. In the case of binary polynomial programs, we show special cases for which the proposed scheme converges to the global optimal solution. We also present several examples illustrating the computational behavior of the scheme and provide comparisons with Lasserre's approach and, for the binary linear case, with the lift-and-project method of Balas, Ceria, and Cornuéjols.

Keywords Polynomial programming · Binary polynomial programming · Semidefinite programming · Inequality generation

1 Introduction

A polynomial program is a mathematical optimization problem whose objective and constraints are multivariate polynomials. Polynomial programming generalizes several special cases that have been thoroughly studied in optimization, including mixed binary linear programming, convex and non-convex quadratic programming, and linear complementarity problems. Polynomial programs arise in several practical applications in the context of control, process engineering, facility location, economics and equilibrium, and finance. It is well known that solving polynomial programs is an $\mathcal{NP}$-hard problem.

Since the work of Lasserre [15] and Parrilo [25], there has been a lot of research activity to devise solution schemes to solve polynomial programs. These schemes are based on applying representation theorems from algebraic geometry to characterize the set of polynomials that are non-negative on a given domain. This research includes the recent work of de Klerk and Pasechnik [5], Lasserre [14,15], Laurent [17,18], Nie, Demmel, and Sturmfels [22], Parrilo [23,25], Peña, Vera, and Zuluaga [28,36], and the early work of Nesterov [20], Shor [34], and the $\mathcal{S}-$Lemma of Yakubovich (see [30]) among others. The recent handbook [1] provides an overview of the research activity in the area of polynomial programming.

An extended abstract version of this paper has appeared in the Proceedings of IPCO 2011.

B. Ghaddar
IBM Research, Dublin Technology Campus,
Damastown Ind. Park, Mulhuddart, Dublin 15, Ireland.
Research supported by a Canada Graduate Scholarship from the Natural Sciences and Engineering Research Council of Canada.
E-mail: bghaddar@uwaterloo.ca

J. C. Vera
Tilburg School of Economics and Management, Tilburg University,
Tilburg, The Netherlands.
E-mail: j.c.veralizcano@uvt.nl

M. F. Anjos
Canada Research Chair in Discrete Nonlinear Optimization in Engineering, GERAD & École Polytechnique de Montréal, Montréal, QC, Canada H3C 3A7.
Research partially supported by the Natural Sciences and Engineering Research Council of Canada, and by a Humboldt Research Fellowship.
E-mail: anjos@stanfordalumni.org

The general approach to solve polynomial programs is based on sum-of-squares (SOS) certificates of non-negativity for multivariate polynomials. This approach builds hierarchies of relaxations leading to solving a sequence of positive semidefinite programs. Under mild conditions, the resulting semidefinite relaxations provide bounds that converge to the optimal value of the original polynomial program. Allowing for more complex certificates produces better approximations to the original polynomial program, but the size of such approximations explodes quickly, even for instances with a small number of variables. As a result, despite the theoretical strength of SOS representations, the current schemes are only able to handle problems of small sizes. For efficient practical performance in medium or large-scale problems, it becomes necessary to exploit the problem structure. For example, by taking advantage of the symmetry such as the work of Bai, de Klerk, Pasechnik, and Sotirov [2], de Klerk [4], de Klerk, Pasechnik, and Schrijver [6], de Klerk and Sotirov [7], and Gatermann and Parrilo [10] or by exploiting sparsity such as in Kim, Kojima, and Toint [13] and Nie and Demmel [21] and the references therein. However, in the absence of structure, the practical application of the SOS approach is severely limited.

In this paper, a dynamic inequality generation scheme (DIGS) is proposed for general polynomial programs. The key idea of DIGS is to bound the complexity (degree) of the non-negative certificates, avoiding the exponential growth of the relaxations. Instead, our approach makes use of information from the objective function to construct improved approximations of the polynomial program, by dynamically generating polynomial inequalities that are valid on the feasible region. These valid inequalities are used to construct new non-negative certificates. The iterated generation of inequalities yields better and better approximations to the polynomial program without growing the degree of the certificates involved. Depending on the original problem and the type of relaxation used, the iterative procedure solves a sequence of linear, second-order cone, or semidefinite problems. For the purposes of this paper, we focus on semidefinite relaxations and thus obtain a sequence of semidefinite problems.

In the rest of this section we discuss Lasserre's Hierarchy for Polynomial Programs. In Section 2 we introduce our generic master-subproblem dynamic inequality generation scheme (Algorithm 1). Algorithm 1 is a rather abstract scheme where the subproblem is not completely specified. In the rest of the paper we focus on the specification of the subproblem. In Section 2.1, a practical algorithm for general Polynomial Programs is presented (Algorithm 2). In the presence of extra structure, the subproblem can be implemented more efficiently; this is done in Section 2.2 for Polynomial Programs where all variables are non-negative and in Section 3 where (some of) the variables are binary; we refer to such problems as (mixed) binary polynomial programs (BPP). The method presented in Section 3 for the binary case can be seen as a generalization, from binary linear programs to BPPs, of the lift-and-project methods of Balas, Ceria, and Cornuéjols [3], Sherali and Adams [32], and Lovász and Schrijver [19] (see Section 3.3). Our practical methods do not necessarily converge to the optimal solution (see Example 5). The motivation for developing such methods is that, independently of the structure of the problem, they can efficiently strengthen the bounds which is helpful when incorporated into a branch-and-bound framework. Similar to the lift-and-project methods, these techniques can then be integrated in the design of algorithms and solvers to efficiently solve polynomial programming problems. Convergence to the global optimal solution is proven for a family of problems in the binary case (Theorems 1, 2 and 3). This specialized scheme and the convergence results are presented in Section 3. To evaluate the proposed approach, computational results are presented for general polynomial programs (Section 2.1.2), non-negative polynomial programs (Section 2.2.1) and binary polynomial programs (Section 3.4); in each case we compare our methodology to Lasserre's approach [14,15]. For the binary linear case, we compare our methodology to the lift-and-project method of Balas, Ceria, and Cornuéjols [3] (Section 3.3).

1.1 Polynomial Programming

Consider the general polynomial programming (PP) problem whose objective and constraints are multivariate polynomials:

$$\textbf{(PP-P)}\ \rho = \inf_{x} \quad f(x)$$
$$\text{s.t. } g_i(x) \geq 0, \qquad i = 1, \dots, m.$$

When it is convenient, we represent a polynomial $f(x)$ of degree $\deg(f) = d$ using use multinomial notation, i.e., we write $f(x) = \sum_{|\alpha| \leq d} f_\alpha x^\alpha$, where $\alpha = (\alpha_1, \dots \alpha_n) \in \mathbb{N}^n$, $|\alpha| = \sum_i^n \alpha_i$. We use $\mathbf{R}[x] := \mathbf{R}[x_1, \dots, x_n]$ (resp. $\mathbf{R}_d[x]$) to denote the set of polynomials in n variables with real coefficients (resp. of degree at most d).

Let $S = \{x \in \mathbb{R}^n : g_i(x) \geq 0, i = 1, \dots, m\}$ be the feasible set of (PP-P). Then (PP-P) can be rephrased as

$$\sup_{\lambda} \lambda$$
$$\text{s.t. } f(x) - \lambda \geq 0 \quad \forall\, x \in S,$$

that is,

$$\textbf{(PP-D)} \quad \begin{array}{ll} \sup\limits_{\lambda} \lambda \\ \text{s.t. } f(x) - \lambda \in \mathcal{P}_d(S), \end{array}$$

where d is the maximum among $\deg(f(x))$ and $\deg(g_i(x)), i = 1, \ldots, m$, and $\mathcal{P}(S)$ (resp. $\mathcal{P}_d(S)$) is the cone of polynomials (resp. of degree at most d) that are non-negative over $S \subseteq \mathbb{R}^n$. We refer to d as the degree of (PP-P).

The condition $f(x) - \lambda \in \mathcal{P}_d(S)$ is $\mathcal{NP}$-hard for most (interesting) choices of S and $d \geq 1$. Computable relaxations of (PP-D) are obtained using tractable approximations of the cone $\mathcal{P}_d(S)$ which can be re-phrased in terms of a linear system of equations involving positive semidefinite matrices [15, 20, 24–26, 34, 35], second-order cones [12], or linear optimization problems [16, 33, 36]. These approximations can be solved efficiently using interior-point methods.

The general idea in these methods is to relax the condition $f(x) - \lambda \in \mathcal{P}_d(S)$ to $f(x) - \lambda \in \mathcal{K}$ for a suitable $\mathcal{K} \subseteq \mathcal{P}_d(S)$. Defining the relaxation

$$\begin{array}{ll} \mu_{\mathcal{K}} = \sup\limits_{\lambda} \lambda \\ \text{s.t. } f(x) - \lambda \in \mathcal{K}, \end{array}$$

it follows that $\mu_{\mathcal{K}}$ is a lower bound on the original problem (PP-P).

1.2 Lasserre's Hierarchy for Polynomial Programs

Lasserre [15] introduced semidefinite relaxations corresponding to liftings of the polynomial programs into higher dimensions. The construction is motivated by results related to representations of non-negative polynomials as SOS and the dual theory of moments. Lasserre shows that computing the global minimum of $f(x)$ over a set S defined by polynomial inequalities reduces to solving a sequence of SOS-type representations of polynomials that are non-negative on S. The convergence of Lasserre's method is based on the assumption that $\{g_1(x), \ldots, g_m(x)\}$, the given description of S, allows the application of Putinar's Theorem [31]. In particular, it assumes S is compact.

For ease of notation let $g_0(x) \equiv 1$ and $G = \{g_i(x) : i = 0, 1, \ldots, m\}$. For a given $r > 0$, let $\mathcal{K}_G^r$ be the $r-$th truncated quadratic module generated by G, that is,

$$\mathcal{K}_G^r = \sum_{i=0}^{m} g_i(x) \Sigma_{r - \deg(g_i)},$$

where for any $d \geq 0$, Σ_r denotes the cone of real polynomials of degree at most d that are SOS. Note that $\Sigma_d := \{\sum_{i=1}^{N} p_i(x)^2 : p(x) \in \mathbf{R}_{\lfloor \frac{d}{2} \rfloor}[x]\}$, with $N = \binom{n+d}{d}$, and in particular $\Sigma_d = \Sigma_{d-1}$ for every odd degree d.

As $\mathcal{K}_G^r \subseteq \mathcal{P}(S)$, we obtain a relaxation of (PP-D):

$$\begin{array}{ll} \mu_G^r = \sup\limits_{\lambda} \lambda \\ \text{s.t. } f(x) - \lambda \in \mathcal{K}_G^r, \end{array}$$

so that $\mu_G^r \leq \rho$ and

$$\begin{array}{ll} \mu_G^r = \sup\limits_{\lambda, \sigma_i(x)} \lambda \\ \text{s.t. } f(x) - \lambda = \sum\limits_{i=0}^{m} g_i(x) \sigma_i(x) \\ \sigma_i(x) \in \Sigma_{r - \deg(g_i)}, \qquad i = 0, \ldots, m. \end{array} \tag{1}$$

The optimization problem (1) can be reformulated as a (convex) semidefinite optimization problem [34]. By increasing r, a sequence of semidefinite relaxations of increasing size is obtained. Lasserre shows [15] that under mild conditions, the optimal values of these relaxations converge to the global optimal value of the original non-convex problem (PP-P). Proposition 1 states the result using our notation.

Proposition 1 *Let $G = \{g_i(x) : i = 0, \ldots, m\}$ be such that $g_0(x) \equiv 1$. Assume there exists a real-valued polynomial $u(x) \in \sum_{i=0}^{m} g_i(x) \Sigma_m$ for some m, such that $\{x : u(x) \geq 0\}$ is compact. Then (Putinar [31])*

$$\mathcal{K}_G^1 \subseteq \mathcal{K}_G^2 \subseteq \cdots \subseteq \mathcal{K}_G^r \subseteq \cdots \subseteq \mathcal{P}(S) \text{ and } \{p \in \mathbf{R}[x] : p(s) > 0 \; \forall s \in S\} \subseteq \bigcup_{r>0} \mathcal{K}_G^r$$

and therefore (Lasserre [15])

$$\mu_G^1 \leq \mu_G^2 \leq \cdots \leq \mu_G^r \leq \cdots \leq \rho \text{ and } \mu_G^r \to \rho \text{ as } r \to \infty.$$

In other words, using Lasserre's hierarchy for general polynomial programs one may approximate the global optimal value ρ as closely as desired by solving a sequence of semidefinite problems with increasing size of the semidefinite matrices and number of constraints. The computational cost of the procedure depends on r, the number of constraints m, and the number of variables n. For (PP-P) with n variables and m inequality constraints, the size of the optimization problem (1) is as follows:

- one psd matrix of size $\binom{n+r}{r}$;
- m psd matrices, each of size $\binom{n+r-\deg(g_i)}{r-\deg(g_i)}$ for $i = 1, \ldots, m$;
- $\binom{n+r}{r}$ linear constraints.

The number of constraints of (1) grows exponentially in r. Notice that to use all polynomial appearing in the formulation of (PP-P), r should be no smaller than d. When $r > d+2$, the number of variables and constraints of (1) can be large, especially when (P-PP) involves high degree polynomials.

