J Optim Theory Appl (2010) 145: 289–310
DOI 10.1007/s10957-009-9624-2

Decomposition-based Method for Sparse Semidefinite Relaxations of Polynomial Optimization Problems

P.M. Kleniati · P. Parpas · B. Rustem

Published online: 29 October 2009

Abstract We consider polynomial optimization problems pervaded by a sparsity pattern. It has been shown in Lasserre (SIAM J. Optim. 17(3):822–843, 2006) and Waki et al. (SIAM J. Optim. 17(1):218–248, 2006) that the optimal solution of a polynomial programming problem with structured sparsity can be computed by solving a series of semidefinite relaxations that possess the same kind of sparsity. We aim at solving the former relaxations with a decomposition-based method, which partitions the relaxations according to their sparsity pattern. The decomposition-based method that we propose is an extension to semidefinite programming of the Benders decomposition for linear programs (Benders, Comput. Manag. Sci. 2(1):3–19, 2005).

Keywords Polynomial optimization · Semidefinite programming · Sparse SDP relaxations · Benders decomposition

1 Introduction

We consider polynomial optimization problems (POPs) with a sparsity pattern. To handle this class of problems, we introduce a decomposition-based method based on the well-known Benders decomposition [3]. Two interesting properties characterize our method. Firstly, the problem structure plays a key role in the applicability of the presented method. Secondly, although the decomposition method that we propose admits as input a polynomial optimization problem, it is not applicable to it but to its sparse semidefinite (SDP) relaxation.

It has been shown in [1, 2] that the optimal solution of a polynomial optimization problem with a structured sparsity can be computed by solving a series of SDP

Communicated by P.M. Pardalos.

P.M. Kleniati (✉) · P. Parpas · B. Rustem
Department of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ, London, UK
e-mail: pk2003@doc.ic.ac.uk

relaxations. Moreover, the SDP relaxations inherit the sparsity pattern that underlies the polynomial optimization problem. In view of this, we exploit the structure (sparsity pattern) of the SDP relaxation and partition the set of (moment)[1] variables into appropriate subsets. As a result, the SDP problem (relaxation) is decomposed into a *master* problem and several *subproblems*. The master problem is an optimization problem over the coupling variables,[2] and each subproblem is an optimization problem over one of the remaining sets of variables, which are independent of each other. The master problem is equivalent to the SDP problem (relaxation) we intend to solve. However, it possesses an infinite number of constraints and for this reason at each iteration of our procedure we deal with a relaxed version of it. The objective value of the relaxed master problem at each iteration is a lower bound on the optimal objective value of the SDP problem. As a result, by solving a series of relaxed master problems we obtain a sequence of increasing lower bounds on the optimal objective value of the SDP relaxation.

Our algorithm is in line with the Benders decomposition for linear programs [3]. However, there exist two main differences between our procedure and the classical Benders. In the latter, the finiteness of the procedure is guaranteed due to the fact that the feasible regions of the subproblems are polyhedral cones, hence finitely generated. On the other hand, the feasible regions of our subproblems, i.e. the so-called *spectrahedra* [4], are not polyhedral. Therefore, they possess an infinite number of generators. Despite this fact, finite ϵ-convergence is shown in Theorem 4.1. Furthermore, in classical Benders the set of variables is partitioned into two disjoint subsets yielding one subproblem at each iteration. We partition the set of variables into several subsets, based on the problem structure, to yield more than one subproblems.

Our algorithm is divided into two phases. The first phase involves the method of partitioning the variables, thus we will usually refer to it as the preprocess phase. The second phase is the major body of the algorithm and involves the decomposition-based method for the sparse SDP relaxations. For simplicity, we may often refer to the decomposition-based method as our algorithm or our method. Nevertheless, the preprocess phase is equally important since it sets up the problem data and makes the decomposition-based method applicable.

Contribution The contribution of this paper is twofold: (i) an extension of the Benders decomposition to semidefinite programming is introduced; (ii) the proposed decomposition-based method is employed to solve polynomial optimization problems via their sparse SDP relaxations.

We ought to mention that sparse relaxations are weaker than their dense counterparts since they include fewer constraints. As a result, and as pointed out in [5], the solution of the sparsely relaxed problem may be less accurate than the solution of the densely relaxed problem if the latter were possible to be used instead.[3] However, in

[1]We will refer to the objective variables of the SDP relaxations as *moment variables* due to the theory underlying the sparse SDP relaxation technique. This technique is discussed in Sect. 2.2.

[2]These are the (moment) variables that appear in all the constraints.

[3]For quadratic polynomial optimization problems, the quality of bounds achieved by sparse and dense relaxation technique is equally good as reported in [2].

this work we choose to focus on tackling polynomial problems through their sparse relaxations only, due to the great potential sparsity offers, including the applicability of the sparse SDP relaxation technique to large-scale polynomial problems. Moreover, not only can the sparsity pattern that pervades a polynomial optimization problem be automatically detected using the procedure described in [6], but also general polynomial optimization problems can be transformed into their sparse equivalent using the method introduced in [5].