1.2.1 Equality Constraints

Lasserre's approach treats equality constraints as pairs of two inequalities. Alternatively, one can differentiate between equality and inequality constraints as proposed in [28]. Given $S = \{x : g_i(x) \geq 0,\ i = 1, \ldots, m_1,\ h_i(x) = 0,\ i = 1, \ldots, m_2\}$, let $G = \{g_i(x) : i = 0, 1, \ldots, m_1,\ h_i(x) : i = 1, \ldots, m_2\}$ where $g_0(x) = 1$. Consider the following approximation of $\mathcal{P}_d(S)$:

$$\mathcal{K}^r_G = \sum_{i=0}^{m_1} g_i(x)\Sigma_{r-\deg(g_i)} + \sum_{i=1}^{m_2} h_i(x)\mathbf{R}_{r-\deg(h_i)}[x].$$

The corresponding optimization problem over S can be written as:

$$\begin{aligned} \sup_{\lambda,\sigma_i(x),\delta_i(x)} \quad & \lambda \\ \text{s.t. } & f(x) - \lambda = \sum_{i=0}^{m_1} \sigma_i(x) g_i(x) + \sum_{i=1}^{m_2} \delta_i(x) h_i(x) \\ & \sigma_i(x) \in \Sigma_{r-\deg(g_i)}, \qquad i = 0, \ldots, m_1 \\ & \delta_i(x) \in \mathbf{R}_{r-\deg(h_i)}[x], \qquad i = 1, \ldots, m_2. \end{aligned} \tag{2}$$

In a similar way to problem (1), problem (2) can be reformulated as a semidefinite optimization problem.

2 Dynamic Approximation of Polynomial Programs

We propose a new scheme to generate a sequence of improving approximations for (P-PP). Instead of growing r and exponentially increasing the size of the relaxation (1), we fix r in (1) to a small value r_0 (mainly to d, the degree of (PP-P)) and the relaxation (1) is improved by adding valid polynomial inequalities to the description of S, i.e., by growing the set G.

The scheme is dynamic, and consists of a master problem and a subproblem. The master problem is of the form (1) and provides bounds for problem (P-PP), while the subproblem uses the optimal dual information from the master to generate polynomial inequalities that are valid on the feasible region. These valid inequalities are then incorporated into the master to construct new non-negativity certificates, obtaining better approximations of (PP-P).

An augmented description G of S does not necessarily imply a better approximation. If $p \in \mathcal{K}^{r_0}_G$, then $\mathcal{K}^{r_0}_{G\cup\{p(x)\}} = \mathcal{K}^{r_0}_G$, and thus $\mu^{r_0}_G = \mu^{r_0}_{G\cup\{p(x)\}}$. On the other hand, if $p(x) \geq 0$ is a valid inequality for S not in $\mathcal{K}^{r_0}_G$, we have $\mathcal{K}^{r_0}_G$ strictly contained in $\mathcal{K}^{r_0}_{G\cup\{p(x)\}}$ improving the conic approximation to $\mathcal{P}_d(S)$ which may provide a better bound for (PP-P) when using (1). We summarize this observation in Lemma 1.

Lemma 1 *Let $r_0 \geq d$ and $p(x) \in \mathcal{P}_{r_0}(S) \setminus \mathcal{K}^{r_0}_G$. Then*

$$\mathcal{K}^{r_0}_G \subsetneq \mathcal{K}^{r_0}_{G\cup\{p(x)\}} \subseteq \mathcal{P}(S) \text{ and thus } \mu^{r_0}_G \leq \mu^{r_0}_{G\cup\{p(x)\}}.$$

We are interested in generating valid inequalities $p(x)$ for which $\mu^{r_0}_G < \mu^{r_0}_{G\cup\{p(x)\}}$. We call such inequalities *improving inequalities*. Given a finite description $G \subseteq \mathbf{R}_d[x]$ for S and $r_0 \geq d$, Algorithm 1 is a generic dynamic inequality generation scheme (DIGS) based on Lemma 1 to construct a sequence of improving relaxations.

Algorithm 1 Generic Dynamic Inequality Generation Scheme (DIGS) for (PP-P)

Require: $G \subseteq \mathbf{R}_d[x]$ description of S and $r_0 \geq d$
$s \leftarrow 0$, $G_0 \leftarrow G$.
loop
Let

$$\textbf{(PP-M}_s\textbf{)}\ \nu_s = \sup_{\lambda}\ \lambda \quad \text{s.t. } f(x) - \lambda \in \mathcal{K}^{r_0}_{G_s}.$$

Look for an improving inequality $p_s(x) \in \mathcal{P}_{r_0}(S) \setminus \mathcal{K}^{r_0}_{G}$
if an improving inequality does not exist **then**
STOP
else
$G_{s+1} \leftarrow G_s \cup \{p_s(x)\}$
$s \leftarrow s + 1$
end if
end loop

Lemma 1 implies that Algorithm 1 generates a sequence of strictly monotone approximations to $\mathcal{P}_d(S)$, and a corresponding sequence of improving bounds for the optimal value of (PP-P). This is formally stated in Lemma 2.

Lemma 2 *Let G_s and ν_s, $s = 1, 2, \ldots$ be generated by Algorithm 1. Then*

$$\mathcal{K}^{r_0}_{G_0} \subsetneq \mathcal{K}^{r_0}_{G_1} \subsetneq \cdots \subsetneq \mathcal{K}^{r_0}_{G_s} \subsetneq \mathcal{K}^{r_0}_{G_{s+1}} \cdots \subseteq \mathcal{P}(S)$$

$$\nu_0 < \cdots < \nu_s < \nu_{s+1} < \cdots \leq \rho.$$

Moreover if Algorithm 1 stops at iteration s, then that bound is optimal, i.e., $\nu_s = \rho$.

Proof By definition of improving inequality it follows that $\nu_s < \nu_{s+1}$ for all s. In particular $\mathcal{K}^{r_0}_{G_s} \subsetneq \mathcal{K}^{r_0}_{G_{s+1}}$. Also, if the algorithm stops at iteration s is because no improving inequality exists, thus the current relaxation must have optimal value $\nu_s = \rho$. □

Notice that the DIGS presented in Algorithm 1 is a paradigm shift. Instead of approximating the whole set $\mathcal{P}(S)$ by increasing the size of the non-negativity certificates, the complexity of certificates is increased by iteratively using the old certificates to produce new ones while keeping the degree, and hence the size, fixed. In each iteration of the algorithm, the description of S is improved, as new valid inequalities are added. These new valid inequalities are chosen using information about the objective of the optimization problem as well as the type of non-negative certificates wanted (see algorithms 2, 3 and 4). In particular Algorithm 1 may stop after a finite number of steps with the optimal value of (PP-P) without closely approximating the whole set $\mathcal{P}(S)$.

We refer to (PP-M$_s$) as the master problem. The subproblem consists of finding an improving inequality $p_s(x)$ or showing that none exists. Notice that while the master is a semidefinite problem of size polynomial in n and s, the subproblem is NP-hard. Thus, the complexity of the original problem (PP-P) is transferred to the subproblem and the iterative nature of DIGS. In the remainder of this paper, we consider practical versions of the generic DIGS (Algorithm 1). Section 2.1 introduces DIGS-A for general polynomial programs, and Section 2.2 proposes DIGS-A$^+$ for problems with non-negative variables.

Since in general the set $\mathcal{P}_{r_0}(S) \setminus \mathcal{K}^{r_0}_{G_s}$, cannot be efficiently represented or even closely approximated, DIGS-A and DIGS-A$^+$ generate polynomials $p_s(x)$ using approximations of this set. In other words, these practical procedures are obtained by relaxing the subproblem: instead of looking for inequalities that are guaranteed to be improving, we look for elements of the set $\mathcal{P}_{r_0}(S) \setminus \mathcal{K}^{r_0}_{G_s}$ that may be improving. There is in general no guarantee that a practical stopping criterion can always be satisfied. Moreover, the relaxed subproblems may not return a polynomial $p_s(x)$, and the practical algorithms may stop, even if optimality does not hold. However, these algorithms generate a sequence of improving lower bounds for (PP-P), and even in cases where optimality is not reached, better bounds and performance may be obtained in comparison to Lasserre's approach.

If (PP-P) has additional structure it may be possible to exploit this structure to obtain more efficient DIGSs. It may also be possible to prove for special cases that if no $p_s(x)$ is found by the relaxed subproblem then optimality is attained. For such cases, if the practical algorithm terminates, the computed solution is guaranteed to be optimal. We show in Section 3.1 that this holds for DIGS-B, a DIGS specialized for the case of binary PPs.

To illustrate the potential of the proposed scheme, we implemented the three practical DIGSs (Algorithms 2, 3, and 4) in Matlab and applied them to different types of PPs. To obtain a fair comparison, Lasserre's relaxation is implemented and solved using the same code.

2.1 DIGS-A: A General DIGS for all types of PPs

Given a finite $G \subset \mathbf{R}_{r_0}[x]$, we want to find a polynomial $p(x)$ of degree at most r_0 such that $p(x) \in \mathcal{P}_{r_0}(S) \setminus \mathcal{K}_G^{r_0}$. Assuming that $\mathcal{P}_{r_0}(S) \setminus \mathcal{K}_G^{r_0} \neq \emptyset$, the two issues to address are: first how to generate $p(x) \in \mathcal{P}_{r_0}(S)$; and second how to ensure $p(x) \notin \mathcal{K}_G^{r_0}$.

To tackle the first issue, since it is not possible in general to represent the set $\mathcal{P}_{r_0}(S)$ exactly, $\mathcal{P}_{r_0}(S)$ is approximated using $\mathcal{K}_G^r \cap \mathbf{R}_{r_0}[x]$ for some $r > r_0$. In particular, approximations of degree $r = r_0 + 1$ could be used and have $p(x) \in (\mathcal{K}_G^{r_0+1} \setminus \mathcal{K}_G^{r_0}) \cap \mathbf{R}_{r_0}[x]$. However, when r_0 is even, this might result in a very slow improvement in the bound of (PP-P) as $\Sigma_{r_0+1} = \Sigma_{r_0}$. Thus, we use $\mathcal{K}_G^{r_0+2} \cap \mathbf{R}_{r_0}[x]$ for even r_0 and $\mathcal{K}_G^{r_0+1} \cap \mathbf{R}_{r_0}[x]$ for odd r_0. For readability, we only use $\mathcal{K}_G^{r_0+2} \cap \mathbf{R}_{r_0}[x]$ in the sequel, independently of the parity of r_0.

To address the second issue, i.e, to ensure that $p(x) \notin \mathcal{K}_G^{r_0}$, we make use of the optimal dual solution of (5). We represent, by abuse of notation, $\mathbf{R}_{r_0}[x]$ with $\mathbb{R}^N$ where $N = \binom{n+r_0}{r_0}$, i.e., we identify each polynomial $f(x) \in \mathbf{R}_{r_0}[x]$ with its vector of coefficients $f \in \mathbb{R}^N$. In this way $\mathcal{K}_G^{r_0}$ is a cone in $\mathbb{R}^N$. We endow $\mathbb{R}^N$ with an inner product $\langle \cdot, \cdot \rangle$ such that for each $f(x) \in \mathbf{R}_{r_0}[x]$ and each $u \in \mathbb{R}^n$, $\langle f, \mathcal{M}_d(u) \rangle = f(u)$, where for a given u, $\mathcal{M}_d(u) = (u^\alpha)_{|\alpha| \leq d}$ is the vector of monomials of u up to degree d.

In this way, the relaxation (PP-M$_s$) and its semidefinite dual correspond to the conic primal-dual pair

$$\begin{array}{ll} \sup_\lambda \ \lambda & \inf_Y \ \langle f, Y \rangle \\ \text{s.t.} \quad f(x) - \lambda \in \mathcal{K}_G^{r_0} & \text{s.t.} \quad \langle 1, Y \rangle = 1 \\ & \qquad Y \in (\mathcal{K}_G^{r_0})^*, \end{array} \tag{3}$$

where $(\mathcal{K}_G^{r_0})^* = \{Y \in \mathbb{R}^N : \langle p, Y \rangle \geq 0 \text{ for all } p \in \mathcal{K}_G^{r_0}\}$. From the definition of the dual cone $(\mathcal{K}_G^{r_0})^*$, we have the following observation.

Remark 1 Let Y be a feasible solution of (3). For all $p(x) \in \mathcal{K}_G^{r_0}$, $\langle p, Y \rangle \geq 0$.

Thus to generate $p(x) \in \mathcal{P}_d(S) \setminus \mathcal{K}_G^{r_0}$, it suffices to find $p(x) \in \mathcal{K}_G^{r_0+2} \cap \mathbf{R}_{r_0}[x]$ such that $\langle p, Y \rangle < 0$, where Y is an optimal dual solution of (PP-M$_s$). This can be done by solving the following semidefinite problem. This problem is referred to as the *polynomial generating subproblem*:

$$\begin{aligned} \textbf{(PP-Sub)} \ \min_p \ & \langle p, Y \rangle \\ \text{s.t. } & p \in \mathcal{K}_G^{r_0+2} \cap \mathbf{R}_{r_0}[x] \\ & \| \, p \, \|_{Sub} \leq 1. \end{aligned} \tag{4}$$

The normalization constraint is added because otherwise (PP-Sub) is unbounded. Note that since $p(x)$ and $cp(x)$ are equivalent inequalities for any $c > 0$, any norm can be used.

There are several options for choosing $\| \cdot \|_{Sub}$. We use the pseudo-norm $\| \, p \, \|_{Sub} := \sum_{0 < \|\alpha\| \leq r_0} p_\alpha^2$ so that the optimal solution to (4) maximizes the ℓ_2 distance between the given Y and the set $\text{conv}\{\mathcal{M}_{r_0}(x) : p(x) \geq 0\}$ (see Proposition 3 and [11, pp. 80-82]). Summing up the ideas presented in this section, we obtain DIGS-A as presented in Algorithm 2.

As we already mentioned, a key element for the efficacy of the general scheme is the subproblem. Observe that for any $s \geq 0$, the optimization problem (PP-M$_s$) can be written as:

$$\begin{aligned} \nu_s = \sup_{\lambda, \sigma_i(x), \eta_i(x)} \ & \lambda \\ \text{s.t. } & f(x) - \lambda = \sum_{i=0}^{m} \sigma_i(x) g_i(x) + \sum_{i=1}^{s-1} \eta_i(x) p_i(x) \\ & \sigma_i(x) \in \Sigma_{r_0 - \deg(g_i)}, \qquad i = 0, \ldots, m \\ & \eta_i(x) \in \Sigma_{r_0 - \deg(p_i)}, \qquad i = 1, \ldots, s \end{aligned} \tag{5}$$

where $p_1(x), \ldots, p_{s-1}(x)$ are the polynomials generated in the iterations $0, \ldots, s-1$ of DIGS.

At iteration s of Algorithm 2, the size of (5) is:

- one psd matrix of size $\binom{n+r_0}{r_0}$;
- m psd matrices, each of size $\binom{n+r_0-\deg(g_i)}{r_0-\deg(g_i)}$ for $i = 1, \ldots, m$;
- s psd matrices, each of size $\binom{n+r_0-\deg(p_i)}{r_0-\deg(p_i)}$ for $i = 1, \ldots, s$;
- $\binom{n+r_0}{r_0}$ linear constraints.

Since r is increased when using (1) while r_0 is fixed in (5), the size of the positive semidefinite matrices and the number of constraints can be significantly lower in (5) compared to (1) because (1) has $m+1$ psd matrices of size $O(n^r)$ and $O(n^r)$ constraints while (5) has $m+s+1$ psd matrices of size $O(n^{r_0})$ and $O(n^{r_0})$ constraints. This difference is key to limiting the growth of the computational time required to solve (5); while the number of SDP matrices is fixed in (1) the size of the corresponding SDP matrices grows exponentially on r, however in (5) the size of the SDP matrices is fixed while their number grows linearly on s. In particular, if the generation procedure for $p_s(x)$ can ensure that $\deg p_s = r_0$ then $\eta_s(x) = \eta_s \in \mathbf{R}_+$ and the size of (5) grows slowly.

Algorithm 2 DIGS-A: DIGS for general polynomial programs

Require: G description of S, $r_0 \geq d$, $\epsilon > 0$
$s \leftarrow 0$, $G_0 \leftarrow G$.
loop
Let

$$\textbf{(PP-M}_s\textbf{)}\ \nu_s = \sup_{\lambda}\ \lambda \quad \text{s.t. } f(x) - \lambda \in \mathcal{K}^{r_0}_{G_s}.$$

$Y_s \leftarrow$ dual optimal solution to (PP-M$_s$).
Let $p_s(x) \in \mathbf{R}_{r_0}[x]$ be an optimal solution of

$$\alpha_s = \min_{p}\ \langle p, Y_s\rangle \quad \text{s.t. } p \in \mathcal{K}^{r_0+2}_{G_s} \cap \mathbf{R}_{r_0}[x] \quad \| p \| \leq 1.$$

if $\alpha_s < -\epsilon$ **then**
$G_{s+1} \leftarrow G_s \cup \{p_s(x)\}$
$s \leftarrow s + 1$
else
STOP
end if
end loop

2.1.1 Heuristic to find feasible solutions

Each time (PP-M$_s$) is solved and a dual solution Y_s is computed, we apply a heuristic to obtain a candidate solution $\hat{x}_s$ for (PP-P). If this candidate solution is feasible for (PP-P) then we obtain an upper bound ψ_s on ρ, and if this upper bound is close enough to ν_s then ϵ-optimality is achieved.

The heuristic to generate a candidate solution works as follows. Let Y_s be the optimal dual solution of (PP-M$_s$). Set $\hat{x}_s$ to be the entries of Y_s corresponding to the linear monomials. (This is a projection from the dual space to the original variables.) If the candidate solution $\hat{x}_s$ is feasible for (PP-P), i.e., if $g_i(\hat{x}_s) \geq 0$ for $i = 1, \ldots, m$ then the corresponding objective value $f(\hat{x}_s)$ is an upper bound for ρ. If some variables are constrained to be integer, the corresponding components of $\hat{x}$ are first rounded to the nearest integer to obtain the candidate solution.

This approach is illustrated in Example 1 below and in Examples 2 and 3 in Section 2.2.1.

2.1.2 Example

To illustrate how DIGS-A works, consider the following small quadratic problem:

Example 1

$$\begin{aligned} \max_{x}\ & -2x_1 + x_2 - x_3 + 2x_4 + 2x_5 \\ \text{s.t. }\ & (x_1-2)^2 - x_2^2 - (x_3-1)^2 - (x_5-1)^2 \geq 0 \\ & x_1x_3 - x_4x_5 + x_1^2 \geq 1 \\ & x_3 - x_2^2 - x_4^2 \geq 1 \\ & x_1x_5 - x_2x_3 \geq 2 \\ & x_1 + x_2 + x_3 + x_4 + x_5 \leq 14 \\ & 0 \leq x_i. \end{aligned} \tag{6}$$

Let $G = \{1, (x_1-2)^2 - x_2^2 - (x_3-1)^2 - (x_5-1)^2, x_1x_3 - x_4x_5 + x_1^2 - 1, x_3 - x_2^2 - x_4^2 - 1, x_1x_5 - x_2x_3 - 2, 14 - x_1 - x_2 - x_3 - x_4 - x_5, x_i\}$. Setting $s = 0$, $G_0 = G$ and solving the master problem (5), we obtain ν_0=25.000, which is an upper bound on (6), and the optimal dual solution

$$Y_0 = [1.0\ 0.0\ 0.0\ 1.0\ 0.0\ 13.0\ 195.2\ 0.0\ 40.4\ 0.0\ \ldots \\ \ldots\ 22.1\ 0.0\ 0.0\ 0.0\ 0.0\ 24.2\ 0.0\ 17.6\ 0.0\ 0.0\ 184.6].$$

By applying the heuristic, the following candidate solution is obtained

$$\hat{x}_0 = [0.0\ 0.0\ 1.0\ 0.0\ 13.0]$$

which is not feasible.

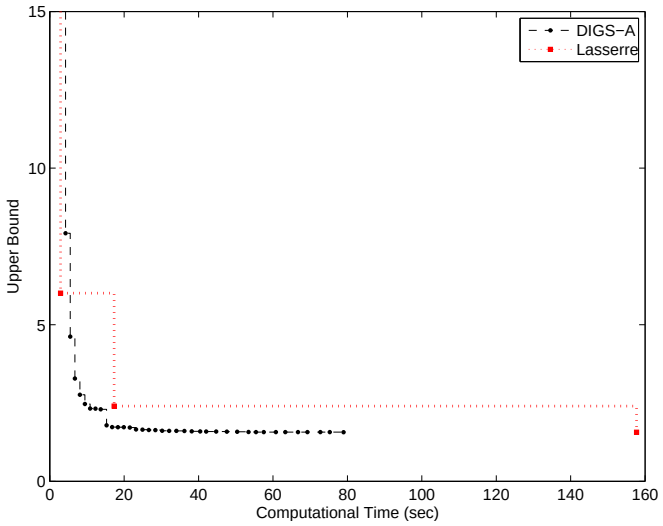

Fig. 1: DIGS-A and Lasserre upper bounds for Example 1.

Subproblem (PP-Sub) is solved with $Y = Y_0$, $G = G_0$, and $d = 2$. We obtain

$$\begin{aligned} p_0(x) =& 0.1490 + 0.3018x_1 + 0.0030x_2 + 0.1801x_3 + 0.0044x_4 + 0.0888x_5 \\ & - 0.0264x_1^2 + 0.0016x_1x_2 - 0.8893x_1x_3 + 0.0016x_1x_4 - 0.0992x_1x_5 + 0.0010x_2^2 \\ & + 0.0046x_2x_3 + 0.0003x_2x_4 + 0.0017x_2x_5 - 0.0699x_3^2 + 0.0072x_3x_4 + 0.1710x_3x_5 \\ & + 0.0018x_4^2 + 0.0018x_4x_5 - 0.1033x_5^2 \end{aligned}$$

which is a valid inequality for (6).

Setting $s = 1$ and $G_1 = G_0 \cup \{p_0\}$ and solving the master problem (5) with $s = 1$, we obtain the upper bound $\nu_1 = 24.98$. After two iterations ($s = 2$), the upperbound $\nu_2 = 5.3205$ is obtained. After 39 iterations, a master objective value (upper bound) of 1.567 and a subproblem value of -0.0008 are obtained. This upper bound is the optimal value of (6) since at iteration 39 the candidate solution extracted by the heuristic is

$$\hat{x}^{39} = \left[1.02\ 0.0\ 1.19\ 0.44\ 1.96\right],$$

which is a feasible solution to (6) with an objective value 1.5674.

A comparison between Lasserre's bounds and DIGS-A is presented in Figure 1. The results of DIGS-A and Lasserre's procedure are presented in Tables 1 and 2 respectively. The total time used by DIGS-A to obtain the optimal value was 78.9 seconds. The dimension and the computational time of the master problem and the subproblem are given in Table 1. Using Lasserre's approach, the optimal solution is obtained by solving a problem over $\mathcal{K}^8_{G_0}$. Notice that using DIGS-A we are able to obtain the optimal value while using cheaper low degree relaxations ($r = 2$ for the master problem and $r = 4$ for the subproblem).

Table 1: DIGS-A Results for Example 1.

Iteration	0	1	5	10	20	30	39
Subproblem							
psd matrices		6×6(10), 21×21(1)	6×6(14), 21×21(1)	6×6(19), 21×21(1)	6×6(29), 21×21(1)	6×6(39), 21×21(1)	6×6(48), 21×21(1)
non-negative vars		0	0	0	0	0	0
free vars		21	21	21	21	21	21
total # of vars		462	546	651	861	1071	1260
# of constraints		126	126	126	126	126	126
time (sec)		0.8	0.8	0.9	1.0	1.4	1.7
Master Problem							
psd matrices	6×6(1)	6×6(1)	6×6(1)	6×6(1)	6×6(1)	6×6(1)	6×6(1)
non-negative vars	10	11	15	20	30	40	49
total # of vars	31	34	36	41	51	61	70
# of constraints	21	21	21	21	21	21	21
time (sec)	0.3	0.5	0.5	0.6	0.6	0.8	1.0
bound	25.000	24.988	3.281	2.299	1.612	1.575	1.567
accumulated time (sec)	0.3	1.6	6.8	13.7	30.2	53.4	78.9

Table 2: Lasserre's Hierarchy for Example 1.

r	2	4	6	8
psd matrices	6×6(1)	6×6(10), 21×21(1)	21×21(10), 56×56(1)	56×56(10), 126×126(1)
non-negative vars	10	0	0	0
total # of vars	31	441	3906	23961
# of constraints	21	126	462	1287
bound	25.000	6.006	2.399	1.567
time (sec)	0.3	2.9	17.3	157.7

2.2 DIGS-A$^+$: DIGS for polynomial programs with non-negative variables

We now consider the case in which the feasible set is contained in the non-negative orthant. Let $\mathbf{R}^+_d[x]$ denote the cone of polynomials in n variables with non-negative coefficients of degree at most d. Lasserre's relaxations can be applied replacing the cone Σ_d of SOS by the cone $\Sigma_d + \mathbf{R}^+_d[x]$ of polynomials that can be represented as a sum of squares plus a polynomial with non-negative coefficients.

For $S = \{x : g_i(x) \geq 0,\, i = 1, \ldots, m\} \subseteq \mathbf{R}^n_+$, consider the following approximation of $\mathcal{P}(S)$:

$$\mathcal{CP}^r_G = \sum_{i=0}^{m} g_i(x)(\Sigma_{r-\deg(g_i)} + \mathbf{R}^+_{r-\deg(g_i)}[x]). \tag{7}$$

The corresponding optimization problem, (CP-M$_s$), over S can be written as:

$$\begin{aligned} \nu_s = \ & \sup \lambda \\ & \text{s.t. } f(x) - \lambda = \sum_{i=0}^{m}(\sigma_i(x) + \gamma_i(x))g_i(x) + \sum_{i=1}^{s}(\eta_i(x) + \epsilon_i(x))p_i(x) \\ & \quad \sigma_i(x) \in \Sigma_{r_0-\deg(g_i)},\ \gamma_i(x) \in \mathbf{R}^+_{r-\deg(g_i)}[x] \qquad i = 0, \ldots, m \\ & \quad \eta_i(x) \in \Sigma_{r_0-\deg(p_i)},\ \epsilon_i(x) \in \mathbf{R}^+_{r-\deg(p_i)}[x] \qquad i = 1, \ldots, s. \end{aligned} \tag{8}$$

If (PP-P) has m inequality constraints at iteration s of Algorithm 1, the size of (8) is:

- $m + 1$ psd matrices, each of size $\binom{n+r_0-\deg(g_i)}{r_0-\deg(g_i)}$ for $i \in \{0, 1, \ldots, m\}$;
- s psd matrices, each of size $\binom{n+r_0-\deg(p_i)}{r_0-\deg(p_i)}$ for $i \in \{1, \ldots, s\}$;
- $\sum_{i=0}^{m}\binom{n+r_0-\deg(g_i)}{r_0-\deg(g_i)} + \sum_{i=1}^{s}\binom{n+r_0-\deg(p_i)}{r_0-\deg(p_i)}$ non-negative variables;
- $\binom{n+r_0}{r_0}$ linear constraints.