The paper is organized as follows. Section 2 is devoted to a brief overview of the underlying theory. In Sect. 3, the preprocess phase is described. In Sect. 4, we introduce the decomposition-based method for solving sparse SDP problems. Theoretical results, including convergence of our procedure, are analyzed. Section 5 includes technical details of our procedure, while Sect. 6 discusses the performance of our algorithm when tested on benchmark problems and presents the corresponding numerical results. Section 7 concludes.

Notation Let S_n be the vector space in $\mathbb{R}^{\binom{n+1}{2}}$ of symmetric $n \times n$ matrices. The inner product in this space is: $\langle A, B\rangle = \operatorname{tr}(AB)$, for $A, B \in S_n$. The trace $\operatorname{tr}(\cdot)$ is the sum of diagonal elements of a square matrix and is a linear function. A matrix $A \in S_n$ is positive semidefinite if $x^{\mathrm{T}}Ax \geq 0$ for all vectors $x \in \mathbb{R}^n$. Similarly, a matrix $A \in S_n$ is positive definite if $x^{\mathrm{T}}Ax > 0$ for all vectors $x \neq 0 \in \mathbb{R}^n$. It is common to write $A \succeq 0$ ($A \succ 0$) to denote that A is positive semidefinite (positive definite) matrix and the notation $A \succeq B$ implies that $A - B \succeq 0$. In addition, by $X := \operatorname{Mat}(x)$ we denote the $n \times n$ symmetric matrix whose (i, j)th element is the $((j-1)n + i)$th element of a vector $x \in \mathbb{R}^{n^2}$. In the same vein, $x := \operatorname{vec}(X)$ denotes a vector $x \in \mathbb{R}^{n^2}$ whose $((j-1)n + i)$th component is the (i, j)th component of a matrix $X \in S_n$. The cone $\mathcal{K}^n = \{x \in R^{n^2} \mid x = \operatorname{vec}(X); X \succeq 0\}$ is the cone of vectors obtained from the vectorization of symmetric positive semidefinite matrices and $x \succeq_{\mathcal{K}^n} 0$ means that $x \in \mathcal{K}^n$.

2 Relevant Theory

2.1 Semidefinite Programming (SDP)

Consider the primal semidefinite programming problem

$$z_1 = \left\{\min_x c^{\mathrm{T}}x \mid \mathcal{A}x = b, x \succeq_{\mathcal{K}^n} 0\right\}, \tag{1}$$

where $c, x \in \mathbb{R}^{n^2}$, $\mathcal{A} \in \mathbb{R}^{m\times n^2}$, and its dual

$$z_2 = \left\{\max_y b^{\mathrm{T}}y \mid c - \mathcal{A}^{\mathrm{T}}y \succeq_{\mathcal{K}^n} 0\right\}, \tag{2}$$

where $b, y \in \mathbb{R}^m$. Semidefinite programming is underpinned by two important theorems, the *strong duality theorem* and the *extended Farkas lemma*. Both theorems are stated below.

Theorem 2.1 (Strong Duality [7]) *Let z_1 and z_2 be the objective values of* (1) *and* (2), *respectively. Assume that there exists an m-vector y such that* $\mathrm{Mat}(\mathcal{A}y^{\mathrm{T}}) \succ 0$. *Then*, $z_1 = z_2$.

Lemma 2.1 (Extended Farkas Lemma [7]) *Let $b \in \mathbb{R}^m$ and $\mathcal{A} \in \mathbb{R}^{m\times n^2}$ be a matrix such that its rows $\mathcal{A}_i^{\mathrm{T}} = \mathrm{vec}(A_i)$, where A_i are symmetric $n \times n$ matrices for $i = 1, \dots, m$. Furthermore, let there be an m-vector y such that* $\mathrm{Mat}(\mathcal{A}^{\mathrm{T}}y) \succ 0$. *Then, there exists a symmetric matrix $X \succeq 0$, with $\mathcal{A}\mathrm{vec}(X) = b$ or $\mathcal{A}x = b$, if and only if $y^{\mathrm{T}}b \geq 0$ for all y for which* $\mathrm{Mat}(\mathcal{A}^{\mathrm{T}}y) \succeq 0$.

There are several variations of the extended Farkas lemma. For the purposes of our work, we need to state one of these variations.

Lemma 2.2 ([7]) *Let $\mathcal{A} \in \mathbb{R}^{n^2\times m}$ be a matrix such that its columns are linearly independent and are of the form $\mathrm{vec}(A_i)$, for symmetric A_i, and let $B \in \mathbb{R}^{n\times n}$. Assume that there exists some symmetric matrix $Y \succ 0$ such that $\mathrm{vec}(Y)^{\mathrm{T}}\mathcal{A} = 0$. Then,* $\mathrm{Mat}(\mathcal{A}x) \preceq B$ *has a solution in x if and only if $\langle B, Y\rangle \geq 0$ for all $Y \succeq 0$ for which* $\mathrm{vec}(Y)^{\mathrm{T}}\mathcal{A} = 0$.

In other words, when we deal with the feasibility of the dual SDP problem (2), one of the two systems will be consistent:

$$c - \mathcal{A}^{\mathrm{T}}y \succeq_{\mathcal{K}^n} 0, \tag{3}$$

$$u^{\mathrm{T}}c = -1, \quad u^{\mathrm{T}}\mathcal{A} = 0, \quad u \succeq_{\mathcal{K}^n} 0. \tag{4}$$

The solution of the system (4) is called the *Farkas dual solution*. For further reading on semidefinite programming, the interested reader is referred to [8–11] and the rich bibliography therein.

2.2 Sparse SDP Relaxations of Polynomial Problems

Consider the following polynomial optimization problem:

$$p^* = \left\{ \min_{x\in\mathbb{R}^n} \sum_{k=1}^{p} p_k(x_k) \mid g_j(x_k) \geq 0,\ \sum_{j\in\mathcal{J}_k} j = m,\ k = 1, \dots, p \right\}, \tag{5}$$

where m denotes the total number of constraints.[4] Every polynomial involved in the above problem is a polynomial dependent only on some subset $\{x_k \mid k \in \mathcal{I}_k\}$ of the objective variables $x \in \mathbb{R}^n$, where $\mathcal{I}_k \subset \{1, \dots, n\}$ and $\bigcup_{k=1}^{p} \mathcal{I}_k = \{1, \dots, n\}$. Notice that these index sets $\{\mathcal{I}_1, \dots, \mathcal{I}_p\}$ may not be disjoint, in which case their intersection is equal to the set of *linking* or *coupling* variables, namely those variables that appear

[4]We assume that the m constraints of problem (5) also include the p redundant constraints $n_k M^2 - \|x(\mathcal{I}_k)\|^2 \geq 0$, where n_k is the cardinality of index set $\mathcal{I}_k$ and $M > \|x\|_\infty$ for all feasible points x, as indicated in [1].

in all the constraints. In addition, in the definition of problem (5) observe the existence of another collection of p index sets $\mathcal{J}_k$. These sets are defined as follows:

$$\mathcal{J}_k = \{j \in \{1, \dots, m\} \mid g_j \in \mathbb{R}[x(\mathcal{I}_k)]\},$$

where $x(\mathcal{I}_k) = \{x_k \mid k \in \mathcal{I}_k\}$. In other words, for every $j \in \mathcal{J}_k$, the constraint g_j is only dependent on the variable set $x(\mathcal{I}_k)$. The sets $\{\mathcal{J}_1, \dots, \mathcal{J}_p\}$ are disjoint.

The SDP relaxation of order ω for problem (5) is given below [1]:

$$\begin{aligned} p^*_\omega = \min_y & \sum_{k=1}^{p} \sum_{\alpha_k \in N^n} p_{\alpha_k} y_{\alpha_k}, \\ \text{s.t.}\quad & M_\omega(y, \mathcal{I}_k) \succeq 0, \quad k = 1, \dots, p, \\ & M_{\omega - d_j}(g_j y, \mathcal{I}_k) \succeq 0, \quad j \in \mathcal{J}_k,\ k = 1, \dots, p, \\ & y_0 = 1, \end{aligned} \tag{6}$$

for $2\omega \geq \max\{\deg f, \max_j \deg g_j\}$, where ω is called *order* of the relaxation. By increasing ω and formulating the corresponding sparse relaxations, one obtains a hierarchy of convergent sparse SDP relaxations. In particular, Theorem 3.1 in [1] shows that, under a certain assumption on the sparsity pattern, or in other words under an assumption on the sets $\{\mathcal{I}_1, \dots, \mathcal{I}_p\}$, the resulting sequence of optimal objective values of the relaxations converges to the global optimal solution p^* of (5). Moreover, if (5) has a unique global minimizer x^*, then the resulting sequence of optimal solution vectors of the relaxations converges to the global minimizer x^* [1, Theorem 3.1].

The matrices $M_\omega(y, \mathcal{I}_k)$ and $M_{\omega - d_j}(g_j y, \mathcal{I}_k)$ in (6) are called *moment* and *localizing* matrices, respectively. The interested reader can find all the details of the sparse SDP relaxation technique in [1] and [2]. Given that our paper is focusing on solving the sparse SDP relaxations (6), we restrict ourselves to addressing the sparse SDP relaxation technique only. For a thorough investigation of the underlying theory on dense[5] SDP relaxations of polynomial programming problems, the reader is referred to [12, 13] and the references therein. Also, [14] contains an explanatory survey on the topic.