The subproblem is of the form

$$\begin{aligned} \textbf{(CP-Sub)} \ \min_p \ & \langle p, Y \rangle \\ & \text{s.t. } (1 + \sum_{i=1}^{n} x_i)p(x) \in \mathcal{CP}^{r_0+1}_G \cap \mathbf{R}_{r_0}[x] \\ & \quad \| \, p \, \| \leq 1. \end{aligned} \tag{9}$$

We call the resulting scheme DIGS-A$^+$ and present it as Algorithm 3.

DIGS-A$^+$ is computationally more efficient than DIGS-A. Even though the master problem for DIGS-A$^+$ is larger due to the non-negative variables, at each iteration subproblem (9) has $\frac{n+r_0+1}{r_0+1}$ times the number of variables and $\frac{n+r_0+1}{r_0+1}$ times the number of constraints of the master problem. This is much smaller than subproblem (4) in DIGS-A.

2.2.1 Examples

We present four examples comparing the bounds obtained using DIGS-A, DIGS-A$^+$ and Lasserre's hierarchy. For DIGS-A and DIGS-A$^+$ we stop when the objective value of the subproblem is greater than $-\epsilon = -10^{-3}$. All the algorithms are given a time limit of 5 hours (18000 seconds).

Algorithm 3 DIGS-A$^+$:DIGS for polynomial programs with non-negatives variables

Require: G description of $S \subseteq \mathbb{R}^+$, $r_0 \geq d$, $\epsilon > 0$
$s \leftarrow 0$, $G_0 \leftarrow G$.
loop
Let

$$\textbf{(CP-M}_s\textbf{)}\ \nu_s = \sup_{\lambda}\ \lambda \quad \text{s.t. } f(x) - \lambda \in \mathcal{CP}^{r_0}_{G_s}.$$

$Y_s \leftarrow$ dual optimal solution to (CP-M$_s$).
Let $p_s(x) \in \mathbf{R}_{r_0}[x]$ be optimal solution of

$$\alpha_s = \min_{p}\ \langle p, Y_s \rangle \quad \text{s.t. } (1 + \sum_{i=1}^{n} x_i)p(x) \in \mathcal{CP}^{r_0+1}_{G_s} \cap \mathbf{R}_{r_0}[x] \quad \| p \| \leq 1.$$

if $\alpha_s < -\epsilon$ **then**
$G_{s+1} \leftarrow G_s \cup \{p_s(x)\}$
$s \leftarrow s + 1$
else
STOP
end if
end loop

Example 2
Consider the following polynomial program of degree 2:

$$\begin{aligned}
\max_{x\in\mathbb{R}}\quad & x_1 + x_2 - x_3 + 2x_4 + x_5 - x_6 - x_7 + x_8 - x_9 + 2x_{10} \\
\text{s.t. } & (x_3 - 2)^2 - (x_5 - 1)^2 - 2x_6 + x_8^2 - (x_9 - 2)^2 \geq -4 \\
& - x_2^2 + x_3x_{10} - x_4^2 - x_5^2 + x_6x_7 \geq 1 \\
& x_1x_8 - x_2x_3 + x_4x_7 - x_5x_{10} \geq 2 \\
& \sum_{i=1}^{10} x_i \leq 5 \\
& x_i \geq 0 \qquad \forall i \in \{1, \ldots, 10\}.
\end{aligned}$$

Tables 3a, 3b, and 3c present bounds and computational time for Lasserre's method, DIGS-A ($r_0 = 2$), and DIGS-A$^+$ ($r_0 = 2$) respectively. Figure 2 illustrates the bound improvements.

Table 3: Bounds for Example 2.

(a) Lasserre's method

r	2	4	6
bound	10.000	7.756	5.183*
T(sec)	0.2	35.8	4012.7

(b) DIGS-A

Iter.	0	1	5	9
bound	10.000	9.992	5.276	5.183*
T(sec)	0.2	15.9	66.2	114.9

(c) DIGS-A$^+$

Iter.	0	1	5	9
bound	7.760	5.749	5.192	5.183*
T(sec)	0.3	2.4	9.4	16.3

Values with $*$ indicate termination reporting optimality.

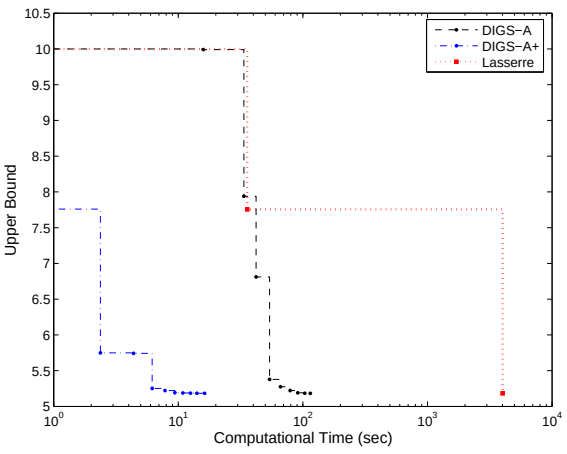

Fig. 2: Bounds comparison for Example 2.

All three approaches find the optimal value 5.183. For $r = 6$ Lasserre's method stops with the moment matrix optimality condition satisfied, obtaining the optimal solution in just over one hour.

Using DIGS-A and DIGS-A$^+$ with $r_0 = 2$, degree-two relaxations are used to obtain the optimal value in much less computational time. Both require the same number of iterations, but DIGS-A$^+$ is seven times faster than DIGS-A and 250 times faster than Lasserre's $r = 6$ relaxation. For DIGS-A, at iteration 9, the heuristic finds the solution $\hat{x}_1 = \hat{x}_8 = 1.4143$, $\hat{x}_3 = 0.6628$, $\hat{x}_{10} = 1.5088$, and $\hat{x}_2 = \hat{x}_4 = \hat{x}_5 = \hat{x}_6 = \hat{x}_7 = \hat{x}_9 = 0$ which is optimal (tolerance of 10^{-3}). For DIGS-A$^+$, also at iteration 9, the heuristic finds the solution $\hat{x}_1 = 1.4137$, $\hat{x}_3 = 0.6627$, $\hat{x}_8 = 1.4146$, $\hat{x}_{10} = 1.5089$, and $\hat{x}_2 = \hat{x}_4 = \hat{x}_5 = \hat{x}_6 = \hat{x}_7 = \hat{x}_9 = 0$ which is also optimal (tolerance of 10^{-3}).

Example 3
Consider the following polynomial program of degree 2:

$$
\begin{aligned}
\max_{x\in\mathbb{R}} \quad & -2x_1 + x_2 - x_3 + 2x_4 + x_5 - x_6 - x_7 + x_8 - x_9 + 2x_{10} \\
\text{s.t. } & (x_1-2)^2 - x_2^2 - (x_7-2)^2 - (x_5-1)^2 + (x_6-2)^2 + x_{10}^2 \geq 0 \\
& x_4^2 - x_1^2 - (x_3-1)^2 - (x_8-2)^2 + (x_9-2)^2 + (x_{10}-2)^2 \geq 0 \\
& x_3x_8 - x_4x_5 + x_1^2 + x_6x_9 + x_1x_7 \geq 1 \\
& \sum_{i=1}^{10} x_i \leq 10 \\
& x_i \geq 0 \qquad \forall i \in \{1,\ldots,10\}.
\end{aligned}
$$

Tables 4a, 4b, and 4c present bounds and computational time for Lasserre's hierarchy, DIGS-A ($r_0 = 2$), and DIGS-A$^+$ ($r_0 = 2$) respectively. Figure 3 illustrates the bound improvements.

Table 4: Upper bounds for Example 3.

(a) Lasserre's method

r	2	4	6	8
bound	20.00	19.59	16.92	-
T(sec)	0.2	40.6	17350.0	-

(b) DIGS-A

Iter.	0	1	5	10	64
bound	20.00	19.73	18.15	16.92	16.51*
T(sec)	0.2	11.2	70.9	149.4	1353.1

(c) DIGS-A$^+$

Iter.	0	1	5	10	100	156
bound	19.60	18.75	18.62	18.16	16.62	16.53
T(sec)	0.3	2.3	10.6	20.3	237.5	416.8

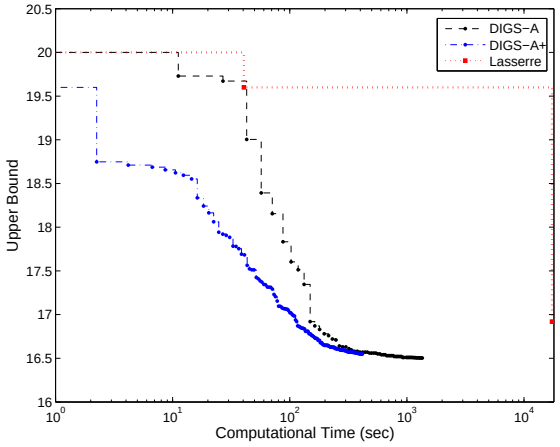

Fig. 3: Bounds comparison for Example 3.

For this example, Lasserre's hierarchy does not find the optimal value in the five-hour limit. It needs $r > 6$ to obtain the optimal value, but solving the $r = 6$ relaxation requires nearly 5 hours of computational time and for $r = 8$ even constructing the problem takes more than 5 hours.

Using DIGS-A and DIGS-A$^+$, we are able to use relaxations of degree $r_0 = 2$ and obtain improved bounds in less computational time. While Lasserre's $r = 6$ relaxation provides the bound 16.92 in almost 5 hours, DIGS-A and DIGS-A$^+$ both provide the same bound after less than 5 minutes.

DIGS-A terminates in 64 iterations and less than 23 minutes. The heuristic finds the solution $\hat{x}_2 = 0.008$, $\hat{x}_3 = 0.557$, $\hat{x}_4 = 2.277$, $\hat{x}_8 = 1.808$, $\hat{x}_{10} = 5.349$, and $\hat{x}_1 = \hat{x}_5 = \hat{x}_6 = \hat{x}_7 = \hat{x}_9 = 0$ which is optimal (tolerance of 10^{-3}). DIGS-A$^+$ terminates in less than 7 minutes after 156 iterations but with no feasible solution found by the heuristic, and thus we cannot conclude ϵ-optimality of the bound.

DIGS-A$^+$ performed in this case more than double the number of iterations than DIGS-A but remained three times faster.

Example 4
Consider the following polynomial program of degree 2:

$$\begin{aligned}
\max_{x\in\mathbb{R}} \quad & -x_1+x_2-x_3+x_4+x_5-x_6-x_7+x_8-x_9+x_{10}-x_{11}+x_{12}-x_{13}+x_{14}-x_{15} \\
\text{s.t. } & (x_1-2)^2-x_2^2+(x_3-2)^2-(x_4-1)^2-(x_5-1)^2+(x_6-1)^2-(x_7-2)^2-x_8^2 \\
& \quad -(x_9-2)^2-(x_{10}-1)^2+x_{11}^2-x_{12}^2+(x_{13}-2)^2+x_{14}^2-(x_{15}-1)^2\geq 0 \\
& -x_1x_7-x_4x_5-x_{13}^2+x_6x_9+x_{10}x_{12}\geq 3 \\
& x_2x_3-x_8x_{11}-x_{14}^2+x_5x_{15}\geq 3 \\
& \sum_{i=1}^{15} x_i \leq 10 \\
& x_i\geq 0 \qquad \forall i\in\{1,\dots,15\}.
\end{aligned}$$

Table 5a presents Lasserre's Hierarchy results. As in the previous examples, quadratic inequalities ($r_0 = 2$) are added using DIGS-A and DIGS-A$^+$ to improve the bound. Tables 5b and 5c present the bounds and computational time for DIGS-A and DIGS-A$^+$ respectively.

Table 5: Upper bounds for Example 4.

(a) Lasserre's method

r	2	4	6
bound	10.00	8.0593	-
T(sec)	0.3	2754.3	-

(b) DIGS-A

Iter.	0	1	5	10	18
bound	10.00	10.00	7.93	7.58	7.43*
T(sec)	0.3	717.8	3128.0	5720.1	10236.1

(c) DIGS-A$^+$

Iter.	0	1	5	7
bound	8.07	7.67	7.44	7.43*
T(sec)	0.6	6.7	33.4	43.9

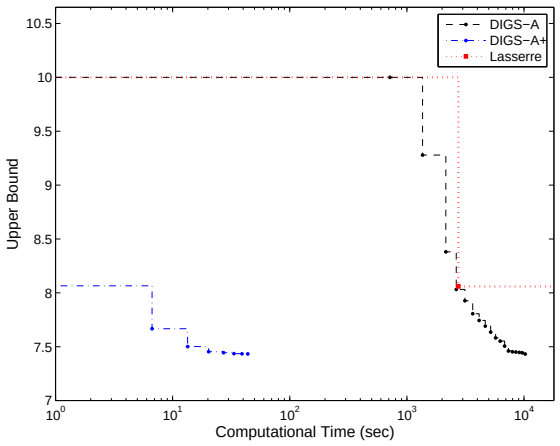

Fig. 4: Bounds comparison for Example 4.

This example illustrates how the dimension affects the three methods. Lasserre's relaxations could not be solved for $r > 4$ due to memory and time limitations. Both DIGS-A and DIGS-A$^+$ obtain better bounds than Lasserre's $r = 4$ relaxation, and DIGS-A$^+$ outperforms DIGS-A (see Figure 4). Indeed DIGS-A$^+$ is three orders of magnitude faster than DIGS-A and nevertheless both DIGS-A and DIGS-A$^+$ terminate (after 7 and 18 iterations respectively) with the optimal objective value of 7.43 and ϵ-optimal heuristic solution $\hat{x}_1 = \hat{x}_3 = \hat{x}_6 = \hat{x}_7 = \hat{x}_9 = \hat{x}_{13} = 0.0000$, $\hat{x}_2 = 0.5881$, $\hat{x}_4 = 0.3344$, $\hat{x}_5 = 2.6244$, $\hat{x}_8 = 0.5673$, $\hat{x}_{10} = 2.3443$, $\hat{x}_{11} = 0.0008$, $\hat{x}_{12} = 1.6538$, $\hat{x}_{14} = 0.6047$, and $\hat{x}_{15} = 1.2822$.

Example 5
This is an example where DIGS-A$^+$ fails to converge to the optimal solution of the polynomial program. The PP is a formulation of the maximum stable set number for the icosahedron graph.

Given an undirected graph $G = (V, E)$, a stable set of G is a set of vertices $U \subseteq V$ such that there is no edge connecting any two vertices in U. The stable set number problem is to find $\alpha(G)$ the maximum k for which the graph has a stable set of cardinality k. Letting $n = |V|$, and identifying V with $\{1,\dots,n\}$, the maximum stable set problem can be formulated as follows:

$$\begin{aligned}
\textbf{(SS-POP)} \quad \alpha(G) = \max_x \quad & \left(\sum_{i=1}^n x_i\right)^2 \\
\text{s.t. } & x_ix_j = 0 \qquad \forall (i,j)\in E \\
& \sum_{i=1}^n x_i^2 = 1 \\
& x\geq 0.
\end{aligned} \tag{10}$$

Notice that Lasserre's relaxation of (SS-POP) for $r = 2$ is equivalent to the Lovász theta relaxation. Thus, DIGS-A and DIGS-A$^+$ can be interpreted here as adding quadratic valid inequalities to strengthen the Lovász theta relaxation.

We consider the case in which G is the icosahedron graph on $n = 12$ vertices (see Figure 5). Applying DIGS-A$^+$ results in an objective function of 3.236 at the initial iteration, and the algorithm immediately stops because the subproblem objective function value is of order -10^{-8}. However, using Lasserre's hierarchy for $r = 4$ gives the optimal bound 3.000. DIGS-A terminates in 16 iterations with the bound 3.002.

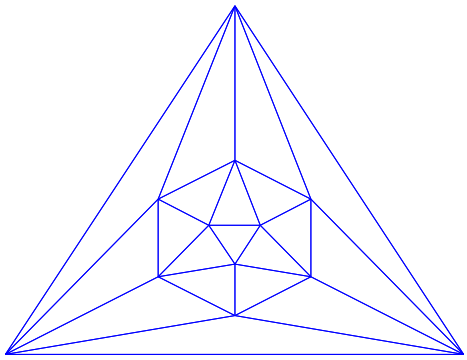

Fig. 5: Icosahedron graph for Example 5.

Table 6: Bounds for Example 5.

(a) Lasserre's method

r	2	4	6
bound	3.708	3.000	-
T(sec)	0.1	94.8	-

(b) DIGS-A

Iter.	0	1	5	10	16
bound	3.708	3.164	3.002	3.002	3.002
T(sec)	0.1	31.9	209.2	547.9	1057.5

(c) DIGS-A$^+$

Iter.	0	1
bound	3.2361	3.2361
T(sec)	0.1	2.0

3 The Special Case of Binary Polynomial Programs

This section focus on the special case of polynomial programs with binary variables. We specialize Algorithm 1 to this class of problems and propose a specialized DIGS called DIGS-B. The main improvement in DIGS-B is the use of a computationally cheaper subproblem for the binary case. The subproblem is obtained using the approach proposed in [28] and [37]. Furthermore, we show that if the initial set of constraints G is *rich enough*, the stopping criterion is optimal, that is, DIGS-B only stops when optimality is reached. Theorems 1 and 2 show that DIGS-B terminates only at optimality when starting from the exact representation of the domain set excluding the binary constraints. Such a representation is not tractable in general, but these results hint that if our approximation $\mathcal{Q}_G$ of $\mathcal{P}_d(S \cap \{-1,1\}^n)$ captures enough of $\mathcal{P}_d(S)$, then the optimality criterium could be applied. Recall that we use a heuristic approach (see Section 2.1.1) to check optimality. The advantage of this heuristic approach is that when successful it also generates an optimal solution. Theorem 3 shows that a LP-variation of DIGS-B always stops after a finite number of iterations.

Several techniques have been developed to construct hierarchies of linear and semidefinite relaxations to solve binary linear programs; in particular, by Balas, Ceria, and Cornuéjols [3], Sherali and Adams [32], and Lovász and Schrijver [19]. The method presented in this section can be seen as a generalization of the lift-and-project methods based on these hierarchies. For the pure binary case, Lasserre [14] presents necessary and sufficient conditions under which his method converges to the optimal objective value of the 0-1 polynomial program after a finite sequence of liftings. A comparison of these different schemes and bounds on the number of steps required for convergence is given by Laurent [17].

3.1 DIGS-B: DIGS for polynomial programs with binary variables

We assume that the feasible set of (PP-P) has the form $S = D \cap \{-1,1\}^n$ where $D = \{x : g_i(x) \geq 0,\ i = 1,\ldots,m\}$. We further assume without loss of generality that $D \subseteq [-1,1]^n$; this can be ensured for example by adding the inequalities $-1 \leq x_i \leq 1$ to the definition of D. Problem (PP-D) becomes

$$\begin{aligned} \rho \ = \ & \sup_{\lambda} \lambda \\ & \text{s.t. } f(x) - \lambda \in \mathcal{P}_d(D \cap \{-1,1\}^n). \end{aligned} \tag{11}$$

Proceeding similarly as in Section 2, we let $G = \{g_0(x), g_1(x), \ldots, g_m(x)\}$ where $g_0(x) = 1$. Define $\mathcal{Q}_G^r$ as the following approximation to $\mathcal{P}_d(S)$:

$$\mathcal{Q}_G^r = \sum_{i=0}^{|G|} g_i(x)\Sigma_{r-\deg(g_i)} + \sum_{i=1}^{n}(1-x_i^2)\mathbf{R}_{r-2}[x]$$

Fix $r_0 \geq d$ and define the polynomial programming master problem

$$\begin{aligned} \varphi_G^{r_0} &= \sup_{\lambda} \lambda \\ &\text{s.t. } f(x) - \lambda \in \mathcal{Q}_G^{r_0}, \end{aligned} \tag{12}$$

which can be written as

$$\begin{aligned} \varphi_G^{r_0} = \sup_{\lambda, \sigma_i(x), \delta_i(x)} \quad & \lambda \\ \text{s.t.} \quad & f(x) - \lambda = \sum_{i=0}^{m} \sigma_i(x) g_i(x) + \sum_{i=1}^{n} \delta_i(x)(1 - x_i^2). \\ & \sigma_i(x) \in \Sigma_{r_0 - \deg(g_i)}, \qquad i = 0, 1, \ldots, m \\ & \delta_i(x) \in \mathbf{R}_{r_0 - 2}[x], \qquad i = 1, \ldots, n. \end{aligned}$$

Let $H_j = \{x \in \mathbb{R}^n : x_j \in \{-1, 1\}\}$ and $H = \{-1, 1\}^n$. Notice that $H = \bigcap_{j \in \{1,\ldots,n\}} H_j$. Instead of solving the polynomial generating subproblem over $\mathcal{Q}_G^{r_0+2}$ as defined in Section 2.1, we use Proposition 2 to obtain a computationally-cheaper polynomial generating subproblem.

Proposition 2 *[28] For any degree d and compact set D,*

$$\mathcal{P}_d(D \cap H_j) = \Big((1 + x_j)\mathcal{P}_d(D) + (1 - x_j)\mathcal{P}_d(D) + (1 - x_j^2)\mathbf{R}_{d-1}[x]\Big) \cap \mathbf{R}_d[x].$$

From Proposition 2, defining for any $\mathcal{Q} \subseteq \mathbf{R}_{r_0}[x]$,

$$\mathcal{C}_j^{r_0}(\mathcal{Q}) := \frac{(1 + x_j)}{2}\mathcal{Q} + \frac{(1 - x_j)}{2}\mathcal{Q} + (1 - x_j^2)\mathbf{R}_{r_0-1}[x],$$

we have $\mathcal{C}_j^{r_0}(\mathcal{P}_d(D)) \cap \mathbf{R}_d[x] = \mathcal{P}_d(D \cap H_j)$ for any $r_0 \geq d$ and any compact set D. Another important property of the operator $\mathcal{C}_j^{r_0}$ is that if $\mathcal{Q} \subsetneq \mathcal{P}_d(S)$ then $(\mathcal{C}_j^{r_0}(\mathcal{Q}) \setminus \mathcal{Q}) \cap \mathbf{R}_d[x] \neq \emptyset$ for some j (see Lemma 3). This property is critical to obtain a more efficient DIGS for the binary case: when looking for improving inequalities, $\mathcal{C}_j^{r_0}(\mathcal{Q}_G^{r_0})$ can be used instead of $\mathcal{Q}_G^{r_0+2}$ in the definition of Subproblem (PP-Sub), resulting in a much smaller subproblem.

Let Y be the optimal dual variable for (12), then the j-th polynomial generating subproblem for the binary case is defined as

$$\begin{aligned} &\min_{p} \ \langle p, Y \rangle \\ &\text{s.t. } p \in \mathcal{C}_j^{r_0}(\mathcal{Q}_G^{r_0}) \cap \mathbf{R}_{r_0}[x] \\ &\qquad \| p \| \leq 1. \end{aligned} \tag{13}$$

Replacing the master problem and subproblem in Algorithm 2 by (12) and (13) respectively, we obtain DIGS-B (Algorithm 4).

DIGS-B terminates when the subproblem (13) has objective value equal to zero for all indexes j. In practice, the algorithm stops when the value of every subproblem is sufficiently close to zero.

For the binary case, DIGS-B is a priori computationally more efficient than DIGS-A. The master problems are of the same size but solving the subproblem (13) is of the same order as solving the master problem (12), since at each iteration, (13) has twice the number of variables and $\frac{n+r_0+1}{r_0+1}$ times the number of constraints of the master problem. This is significantly smaller than subproblem (4) for the general case which has $O(n^2/r_0^2)$ times the number of variables and $O(n^2/r_0^2)$ times the number of constraints compared to the master problem.

3.2 Optimality and Finiteness of DIGS-B

In general we do not expect our algorithms to stop, or even if they do, we do not expect them to always provide the optimal value (see example 5). In this section, we prove (see Theorems 1 and 2) that under some assumptions DIGS-B certifies optimality for binary polynomial programs. Optimality is certified in the sense that if DIGS-B stops, i.e., if all sub-problems have optimal value zero, then the bound provided by DIGS-B is equal to the optimal value of (11). We also prove (Theorem 3) finite termination for DIGS-B* (see Algorithm 5), a variation of DIGS-B.

Algorithm 4 DIGS-B: DIGS for binary polynomial problems

Require: G description of S, $r_0 \geq d$, $\epsilon > 0$
$s \leftarrow 0$, $G_0 \leftarrow G$.
loop
Let

$$\textbf{(BP-M}_s\textbf{)} \quad \nu_s = \sup_{\lambda} \lambda \quad \text{s.t. } f(x) - \lambda \in \mathcal{Q}^{r_0}_{G_s}.$$

$Y_s \leftarrow$ dual optimal solution to (BP-M$_s$).
LET $p_s(x) \in \mathbf{R}_{r_0}[x]$ be an optimal solution of

$$\alpha_s = \min_{j=1,\ldots,n} \min_{p} \langle p, Y_s \rangle \quad \text{s.t. } p \in \mathcal{C}_j(\mathcal{Q}^{r_0}_{G_s}) \cap \mathbf{R}_{r_0}[x] \quad \| p \| \leq 1,$$

if $\alpha_s < -\epsilon$ **then**
$G_{s+1} \leftarrow G_s \cup \{p_s(x)\}$
$s \leftarrow s + 1$
else
STOP
end if
end loop

First we discuss the properties of the operator $\mathcal{C}^d_j$ that will be key for our results.

Lemma 3 *Let $D \subset [-1,1]^n$ and let $\mathcal{Q} \subseteq \mathcal{P}_d(D \cap H)$.*

1. *For every j and $r_0 \geq d$, $\mathcal{Q} \subseteq \mathcal{C}^{r_0}_j(\mathcal{Q}) \cap \mathbf{R}_d[x] \subseteq \mathcal{P}_d(D \cap H)$,*
2. *If $\mathcal{P}_d(D) \subseteq \mathcal{Q}$ then $\mathcal{P}_d(D \cap H_j) \subseteq \mathcal{C}^{r_0}_j(\mathcal{Q}) \cap \mathbf{R}_d[x]$,*
3. *Moreover, if $\mathcal{P}_d(D) \subseteq \mathcal{Q} \subsetneq \mathcal{P}_d(D \cap H)$ then for some j, $\mathcal{Q} \subsetneq \mathcal{C}^{r_0}_j(\mathcal{Q}) \cap \mathbf{R}_d[x]$.*

Proof

1. From the definition of $\mathcal{C}^{r_0}_j$, $\mathcal{Q} \subseteq \frac{(1-x_j)}{2}\mathcal{Q} + \frac{(1+x_j)}{2}\mathcal{Q} \subseteq \mathcal{C}^{r_0}_j(\mathcal{Q})$. On the other hand if $\mathcal{Q} \subseteq \mathcal{P}_d(D \cap H)$ then $\frac{(1-x_j)}{2}\mathcal{Q}, \frac{(1+x_j)}{2}\mathcal{Q} \subseteq \mathcal{P}_{d+1}(D \cap H)$ and thus $\mathcal{C}^{r_0}_j(\mathcal{Q}) \subseteq \mathcal{P}_{r_0+1}(D \cap H)$.
2. Follows from Proposition 2.
3. For the sake of contradiction, assume $\mathcal{Q} = \mathcal{C}^{r_0}_j(\mathcal{Q}) \cap \mathbf{R}_d[x]$ for all j. Applying part 2 for all $i = 1, \ldots, n$ we have $\mathcal{P}_d(D \cap \bigcap_{j \leq i} H_j) \subseteq \mathcal{C}^{r_0}_i(\mathcal{P}_d(D \cap \bigcap_{j \leq i-1} H_j)) \cap \mathbf{R}_d[x]$ and by induction it follows that $\mathcal{P}_d(D \cap \bigcap_{j \leq i} H_j) = \mathcal{Q}$ for all $i = 1, \ldots, n$. In particular $\mathcal{P}_d(D \cap H) \subseteq \mathcal{Q}$.