3 Preprocess Phase (Partitioning of Variables)

Our decomposition-based method intends to solve problems (6) by exploiting their decomposable sparse structure. But the question that arises is how to find (compute) such structure/pattern and how this would help us partition the set of (moment) variables y. The answer comes from the fact that the sparsity pattern that underlies the original polynomial optimization problem (5) is inherited into its sparse SDP relaxation. Therefore, if the polynomial problem has a specific sparsity pattern expressed

[5] Sparsity pattern is not taken into account.

by the collection $\{\mathcal{I}_1, \ldots, \mathcal{I}_p\}$, we are able to specify the sparsity pattern of the semidefinite relaxation in an equivalent way.

In fact, the sets $\{\mathcal{I}_1, \ldots, \mathcal{I}_p\}$ are the *maximal cliques* of a chordal graph with as many nodes as the number of polynomial variables [1, 2]. When the intersection of these sets is nonempty, the resulting set is the index set of the coupling variables, i.e. the variables that appear in all the constraints. This phenomenon is known as *weak coupling*, in contrast to *strong coupling* where $\mathcal{I}_k \cap \mathcal{I}_{k+j} = \emptyset$ for $j > 1$. In the former case, what is essential to note is that, if we remove (fix) the coupling variables, there remain p disjoint subsets of independent variables. In other words, if $\mathcal{I}_0'$ is the set of coupling variables, where $\mathcal{I}_0' \subset \{1, \ldots, n\}$, then the set $\{1, \ldots, n\} \setminus \mathcal{I}_0'$ is partitioned into p disjoint sets $\mathcal{I}_k'$ such that $\mathcal{I}_k = \mathcal{I}_0' \cup \mathcal{I}_k'$, $k = 1, \ldots, p$, and $\mathcal{I}_k \cap \mathcal{I}_j = \mathcal{I}_0'$, for all $j \neq k$ [1]. In view of this, if there exists a weak coupling, i.e. $p > 1$, we automate the partitioning of the moment variables in problem (6).

In particular, we partition the moment variables in one subset of *coupling moment variables* and p disjoint subsets of *independent moment variables*. To do so, we use the information taken from the collections $\mathcal{I}_1, \ldots, \mathcal{I}_p$ and $\mathcal{I}_0', \ldots, \mathcal{I}_p'$. Then, the subset of the coupling moment variables[6] is derived from the set of *coupling polynomial variables* $\mathcal{I}_0'$, and the ith set of independent moment variables is derived from the index set or *clique* $\mathcal{I}_i$ of indices of *independent polynomial variables* together with the indices of the coupling polynomial variables, $i = 1, \ldots, p$. For convenience of the reader, let us recapitulate.

Remark 3.1 The coupling moment variables are derived from the coupling polynomial variables index set $\mathcal{I}_0'$.

Remark 3.2 The ith set of independent moment variables is derived from the ith index set $\mathcal{I}_i$ $(= \mathcal{I}_0' \cup \mathcal{I}_i')$, $i = 1, \ldots, p$.

The way the aforesaid subsets are derived is based on the fact that the moment variables correspond to the products of powers of certain variables, i.e. *monomials*. Since the sparsity pattern remains unchanged, the ith subset of the moment variables, $i = 1, \ldots, p$, corresponds to the set of monomials formed by the specific polynomial variables belonging to the index set $\mathcal{I}_i$, up to the specified relaxation order. Similarly, the set of the coupling moment variables corresponds to the set of monomials formed by the specific polynomial variables belonging to the index set $\mathcal{I}_0'$, up to the specified relaxation order. In other words, two parameters affect the generation of the subset of coupling moment variables and the p disjoint subsets: the collection $\mathcal{I}_0', \mathcal{I}_1, \ldots, \mathcal{I}_p$, and the relaxation order. In fact, the relaxation order determines the number of moment variables[7] and the collection $\mathcal{I}_0', \mathcal{I}_1, \ldots, \mathcal{I}_p$ determines which moment variable belongs to which subset. For instance, let us examine the following example taken from [15]:

[6]The coupling moment variables appear in all the semidefinite constraints of problem (6).

[7]The bigger the relaxation order is, the more monomials are considered.

Example 3.1

$$\begin{aligned}
\max \quad & 0.5 \cdot (x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2) + 10.5x_1 + 7.5x_2 + 3.5x_3 + 2.5x_4 \\
& \quad + 1.5x_5 + 10x_6, \\
\text{s.t.} \quad & 6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 \leq 6.5, \\
& 10x_1 + 10x_3 + x_6 \leq 20, \\
& 0 \leq x_i \leq 1, \quad i = 1, \ldots, 5, \\
& 0 \leq x_6 \leq 20.
\end{aligned}$$

The sparsity pattern of Example 3.1 is expressed by $p = 2$ *maximal cliques*, i.e. $\mathcal{I}_1 = \{1, 3, 6\}$ and $\mathcal{I}_2 = \{1, 2, 3, 4, 5\}$. The coupling polynomial variables are then given by the set $\mathcal{I}'_0 = \{1, 3\}$ and the $p = 2$ disjoint sets of independent polynomial variables are $\mathcal{I}'_1 = \{6\}$ and $\mathcal{I}'_2 = \{2, 4, 5\}$. If we form the sparse SDP relaxation of order 1, we get a semidefinite problem with 21 moment variables. These are given in Table 1 along with the corresponding monomials.