□

Recall from (3) that the optimal dual solution $Y \in \{X \in (\mathcal{Q}^d_G)^* : \langle 1, X \rangle = 1\}$. When D is compact, the set $\mathcal{P}_d(D)^*$ can be characterized in terms of truncated moment matrices of atomic measures, using classical theorems from [**?**] and [**?**] (see [**?**, Thms 5.9 and 5.13]). Using the compactness of D implies that $\mathcal{P}_d(D)^*$ is pointed, $\{\mathcal{P}_d(D)^* : \langle 1, X \rangle = 1\}$ could be then characterized in terms of truncated moment matrices of atomic probability measures, which are themselves convex combinations of truncated moment matrices of Dirac measures. Truncated moment matrices of Dirac measures are in correspondence with $\mathcal{M}_d(u)$, *the vector of monomials of u up to degree d* for some u. All this is subsumed in Proposition 3, where $\{\mathcal{P}_d(D)^* : \langle 1, X \rangle = 1\}$ is characterized as the convex hull of $\mathcal{M}_d(D) = \{\mathcal{M}_d(u) : u \in D\}$, for compact D. A more elemental, self-contained proof is also given.

Proposition 3 *Let $D \subseteq \mathbb{R}^n$ be a compact set, then*

$$\{X \in \mathcal{P}_d(D)^* : \langle 1, X \rangle = 1\} = \text{conv}(\mathcal{M}_d(D)).$$

Proof By definition,

$$\begin{aligned} \mathcal{M}_d(D)^* &= \{p : \langle p, X \rangle \geq 0 \, \forall X \in \mathcal{M}_d(D)\} \\ &= \{p : \langle p, \mathcal{M}_d(s) \rangle \geq 0 \, \forall s \in D\} \\ &= \{p : p(s) \geq 0 \, \forall s \in D\} \\ &= \mathcal{P}_d(D). \end{aligned}$$

Therefore, $\mathcal{P}_d(D)^* = \mathcal{M}_d(D)^{**} = \text{closure}(\text{cone}(\text{conv}(\mathcal{M}_d(D))))$. Since D is compact, $\mathcal{M}_d(D)$ is compact. Also, for all $x \in D$, $\langle 1, M_d(x) \rangle = 1$ and thus

$$\{X \in \mathcal{P}_d(D)^* : \langle 1, X \rangle = 1\} = \{X \in \mathcal{M}_d(D)^{**} : \langle 1, X \rangle = 1\} = \text{conv}(\mathcal{M}_d(D)).$$

□

The set $\operatorname{conv}\mathcal{M}_2(\{-1,1\}^n)$ is closely related to $\operatorname{conv}\{xx^T : x \in \{-1,1\}^n\}$, the *Boolean Quadratic Polytope* [8,9]. More generally $\operatorname{conv}\{\mathcal{M}_d(u) : u \in D\}$ is a polytope for any finite D. We will use this fact later in the proof of Theorem 3.

Lemma 4 *Let $d \geq 1$ and let D be a finite set. Then $\operatorname{conv}\mathcal{M}_d(D)$ is a polytope, and $\mathcal{P}_d(D)$ is a polyhedral cone.*

3.2.1 Optimality

In this section we prove optimality for DIGS-B under certain conditions. More precisely we show that if the starting set G is such that the approximation $\mathcal{Q}_G^{r_0}$ contains $\mathcal{P}_d(D)$, DIGS-B only stops if the bound given by the master problem is optimal. If $\mathcal{Q}_G^{r_0} \supseteq \mathcal{P}_d(D)$ then subproblem (13) is equivalent to optimizing over a set containing $\mathcal{P}_d(D \cap H_j)$ (by Lemma 3, part 2). Intuitively, if the subproblem has a value of 0, it is because restricting x_j to be binary does not change the objective value. Thus if the value of all subproblems is zero, then the algorithm has converged to the optimal value. This intuition is formally expressed in Theorems 1 and 2.

Theorem 1 *Let $d \geq 2$. Suppose that $D \subseteq [-1,1]^n$ is a compact set, and that $r_0 \geq d$ and $G := \{g_i(x) : i = 1,\ldots,m\} \subset \mathbf{R}[x]$ are such that $\mathcal{P}_d(D) \subseteq \mathcal{Q}_G^{r_0} \cap \mathbf{R}_d[x] \subseteq \mathcal{P}_d(D \cap \{-1,1\}^n)$. If the optimal value of subproblem (13) is zero for all $j = 1,\ldots,n$, then the optimal value of the dual problem of the master (12) equals the optimal value of the original binary polynomial program (11).*

Proof Let $\mathcal{Q} = \mathcal{Q}_G^{r_0} \cap \mathbf{R}_d[x]$. Let ρ be the optimal value of (11) and ω_j be the optimal value of the j-th subproblem (13). Further let φ be the optimal value and Y be an optimal solution of the dual problem of (12). Since $\mathcal{Q} \subseteq \mathcal{P}_d(D \cap \{-1,1\}^n)$ then $\varphi \leq \rho$. By Proposition 3,

$$Y \in \{X \in \mathcal{Q}^* : \langle 1, X\rangle = 1\} \subseteq \{X \in \mathcal{P}_d(D)^* : \langle 1, X\rangle = 1\} = \operatorname{conv}(\mathcal{M}_d(D)).$$

As D is compact, $\operatorname{conv}(\mathcal{M}_d(D))$ is compact, and by Caratheodory's Theorem, Y can be expressed in the form $Y = \sum_i a_i \mathcal{M}_d(u_i)$ with $a_i > 0$, $\sum_i a_i = 1$ and every $u_i \in D$. It follows that for any $p \in \mathbf{R}_d[x]$ we have $\langle p, Y\rangle = \sum_i a_i \langle p, \mathcal{M}_d(u_i)\rangle = \sum_i a_i p(u_i)$. It is enough to show that $u_i \in H$ for all i, as then $\varphi = \langle f, Y\rangle = \sum_i a_i f(u_i) \geq \rho$ and we are done. Assume, there exists $u_k \notin H$ for some k. Then there is a $j \leq n$ such that $u_k \notin H_j$. Consider $p(x) = 1 - x_j^2$. We have

$$\begin{aligned}
\omega_j &\geq \langle p, Y\rangle && \text{as } p(x) \in \mathcal{C}_j^{r_0}(\mathcal{Q}_G^{r_0}) \\
&= \sum_i a_i p(u_i) \\
&\geq a_k p(u_k) && \text{as for all } i,\ u_i \in D \subseteq [-1,1]^n \\
&> 0, && \text{because } u_k \notin H_j
\end{aligned}$$

which is a contradiction. □

For $d \geq 4$ the condition $D \subseteq [-1,1]^n$ can be dropped from Theorem 1. This allows more freedom in choosing the set D.

Theorem 2 *Let $d \geq 4$. Suppose that D is a compact set, and that $r_0 \geq d$ and $G := \{g_i(x) : i = 1,\ldots,m\} \subset \mathbf{R}[x]$ are such that $\mathcal{P}_d(D) \subseteq \mathcal{Q}_G^{r_0} \cap \mathbf{R}_d[x] \subseteq \mathcal{P}_d(D \cap \{-1,1\}^n)$. If the optimal value of subproblem (13) is zero for all $j = 1,\ldots,n$, then the optimal value of the dual problem of the master (12) equals the optimal value of the original binary polynomial program (11).*

Proof We follow the lines of the proof of Theorem 1. We write Y as the convex combination $Y = \sum_i a_i \mathcal{M}_d(u_i)$, where each $u_i \in D$. Again, it is enough to show that each u_i belongs to H.

Consider $p(x) = -x_j^4 + 2x_j^2 - 1 = -(x_j^2 - 1)^2 \in \mathcal{C}_j^{r_0}(\mathcal{Q}_G^{r_0})$. We have,

$$0 \leq \langle p, Y\rangle = -\sum_i a_i u_{ij}^4 + 2\sum_i a_i u_{ij}^2 - 1. \tag{14}$$

Now consider $q(x) = 1 - x_j^2$; we have $\pm q(x) \in \mathcal{C}_j^{r_0}(\mathcal{Q}_G^{r_0})$, and thus

$$0 \leq \langle \pm q, Y\rangle = \pm \sum_i a_i q(u_i) = \pm\left(1 - \sum_i a_i u_{ij}^2\right),$$

and thus,

$$\sum_i a_i u_{ij}^2 = 1. \tag{15}$$

Plugging (15) into (14) we obtain

$$\sum_i a_i u_{ij}^4 \leq 1. \tag{16}$$

Now, using the Cauchy-Schwarz inequality,

$$1 = \left(\sum_i a_i u_{ij}^2\right)^2 \leq \left(\sum_i a_i u_{ij}^4\right)\left(\sum_i a_i\right) = \sum_i a_i u_{ij}^4 \tag{17}$$

Using (16), equality follows in (17), and thus for each i we have $\sqrt{a_i}u_{ij}^2 = t\sqrt{a_i}$ for some constant t. Therefore, $u_{ij}^2 = t$ for all i and from Equation (15): $\sum_i a_i t = 1$, that is $t = 1$. □

Using that when DIGS-B stops, all the subproblems have value 0, we obtain the following corollary to Theorems 1 and 2.

Corollary 1 *Let D be a compact set. Assume $d \geq 4$, or $d \geq 2$ and $D \subseteq [-1,1]^n$. Let $r_0 \geq d$ and $G := \{g_i(x) : i = 1,\ldots,m\} \subset \mathbf{R}[x]$ be a description of $D \cap \{-1,1\}^n$ such that $\mathcal{P}_d(D) \subseteq \mathcal{Q}_G^{r_0}$. Assume DIGS-B (Algorithm 4) is run starting with $G_0 = G$, and it stops with all subproblems having value 0. Then, at stopping, the optimal value of the dual problem of the master* (12) *equals the optimal value of the original binary polynomial program* (11).

Proof Notice that as G is a description of $D \cap \{-1,1\}^n$ we have $\mathcal{Q}_G^{r_0} \cap \mathbf{R}_d[x] \subseteq \mathcal{P}_d(D \cap \{-1,1\}^n)$. Also $\mathcal{P}_d(D) \subseteq \mathcal{Q}_G^{r_0} \cap \mathbf{R}_d[x]$ by assumption. By induction, At each iteration $s \geq 0$, we have $\mathcal{P}_d(D) \subseteq \mathcal{Q}_{G_s}^{r_0} \cap \mathbf{R}_d[x] \subseteq \mathcal{P}_d(D \cap \{-1,1\}^n)$. Thus if DIGS-B stops at iteration k, the assumptions of Theorem 1 or Theorem 2 hold for $G = G_k$. □

3.2.2 Finite Termination

For the next theorem we need to define a variation of DIGS-B which we refer to as DIGS-B*. We show that DIGS-B* stops after a finite number of steps. Notice that Theorems 1 and 2 also hold for this version of DIGS-B (i.e. replacing $\mathcal{Q}_G^r$ by $\hat{\mathcal{Q}}_G^r$). To ensure finite termination, we use polyhedral approximations to the set $P_d(D \cap \{-1,1\}^n)$. Recall that for any D the set $\mathcal{P}_d(D \cap \{-1,1\}^n)$ is a polyhedron (Lemma 4).

Proceeding similarly as in Section 3.1, we let $G = \{g_0(x), g_1(x), \ldots, g_m(x)\}$ where $g_0(x) = 1$. Define $\widehat{\mathcal{Q}}_G^r$ as the following approximation to $P_d(D \cap \{-1,1\}^n)$:

$$\widehat{\mathcal{Q}}_G^r = \sum_{i=0}^{|G|} g_i[1-x, 1+x]\mathbf{R}^+_{r-\deg(g_i)}[x] + \sum_{i=1}^{n}(1-x_i^2)\mathbf{R}_{r-2}[x]$$

Fix $r_0 \geq d$ and define the polynomial programming master problem

$$\begin{aligned} \varphi_G^{r_0} \; &= \sup_{\lambda} \lambda \\ &\quad \text{s.t. } f(x) - \lambda \in \widehat{\mathcal{Q}}_G^{r_0}, \end{aligned}$$

which can be written as

$$\begin{aligned} \varphi_G^{r_0} \; = \; & \sup_{\lambda, \sigma_i(x), \delta_i(x)} \lambda \\ & \text{s.t.} \qquad f(x) - \lambda = \sum_{i=0}^{m} \sigma_i(x) g_i(x) + \sum_{i=1}^{n} \delta_i(x)(1-x_i^2). \\ & \qquad\quad \sigma_i(x) \in \mathbf{R}^+_{r_0-\deg(g_i)}[x], \qquad i = 0,1,\ldots,m \\ & \qquad\quad \delta_i(x) \in \mathbf{R}_{r_0-2}[x], \qquad i = 1,\ldots,n. \end{aligned}$$

In each iteration of DIGS-B* we add to the master problem the inequality produced by the subproblem with negative value of largest index. The main differences with the DIGS-B subproblem, besides using the polyhedral approximation, is that we use the 1-pseudo-norm, $\|p\|_1 = \sum_{0<\|\alpha\|\leq r_0} |p_\alpha|$ and for the j-subproblem we only take into account inequalities added when using subproblems of lower index. Thus we maintain a separate list of inequalities for each index. With these changes the master and all subproblems are linear programs.

Algorithm 5 DIGS-B*: DIGS for binary polynomial problems

Require: G description of S, $r_0 \geq d$,
$s \leftarrow 0$, $G_0^j = \leftarrow G$ for $j = 1, \ldots, n$
loop
Let

$$\textbf{(BP-M}_s\textbf{)} \quad \nu_s = \sup_{\lambda} \lambda \quad \text{s.t. } f(x) - \lambda \in \widehat{\mathcal{Q}}^{r_0}_{G_s^n}.$$

$Y_s \leftarrow$ dual optimal solution to (BP-M$_s$).
LET $p_s^j(x) \in \mathbf{R}_{r_0}[x]$ be an optimal (extreme) solution of

$$\alpha_s^j = \min_{p} \langle p, Y_s \rangle \quad \text{s.t. } p \in \mathcal{C}_j^{r_0}(\widehat{\mathcal{Q}}^{r_0}_{G_s^j}) \cap \mathbf{R}_{r_0}[x] \quad \| p \|_1 \leq 1,$$

if $\alpha_s^j \geq 0$ for all $j = 1, \ldots, n$ **then**
STOP
else
LET $j_s^* = \max\{j : \alpha_s^j < 0\}$
$G_{s+1}^i \leftarrow G_s^i \cup \{p_s^{j_s^*}(x)\}$ for $i = j_s^*, \ldots, n$
$s \leftarrow s + 1$
end if
end loop

Theorem 3 *Algorithm 5 terminates in finitely many iterations.*

Proof For sake of contradiction, assume Algorithm 5 does not stop for some $d > 0$ some finite $G \subset \mathcal{P}_d(S)$ and some $r_0 \geq d$. Consider the infinite sequence $\langle j_s^* \rangle_{s=1,2,3,\ldots}$ generated by the algorithm. As each $j_s^* \in \{1, \ldots, n\}$, the set $U_i = \{s : j_s^* = i\}$ is infinite for some $1 \leq i \leq n$. Let i^* be the minimum such i. For each $s \in U_{i^*}$ we have that $p_s^{j_s^*} = p_s^{i^*}$ is a extreme point of $W_s = \{p \in \mathcal{C}_{i^*}^{r_0}(\widehat{\mathcal{Q}}^{r_0}_{G_s^{i^*}}) \cap \mathbf{R}_{r_0}[x] : \|p\|_1 = 1\}$. As each W_s is a polytope, it has finitely many extreme points. Algorithm 5 does not generate repeated inequalities as every inequality generated has value 0 for all subproblems in future iterations. Thus the sequence $\langle W_s \rangle_{s \in U_{i^*}}$ has to change infinitely many times.

Now if $s_1 < s_2$ are such that $j_{s_1}^* = j_{s_2}^* = i^*$ and $j_s^* > i^*$ for all $s_1 < s < s_2$, then $G_{s_1}^{i^*} = G_{s_1+1}^{i^*} = \cdots = G_{s_2-1}^{i^*}$ and $G_{s_2}^{i^*} = G_{s_1}^{i^*} \cup \{p\}$ where $p = p_{s_2}^{i^*}$ is a extreme point of $W_{s_2-1} = W_{s_1}$. We claim that $W_{s_2} = W_{s_1}$. To show this write

$$p(x) = \frac{1 - x_{j^*}}{2} p^-(x) + \frac{1 + x_{j^*}}{2} p^+(x) + (1 - x_j^2) r(x), \tag{18}$$

where $p^-, p^+ \in \widehat{\mathcal{Q}}^{r_0}_{G_{s_1}^{i^*}}$ and $r \in \mathbf{R}_{r_0-1}[x]$. Notice that

$$\frac{1 - x_{j^*}}{2} p(x) = \frac{1 - x_{j^*}}{2} p^-(x) + \frac{1 - x_j^2}{4} (2(1 - x_{j^*}) r(x) + p^+(x) - p^-(x)). \tag{19}$$

From (18), $d = \deg p(x) \geq \deg(x_{j^*}(p^-(x) - p^+(x) + 2x_j r(x)))$ and thus $\deg(p^-(x) - p^+(x) + 2x_j r(x)) \leq d-1$. Using (19) we get $\frac{1-x_{j^*}}{2} p(x) \in \mathcal{C}_{i^*}^{r_0}(\widehat{\mathcal{Q}}^{r_0}_{G_{s_1}^{i^*}})$. Similarly, $\frac{1+x_{j^*}}{2} p(x) \in \mathcal{C}_{i^*}^{r_0}(\widehat{\mathcal{Q}}^{r_0}_{G_{s_1}^{i^*}})$ and thus $\mathcal{C}_{i^*}^{r_0}(\widehat{\mathcal{Q}}^{r_0}_{G_{s_2}^{i^*}}) = \mathcal{C}_{i^*}^{r_0}(\widehat{\mathcal{Q}}^{r_0}_{G_{s_1}^{i^*} \cup \{p\}}) = \mathcal{C}_{i^*}^{r_0}(\widehat{\mathcal{Q}}^{r_0}_{G_{s_1}^{i^*}})$ and the claim follows.

We have that the sequence $\langle W_s \rangle_{s \in U_{i^*}}$ has to change infinitely many times, but every time it changes there is an s between two consecutive elements of U_{i^*} such that $j_s^* < i^*$. Therefore $\{s : j_s^* < i^*\}$ is infinite, but this contradicts the selection of i^*. □

3.3 Comparison with Lift-And-Project Methods

For the case of binary linear programming, i.e. when $d = 1$, Algorithm 5 is equivalent to the Balas-Ceria-Cornuéjols (BCC) lift-and-project method [3].

Theorem 4 *Suppose D is polyhedral and $f(x)$ is linear in* (11)*. With $r_0 = 1$, Algorithm 5 is equivalent to (the dual of) BCC lift-and-project.*

Proof The Theorem follows from the fact that the projection of the (linearized) BCC lifting, denoted $P_j(x)$ in [3], is the dual of $\{p(x) \in \mathbf{R}_d[x] : p(x) \in \mathcal{C}_j^1(\mathcal{Q})\}$, the set of valid inequalities of the convex hull of $D \cap H_j$.

This fact follows by Farkas' Lemma: if $G = \{a_i^T x \geq b_i,\ i = 0, 1, \ldots, m\}$ with $a_0 = 0$ and $b_0 = 1$, then $\mathcal{Q}_G^1 := \sum_{i=1}^m (a_i^T x - b_i)\Sigma_0 = \mathcal{P}_1(D)$. □

We illustrate the dual relation between BCC and our approach by considering the LP relaxation of the binary linear programming problem

$$\begin{aligned} &\min c^T x \\ &\text{s.t. } Ax \geq b, \\ &\quad x \in \{-1, 1\}^n. \end{aligned}$$

For any given j, applying BCC with the variable x_j yields the lifting step

$$\begin{aligned} &\min c^T x \\ &\text{s.t. } (1 + x_j)(Ax - b) \geq 0 \\ &\quad (1 - x_j)(Ax - b) \geq 0, \end{aligned}$$

where the constraints include the box constraints $-1 \leq x_i \leq 1$ for all i.

Next we linearize and project by substituting the product of $x_j x_k$ by a new variable x_{jk} and using the substitution $x_j^2 = 1$:

$$\begin{aligned} \textbf{(P}_\textbf{L}\textbf{)}\ \min &\sum_i c_i x_{0i} \\ \text{s.t. } &\sum_i a_{ik} x_{0i} + \sum_i a_{ik} x_{ij} - b_k x_{0j} \geq b_k \qquad \forall k \qquad (20) \\ &\sum_i a_{ik} x_{0i} - \sum_i a_{ik} x_{ij} + b_k x_{0j} \geq b_k \qquad \forall k \qquad (21) \\ &x_{jj} = 1. \qquad (22) \end{aligned}$$

On the other hand, the problem

$$\begin{aligned} &\max \lambda \\ &\text{s.t. } \sum_i c_i x_i - \lambda \in \mathcal{C}_j^1(\mathcal{Q}) \end{aligned}$$

is equivalent to

$$\max \lambda$$
$$\text{s.t. } \sum_i c_i x_i - \lambda = (1 + x_j)\left[\sum_k \mu_k (a_k^T x - b_k)\right] + (1 - x_j)\left[\sum_k \nu_k (a_k^T x - b_k)\right] + r(1 - x_j^2).$$

Equating the coefficients of the monomials of the left side to the ones on the right side, we obtain

$$\begin{aligned} &\max \lambda \\ &\text{s.t. } \lambda = \sum_k \mu_k b_k + \sum_k \nu_k b_k - r \\ &\quad c_j = -\sum_k b_k \mu_k + \sum_k b_k \nu_k + \sum_k a_{kj} \mu_k + \sum_k a_{kj} \nu_k \\ &\quad c_i = \sum_k a_{kj} \mu_k + \sum_k a_{kj} \nu_k \\ &\quad 0 = -\sum_k a_{kj} \mu_k - \sum_k a_{kj} \nu_k - r \\ &\quad 0 = -\sum_k a_{kj} \mu_k - \sum_k a_{kj} \nu_k. \end{aligned}$$

Substituting λ by $\sum_k \mu_k b_k + \sum_k \nu_k b_k - r$ in the objective, we obtain the dual problem of ($\mathrm{P_L}$) where μ_k, ν_k, and r are the dual variables of constraints (20)-(22) respectively.

Theorem 4 shows that DIGS-B* is a generalization of BCC from degree 1 to higher degrees. This generalization suggests not only solving binary PPs of degree higher than one, but also the possibility for solving

degree-one binary PPs as problems of higher degrees. In other words, when solving a binary linear program using DIGS-B*, setting $r_0 = 1$ will generate the same linear inequalities as those generated by BCC. However, if we set r_0 to a higher value, then we can generate valid polynomial inequalities of higher degree. It is not clear a priory which one of the two methods is more efficient. While $r_0 = 1$ reduces to using linear programming, making each iteration much cheaper, larger values of r_0 might produce stronger valid inequalities, and possibly fewer iterations.

As an illustration, the next example compares DIGS-B with $r_0 = 2$ to BCC (DIGS-B with $r_0 = 1$) on the maximum stable set problem on instances taken from [27].

Example 6
The stable set number (see Example 5) can be formulated as a binary linear program as follows:

$$\begin{aligned} \textbf{(SS-LP)}\ \alpha(G) = \max & \sum_{i=1}^{n} x_i \\ \text{s.t. } & x_i + x_j \leq 1 \qquad \forall (i,j) \in E \\ & x \in \{0,1\}^n. \end{aligned}$$

The maximum stable set problem can also be formulated as follows:

$$\begin{aligned} \textbf{(SS-D2)}\ \alpha(G) = \max & \sum_{i=1}^{n} x_i \\ \text{s.t. } & x_i x_j = 0 \qquad \forall (i,j) \in E \\ & x \in \{0,1\}^n. \end{aligned}$$

We compare different versions of DIGS-B. For each instance we impose a 300-seconds time limit for each procedure. The upper bound obtained from BCC (i.e., by starting with the LP-relaxation of (SS-LP) and setting $r_0 = 1$, in which case master and subproblem are LP's) is compared to three different variations of DIGS-B, based on the (SS-D2) formulation. Linear refers to generating linear inequalities that are added to the master problem by using a non-negative multiplier. SOC refers to generating linear inequalities that are added to the master problem by using a polynomial multiplier that is in $\mathcal{P}_1(\mathcal{B})$ [12]. Quadratic refers to generating quadratic inequalities similar to the previous examples described. The results are reported in Table 7.

Table 7: Computational results for the stable set problem with a time limit of 300 seconds.

						DIGS-B					
						Linear		SOC		Quadratic	
n	Optimal value	(SS-LP) UB	Balas et al. UB	Iter.	(SS-D2) UB	UB	Iter.	UB	Iter.	UB	Iter.
8	3	4.00	3.00	197	3.44	3.00	186	3.00	126	3.02	49
11	4	5.50	4.00	160	4.63	4.00	139	4.00	130	4.05	109
14	5	7.00	5.02	135	5.82	5.02	114	5.01	91	5.14	82
17	6	8.50	6.22	121	7.00	6.23	84	6.09	63	6.30	54
20	7	10.00	7.46	104	8.18	7.43	68	7.25	45	7.42	38
23	8	11.50	8.81	88	9.36	8.61	50	8.36	33	8.67	22
26	9	13.00	10.11	77	10.54	9.84	37	9.60	25	9.96	14
29	10	14.50	11.65	65	11.71	11.10	24	10.87	17	11.18	10
32	11	16.00	13.03	56	12.89	12.37	18	12.20	14	12.53	6
35	12	17.50	14.48	49	14.07	13.49	13	13.32	10	13.66	4
38	13	19.00	16.05	43	15.24	14.80	8	14.74	7	14.85	4
41	14	20.50	17.69	39	16.42	15.88	7	15.77	6	16.26	1
44	15	22.00	19.10	34	17.59	17.19	6	17.09	5	17.30	1
47	16	23.50	20.78	29	18.77	18.39	4	18.26	4	18.59	1
50	17	25.00	22.18	27	19.94	19.52	4	19.42	4	19.77	1

From Table 7 it can be observe that the BCC performs the largest number of iterations for these instances. It uses linear programming which is computationally more efficient, however this efficiency comes at the expense of weaker improvement per iteration on the bound. On the other hand, the Quadratic approach provides the stronger average improvement per iteration but it is the method with the most expensive iterations. For all instances the bounds obtained by using the SOC version of DIGS-B are the best within 300 seconds. The three options for DIGS-B present comparable bounds, even though their subproblems correspond to different combinations of cost and bound improvement per iteration.