The sparsity pattern that underpins the first SDP relaxation of Example 3.1 is expressed by the set of coupling moment variables (CMV): $\{y_1, y_3, y_7, y_9, y_{16}\}$ extracted from the set $\mathcal{I}'_0$ of coupling polynomial variables, and $p = 2$ disjoint sets of independent moment variables (IMV): $\{y_6\}$, derived from the index set $\mathcal{I}_1$ and $\{y_2, y_4, y_5, y_8, y_{10}, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{17}, y_{18}, y_{19}, y_{20}, y_{21}\}$, derived from the index set $\mathcal{I}_2$. The three foregoing sets are summarized in Table 2.

Table 1 Moment variables and corresponding monomials for the first SDP relaxation of Example 3.1

Moment Variable (MV)	Monomial (M)	MV	M	MV	M
y_1	x_1	y_8	x_1x_2	y_{15}	x_2x_5
y_2	x_2	y_9	x_1x_3	y_{16}	x_3^2
y_3	x_3	y_{10}	x_1x_4	y_{17}	x_3x_4
y_4	x_4	y_{11}	x_1x_5	y_{18}	x_3x_5
y_5	x_5	y_{12}	x_2^2	y_{19}	x_4^2
y_6	x_6	y_{13}	x_2x_3	y_{20}	x_4x_5
y_7	x_1^2	y_{14}	x_2x_4	y_{21}	x_5^2

Table 2 Partitioning of variables for the first SDP relaxation of Example 3.1

CMV	IMV (1)	IMV (2)		
y_1	y_6	y_2	y_{11}	y_{17}
y_3		y_4	y_{12}	y_{18}
y_7		y_5	y_{13}	y_{19}
y_9		y_8	y_{14}	y_{20}
y_{16}		y_{10}	y_{15}	y_{21}

To sum up, given the sparsity pattern of the polynomial problem, as well as the dimension of its sparse SDP relaxation of order ω, we can automatically derive the sparsity pattern of the relaxation. Next, based on the relaxation sparsity pattern, we partition the moment variables into the set of coupling moment variables and p sets of independent moments variables. Such a partitioning decomposes the SDP relaxation into several smaller problems, i.e. the (relaxed) master problem and the subproblems. In what follows, the coupling moment variables are the objective variables of the master problem in addition to few more objective variables which we add for convenience, namely the scalar variables $z_1, \ldots, z_p$. In the same vein, the ith set of independent moment variables is the set of objective variables of the ith subproblem. More details on the derivation of the *master problem* and the *subproblems* can be found in Sect. 4. Below we recapitulate the preprocess phase.

Algorithm 1 (Preprocess Phase)

1. Input the polynomial optimization problem and the desired relaxation order.
2. Extract the collections $\mathcal{I}_1, \ldots, \mathcal{I}_p$ and $\mathcal{I}'_0, \ldots, \mathcal{I}'_p$, and the dimension/data of the semidefinite relaxation (procedure from [2]).
3. Compute the sparsity pattern of the semidefinite relaxation, namely the coupling moment variables and the p disjoint sets of independent moment variables.
4. Output the data of the semidefinite relaxation and its sparsity pattern.

4 Decomposition-Based Method for Sparse SDP Problems

Taking into account the formerly described sparsity pattern of the sparse SDP relaxations (6), we consider the following SDP problem:

$$\begin{aligned} \min_{y, y^1, \ldots, y^p} \quad & b^{\mathrm{T}} y + \sum_{i=1}^{p} {d^i}^{\mathrm{T}} y^i, \\ \text{s.t.} \quad & T^i y + W^i y^i + h^i \succeq_{\mathcal{K}^{m^i}} 0, \quad i = 1, \ldots, p, \\ & Ay + c \succeq_{\mathcal{K}^v} 0, \end{aligned} \tag{7}$$

where the variable vector y corresponds to the set of coupling moment variables and the variable vectors y_i, $i = 1, \ldots, p$, correspond to the p disjoint sets of independent moment variables. Fixing the coupling variables y yields the following decomposition problem:

$$\min_y b^{\mathrm{T}} y + \rho(y), \quad \text{s.t.} \quad Ay + c \succeq_{\mathcal{K}^v} 0, \tag{8}$$

where

$$\rho(y) = \sum_{i=1}^{p} \rho_i(y), \tag{9}$$

$$\rho_i(y) = \left\{ \min_{y^i} {d^i}^{\mathrm{T}} y^i \mid W^i y^i + (h^i + T^i y) \succeq_{\mathcal{K}^{m^i}} 0 \right\}. \tag{10}$$

The latter problems are the *subproblems*. Their duals read:

$$\max_{\lambda^i} (-h^i - T^i y)^{\mathrm{T}} \lambda^i, \quad \text{s.t.} \quad {W^i}^{\mathrm{T}} \lambda^i = d^i, \lambda \succeq_{\mathcal{K}^{m^i}} 0. \tag{11}$$

The p subproblems (10) and their duals are the means of solving the original SDP problem (7). Note that subproblems (10) or (11) are independent of each other, a fact that gives rise to the possibility of a parallel implementation. More details on the implementation can be found in Sect. 5. At the moment, we are interested in examining whether the subproblems are feasible or not. This issue can be tackled using the extended Farkas Lemma 2.2. According to the systems of (3) and (4), the subproblems (10) are infeasible if, for each $i = 1, \dots, p$, there exists a Farkas dual solution that satisfies the following system:

$$(h^i + T^i y)^{\mathrm{T}} u^i = -1, \quad -{W^i}^{\mathrm{T}} u^i = 0, \quad u^i \succeq_{\mathcal{K}^{m^i}} 0. \tag{12}$$

As a result, we obtain the following lemma.

Lemma 4.1 (Feasibility Constraints) *Let $Y = \{y \mid Ay + c \succeq_{\mathcal{K}^v} 0\}$. Also let $V^i = \{y \mid T^i y + (W^i y^i + h^i) \succeq_{\mathcal{K}^{m^i}} 0$ for some $y^i\}$, for all $i = 1, \dots, p$, and let $V = \bigcap_{i=1}^p V^i$. Then, a point $\hat{y} \in Y$ is also in V if and only if $\hat{y}$ satisfies the inequalities below for $i = 1, \dots, p$:*

$$(h^i + T^i y)^{\mathrm{T}} u^i \geq 0, \tag{13}$$

for all $u^i \succeq_{\mathcal{K}^{m^i}} 0$ such that $-{W^i}^{\mathrm{T}} u^i = 0$.

Proof If $\hat{y} \in V$, then $T^i y + (W^i y^i + h^i) \succeq_{\mathcal{K}^{m^i}} 0$ for some y^i, $i = 1, \dots, p$. Consequently, there are no Farkas dual solutions and conditions (13) are satisfied for all $u^i \succeq_{\mathcal{K}^{m^i}} 0$ such that $-{W^i}^{\mathrm{T}} u^i = 0$. To prove the converse, let us assume that the conditions (13) are satisfied for all $u^i \succeq_{\mathcal{K}^{m^i}} 0$ such that $-{W^i}^{\mathrm{T}} u^i = 0$ and that $\hat{y} \notin V$. Since $\hat{y} \notin V$, for each $i = 1, \dots, p$, there exists a Farkas dual solution that satisfies the following system:

$$(h^i + T^i y)^{\mathrm{T}} u^i = -1, \quad -{W^i}^{\mathrm{T}} u^i = 0, \quad u^i \succeq_{\mathcal{K}^{m^i}} 0. \tag{14}$$

But this contradicts our assumption; hence, $\hat{y} \in V$. □

Conditions (13) are the feasibility constraints. Taking the aforesaid into account, we are able to rewrite our initial problem (7) as follows:

$$\min_{y \in Y \cap V} b^{\mathrm{T}} y + \sum_{i=1}^{p} \left\{ \min_{y^i} {d^i}^{\mathrm{T}} y^i \mid W^i y^i + (h^i + T^i y) \succeq_{\mathcal{K}^{m^i}} 0 \right\}, \tag{15}$$

where as stated earlier, $Y = \{y \mid Ay + c \succeq_{\mathcal{K}^v} 0\}$ and $V = \bigcap_{i=1}^p V_i$ for the sets $V^i = \{y \mid T^i y + (W^i y^i + h^i) \succeq_{\mathcal{K}^{m^i}} 0$ for some $y^i\}$, $i = 1, \dots, p$.

The constraints $y \in Y \cap V$ ensure the feasibility of the inner optimization problems, i.e. the subproblems, so we are able to employ the strong duality Theorem 2.1 and introduce the following corollary.

Corollary 4.1 (Optimality Constraints) *For each subproblem i, $i = 1, \dots, p$, if there exists an m-vector y^i such that* $\mathrm{Mat}(-{W^i}^{\mathrm{T}} y^i \succ 0)$, *then for fixed $y \in Y$ the optimal values of* (10) *equal those of their duals on $Y \cap V$, that is,*

$$\rho_i(y) = \left\{\max_{\lambda^i} (-h^i - T^i y)^{\mathrm{T}} \lambda^i \mid {W^i}^{\mathrm{T}} \lambda^i = d^i, \lambda \succeq_{\mathcal{K}^{m^i}} 0\right\}. \tag{16}$$