3.4 Examples for the Binary Case

Unless otherwise specified, the following stopping criteria are used

- For all algorithms a time limit of 5 hours (18000 seconds) is imposed. When the algorithm does not terminate in the time limit this is expressed using a dash (-). The symbol $^\flat$ besides the computational time indicates that performing one more iteration of the algorithm will exceed the time limit.
- For DIGS-B we stop if the subproblems have a value close to 0 ($\geq -10^{-3}$) or if we are able to extract a feasible solution that certifies optimality.

For the examples presented, we report the objective function value at iteration 0 and after performing a number of iterations of DIGS-B. As a reference for comparison we also present Lasserre's results. Values marked with $*$ indicate that the approach used terminated reporting optimality.

Example 7 Degree Two BPP
We consider quadratic problems with the following formulation

$$\begin{aligned} \max\ & x^T C x \\ \text{s.t. } & a^T x \leq b \\ & x \in \{-1, 1\}^n. \end{aligned}$$

Tables 8a-8c present computational results for quadratic instances where the parameters are generated according to [29]. The results are given for iterations 0, 1, 5, 10, 100, and 1000. In case the algorithm terminates before the given iteration, the number of iterations reached is shown in $(\cdot)$. The results show that DIGS-B is much more efficient than Lasserre's approach, in particular when n gets large. For $n > 20$, we are not able to go beyond $r = 2$ for Lasserre's relaxations in the given time limit of 5 hours while using DIGS-B we are able to improve the bounds by using this iterative scheme.

Table 8: Computational results for quadratic BPP.

(a) Lasserre's hierarchy

		$r=2$		$r=4$		$r=6$	
n	Optimal	bound	T(sec)	bound	T(sec)	bound	T(sec)
10	1653	1857.7	0.8	1707.3	28.1	1653.0*	11251.3
20	8510	9060.3	2.9	8639.7	17269.1	-	-
30	18229	19035.9	4.3	-	-	-	-
40	2679	4735.9	6.8	-	-	-	-
50	16192	21777.9	19.2	-	-	-	-
60	58451	62324.4	126.6	-	-	-	-

(b) DIGS-A

n	Optimal	Iter. 0	Iter. 1	Iter. 5	Iter. 10	Iter. 100	T(sec)
10	1653	1857.7	1781.3	1735.3	1691.3	1653.0*(31)	748.9
20	8510	9060.3	8674.9	-	-	-	1668.4$^\flat$
30	18229	19035.9	-	-	-	-	4.3$^\flat$
40	2679	4735.9	-	-	-	-	6.8$^\flat$
50	16192	21777.9	-	-	-	-	19.2$^\flat$
60	58451	62324.4	-	-	-	-	126.6$^\flat$

(c) DIGS-B

n	Optimal	Iter. 0	Iter. 1	Iter. 5	Iter. 10	Iter. 100	Iter. 1000	T(sec)
10	1653	1857.7	1821.9	1797.4	1782.2	1724.1	1669.8(1065)	17923.4$^\flat$
20	8510	9060.3	9015.3	8925.9	8826.6	8683.2	8653.8(517)	17750.3$^\flat$
30	18229	19035.9	18920.2	18807.4	18737.0	18626.2	18600.7(243)	17840.1$^\flat$
40	2679	4735.9	4590.7	4248.2	4105.3	3556.7(100)	-	18020.1$^\flat$
50	16192	21777.9	21390.3	20162.1	19407.1	18652.7(31)	-	18123.9$^\flat$
60	58451	62324.4	62019.1	60906.0	60585.5	-	-	17961.1$^\flat$

Example 8 Degree Three BPP
We consider the general BPP problem with degree-3 objective. The problem can be formulated as follows

$$\begin{aligned} \max \sum_{|\alpha|\leq 3} c_\alpha x^\alpha \\ \text{s.t. } a^T x \leq b \\ x \in \{-1,1\}^n. \end{aligned}$$

In Tables 9a-9c, we present Lasserre's, DIGS-A, and DIGS-B results on degree three instances where c_α is generated randomly between 0 and 10, each a is generated randomly between 1 and 50, and b is generated randomly between 50 and $\sum_{i=1}^n a_i$. The generated parameters c_α, a, and b are integer.

Lasserre's relaxation is not defined for $r = 2$, the smallest r we can take is $r = 4$. For $n \geq 25$, we are not able to compute any bound using Lasserre's relaxation within the time limit bound of 5 hours.

As we are working with a degree 3 BPP, DIGS-A is applied with a master problem of degree 3 (i.e. $r_0 = 3$) and a subproblem of degree $r_0 + 1 = 4$, creating inequalities of degree 2. That is, we apply Algorithm 2 with step 3 replaced by:

$$\begin{aligned} \min_{p_s} \left\langle p_s, Y^s \right\rangle \\ \text{s.t. } p_s(x) \in \mathcal{K}^4_{G_s} \cap \mathbf{R}_2[x] \\ \| \, p_s \, \| \leq 1. \end{aligned}$$

The results for various iterations are reported and the total time is given in seconds. In case the algorithm terminates before the given iteration, the number of iterations reached is shown in $(\cdot)$.

Table 9: Computational results for degree 3 BPP.

(a) Lasserre's hierarchy

		$r=4$		$r=6$	
n	Optimal	bound	T(sec)	bound	T(sec)
5	58	59.37	2.1	58.00*	9.6
10	139	148.97	35.9	139.00*	4866.0
15	1371	1524.71	1436.2	-	-
20	1654	1707.95	18106.6	-	-
25	-	-	-	-	-

(b) DIGS-A

n	Optimal	Iter. 0	Iter. 1	Iter. 5	Iter. 10	Iter. 100	T(sec)
5	58	67.16	63.59	58.37	58.00*(8)		19.6
10	139	154.59	153.22	142.32	139.19	139.00*(12)	606.3
15	1371	1582.04	1569.92	1498.60	1470.98	1429.89(23)	18037.9$^\flat$
20	1654	1718.53	1716.67	-	-	-	16009.2$^\flat$
25	-	3967.12	-	-	-	-	5038.6$^\flat$

(c) DIGS-B

n	Optimal	Iter. 0	Iter. 1	Iter. 5	Iter. 10	Iter. 100	T(sec)
5	58	67.16	58.45	58.00*(3)			5.1
10	139	154.59	151.43	143.33	139.64	139.00(13)*	100.9
15	1371	1582.04	1550.04	1511.76	1497.37	1392.89(91)	18058.1$^\flat$
20	1654	1718.53	1716.00	1709.40	1701.65	1699.70(12)	18168.6$^\flat$
25	-	3967.12	3960.78	-	-	-	14287.3$^\flat$

Table 9c presents computational results of applying Algorithm 4. We report results for iterations 0, 1, 5, and 10, 100, and 1000 and the total time in seconds. At Iteration 0, we use a relaxation of order 3 and then apply DIGS-B to add valid degree 3 inequalities to improve the bounds of the relaxation. Comparing the results of Table 9c with Table 9b, we see that using the specialized DIGS-B reduces the computational time significantly and provides better bounds.

4 Conclusion

We proposed a dynamic inequality generation scheme to generate valid polynomial inequalities for general polynomial programs. This scheme can be iterated to improve the SOS relaxation of a polynomial program

without incurring an exponential growth in the size of the relaxation. As a result, the proposed scheme is in principle scalable to large general polynomial programming problems. We also proposed specialized schemes for polynomial programs with all variables non-negative or with all variables binary. For binary polynomial programs, we prove that the proposed scheme converges to the global optimal solution for some problem families. We presented several examples illustrating the computational behavior of the scheme and comparing it to the behavior of the well-known Lasserre hierarchy.

DIGS can be applied to improve the bound of a PP in a computationally efficient way that can be used in solvers as an effective tool for strengthening the corresponding PP relaxation, unlike Lasserre's hierarchy that jumps with respect to the bound incurring a drastic increase in computational time. Therefore, given a limited time window, DIGS can improve the bounds iteratively and hence more efficiently than a hierarchical approach, for instance within a branch-and-bound approach.

An important issue for future research is to improve the time efficiency of the inequality generation subproblems. This will be key to improve the computational efficiency of the various DIGSs. A related question is how to generate more than one inequality using the same subproblem. These are important issues for the purpose of applying the proposed approach within a branch-and-cut framework.

Acknowledgements The authors sincerely thank the associate editor and two anonymous referees for their many helpful comments and suggestions.

References

1. M. Anjos and J. Lasserre, editors. *Handbook on Semidefinite, Conic and Polynomial Optimization*. International Series in Operations Research & Management Science. Springer-Verlag, 2012.
2. Y. Bai, E. de Klerk, D. Pasechnik, and R. Sotirov. Exploiting group symmetry in truss topology optimization. *Optimization and Engineering*, 10(3):331–349, 2009.
3. E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.
4. E. de Klerk. Exploiting special structure in semidefinite programming: A survey of theory and applications. *European Journal of Operational Research*, 201(1):1–10, 2010.
5. E. de Klerk and D. Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM Journal on Optimization*, 12(4):875–892, 2002.
6. E. de Klerk, D. Pasechnik, and A. Schrijver. Reduction of symmetric semidefinite programs using the regular*-representation. *Mathematical Programming*, 109(2-3):613–624, 2007.
7. E. de Klerk and R. Sotirov. Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. *Mathematical Programming*, 122(2):225–246, 2010.
8. M. Deza and M. Laurent. Applications of cut polyhedra I. *Journal of Computational and Applied Mathematics*, 55(2):191 – 216, 1994.
9. M. Deza and M. Laurent. Applications of cut polyhedra II. *Journal of Computational and Applied Mathematics*, 55(2):217–247, Nov. 1994.
10. K. Gatermann and P. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *Journal of Pure and Applied Algebra*, 192(1-3):95–128, 2004.
11. B. Ghaddar. *New Conic Optimization Techniques for Solving Binary Polynomial Programming Problems*. PhD thesis, University of Waterloo, 2011. Available at `http://uwspace.uwaterloo.ca/bitstream/10012/6139/1/Ghaddar_Bissan.pdf`.
12. B. Ghaddar, J. C. Vera, and M. F. Anjos. Second-order cone relaxations for binary quadratic polynomial programs. *SIAM Journal on Optimization*, 21(1):391–414, 2011.
13. S. Kim, M. Kojima, and P. Toint. Recognizing underlying sparsity in optimization. *Mathematical Programming*, 9(2):273–303, 2009.
14. J. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM Journal on Optimization*, 12(3):756–769, 2001.
15. J. Lasserre. Global optimization problems with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11:796–817, 2001.
16. J. Lasserre. Semidefinite programming vs. LP relaxations for polynomial programming. *Mathematics of Operations Research*, 27(2):347–360, 2002.
17. M. Laurent. A comparison of the sherali-adams, lovász-schrijver and lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496, 2001.
18. M. Laurent. Semidefinite representations for finite varieties. *Mathematical Programming*, 109(Ser. A):1–26, 2007.
19. L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.
20. Y. Nesterov. Structure of non-negative polynomials and optimization problems. Technical report, Technical Report 9749, CORE, 1997.
21. J. Nie and J. Demmel. Sparse SOS relaxations for minimizing functions that are summation of small polynomials. *SIAM Journal on Optimization*, 19(4):1534–1558, 2008.
22. J. Nie, J. Demmel, and B. Sturmfels. Minimizing polynomials via sum of squares over the gradient ideal. *Mathematical Programming: Series A and B*, 106(3):587–606, 2006.
23. P. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, California, 2000.
24. P. Parrilo. An explicit construction of distinguished representations of polynomials nonnegative over finite sets. Technical report, IFA Technical Report AUT02-02, Zurich - Switzerland, 2002.
25. P. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003.

26. P. Parrilo and B. Sturmfels. Minimizing polynomial functions, algorithmic and quantitative real algebraic geometry. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 60:83–89, 2003.
27. J. Peña, J. C. Vera, and L. Zuluaga. Computing the stability number of a graph via linear and semidefinite programming. *SIAM Journal on Optimization*, 18(1):87–105, February 2007.
28. J. F. Peña, J. C. Vera, and L. F. Zuluaga. Exploiting equalities in polynomial programming. *Operations Research Letters*, 36(2), 2008.
29. D. Pisinger. The quadratic knapsack problem-a survey. *Discrete Applied Mathematics*, 155(5):623–648, 2007.
30. I. Pólik and T. Terlaky. A survey of the $\mathcal{S}$-lemma. *SIAM Review*, 49:371–418, 2007.
31. M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42:969–984, 1993.
32. H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.
33. H. D. Sherali and C. H. Tuncbilek. Comparison of two reformulation-linearization technique based linear programming relaxations for polynomial programming problems. *Journal of Global Optimization*, 10(4):381–390, 1997.
34. N. Shor. A class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6):731–734, 1987.
35. H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite programming -Theory, Algorithms, and Applications*. Kluwer, 2000.
36. L. Zuluaga, J. C. Vera, and J. Peña. LMI approximations for cones of positive semidefinite forms. *SIAM Journal on Optimization*, 16(4), 2006.
37. L. F. Zuluaga. *A conic programming approach to polynomial optimization problems: Theory and applications*. PhD thesis, The Tepper School of Business, Carnegie Mellon University, Pittsburgh, 2004.