Using Corollary 4.1, problem (15) becomes

$$\min_{y \in Y \cap V} b^{\mathrm{T}} y + \sum_{i=1}^{p} \left\{\max_{\lambda^i} (-h^i - T^i y)^{\mathrm{T}} \lambda^i \mid {W^i}^{\mathrm{T}} \lambda^i = d^i,\ \lambda^i \succeq_{\mathcal{K}^{m^i}} 0\right\}. \tag{17}$$

The optimal solution of each inner dual SDP problem, introduced above, consists of the extreme points of the corresponding feasible region. So, denoting the extreme points as $\lambda^i \in \Lambda^i$, where $\Lambda^i = \{{W^i}^{\mathrm{T}} \lambda^i = d^i,\ \lambda^i \succeq_{\mathcal{K}^{m^i}} 0\}$, and similarly denoting the complementary points as $u^i \in U^i$, where $U^i = \{{W^i}^{\mathrm{T}} u^i = 0,\ u^i \succeq_{\mathcal{K}^{m^i}} 0\}$, we obtain the formulation

$$\begin{aligned} &\min_{y} b^{\mathrm{T}} y + \sum_{i=1}^{p} \left\{\max_{\lambda^i \in \Lambda^i} (-h^i - T^i y)^{\mathrm{T}} \lambda^i\right\}, \\ &\text{s.t.} \quad (h^i + T^i y)^{\mathrm{T}} u^i \ge 0, \quad \forall u^i \in U^i,\ i = 1, \dots, p, \\ &\qquad Ay + c \succeq_{\mathcal{K}^v} 0. \end{aligned} \tag{18}$$

Finally, we introduce scalars $z_1, \dots, z_p$ to obtain the following final form of the *master* problem:

$$\begin{aligned} &\min_{y, z_1, \dots, z_p} b^{\mathrm{T}} y + \sum_{i=1}^{p} z_i, \\ &\text{s.t.} \quad z_i \ge (-h^i - T^i y)^{\mathrm{T}} \lambda^i, \quad \forall \lambda^i \in \Lambda^i,\ i = 1, \dots, p, \\ &\qquad 0 \ge (-h^i - T^i y)^{\mathrm{T}} u^i, \quad \forall u^i \in U^i,\ i = 1, \dots, p, \\ &\qquad Ay + c \succeq_{\mathcal{K}^v} 0, \end{aligned} \tag{19}$$

which is equivalent to (7). The first set of constraints consists of the *optimality constraints* and the second set includes conditions (13), i.e. the *feasibility constraints*.

The number of constraints of problem (19) is in general infinite. The feasible regions of the SDP subproblems (10) are nonpolyhedral, which means that they possess an infinite number of generators, i.e. extreme points. The solution strategy that we follow is *relaxation*. Hence, we solve a relaxed version of our master problem ignoring all but few constraints and, at each iteration, based on the solutions of the subproblems for fixed y, we either add p feasibility constraints or p optimality constraints. Despite the infinite number of constraints in (19), in Theorem 4.1 we show finite termination of our procedure within any given accuracy.

The algorithm is characterized by some attractive properties. The subproblems and the master problem are convex programming problems. The optimality and feasibility constraints are linear. Thus, at each iteration, the relaxed master problem is only amended by linear constraints. Such a feature keeps the relaxed master problems simple. Moreover, in case the original polynomial problem lacks constraints on the coupling variables, then the semidefinite relaxation (7) does not have any constraints of type $Ay + c \succeq_{\mathcal{K}^\nu} 0$, which gives rise to the following linear master problem:

$$\begin{aligned} &\min_{y,z_1,\dots,z_p} b^{\mathrm{T}} y + \sum_{i=1}^{p} z_i, \\ &\text{s.t.} \quad z_i \geq (-h^i - T^i y)^{\mathrm{T}} \lambda^i, \quad \forall \lambda^i \in \Lambda^i,\ i = 1, \dots, p, \\ &\qquad\ \ 0 \geq (-h^i - T^i y)^{\mathrm{T}} u^i, \quad \forall u^i \in U^i,\ i = 1, \dots, p. \end{aligned} \tag{20}$$

In such cases, we possess a linear (relaxed) master problem at each iteration and, as is well known, there are numerous fast and reliable linear programming solvers capable of solving large-scale linear programming problems, such as `lp_solve` [16].

4.1 Algorithm

Our decomposition-based method for solving sparse SDP relaxations of polynomial problems is stated next. In this description of the algorithm, we consider problem (7) as our input problem; for simplicity, we assume that it has an optimal solution. Our complete procedure, including the preprocess phase, is presented in the flow diagram of Fig. 1.

Algorithm 2 (Decomposition-Based Method for Sparse SDP Problems)

Step 1: Initialize y (i.e. the set of coupling moment variables) to $\hat{y}_1$, where $\hat{y}_1 \in Y \cap V$. Initialize the iteration counter, e.g. $k = 1$ and set the lower (LB) and upper (UB) bounds to minus infinity ($-\infty$) and plus infinity (∞), respectively. Set $n_{\text{opt}} = 0$ and $n_{\text{feas}} = 0$, where n_{opt} is the counter for the optimality constraints and n_{feas} is the counter for feasibility constraints. Determine the convergence tolerance parameter $\epsilon > 0$.

Step 2: Solve the ith subproblem (10), $i = 1, \dots, p$, for $y = \hat{y}_k$.

Step 2.1: If all p subproblems are infeasible, obtain p Farkas dual solutions $\bar{u}^i_k$ and generate p feasibility constraints

$$(h^i + T^i y)^{\mathrm{T}} \bar{u}^i_k \geq 0, \quad i = 1, \dots, p. \tag{21}$$

Increase the infeasibility counter $n_{\text{feas}} = n_{\text{feas}} + 1$. Go to Step 3.

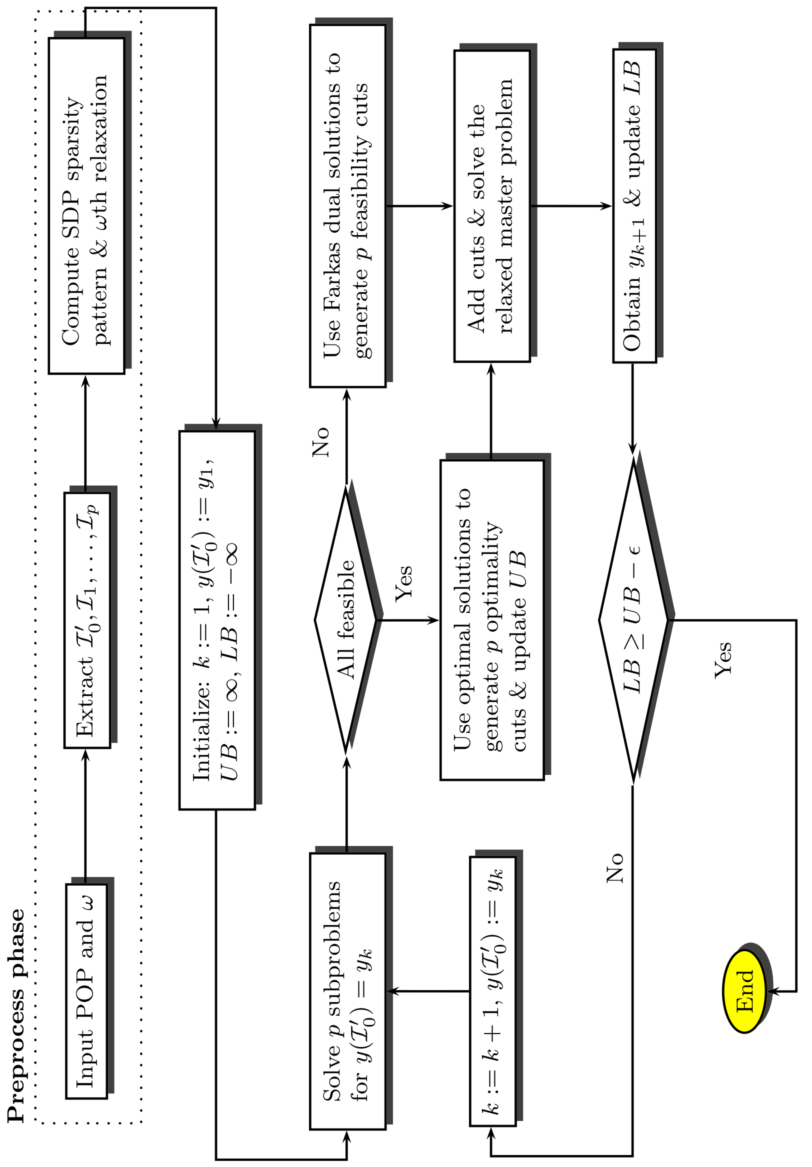

Preprocess phase
Input POP and ω
Extract $\mathcal{I}'_0, \mathcal{I}_1, \ldots, \mathcal{I}_p$
Compute SDP sparsity pattern & ωth relaxation
Initialize: $k := 1$, $y(\mathcal{I}'_0) := y_1$, $UB := \infty$, $LB := -\infty$
Solve p subproblems for $y(\mathcal{I}'_0) = y_k$
All feasible
No
Yes
Use Farkas dual solutions to generate p feasibility cuts
Use optimal solutions to generate p optimality cuts & update UB
Add cuts & solve the relaxed master problem
Obtain y_{k+1} & update LB
$LB \geq UB - \epsilon$
No
Yes
$k := k+1$, $y(\mathcal{I}'_0) := y_k$
End

Fig. 1 Decomposition-based method for sparse POPs (including preprocess phase)

Step 2.2: If all p subproblems are feasible, get the optimal objective values $\rho_i(\hat{y}_k)$ and the optimal solution vectors $\bar{\lambda}_k^i$, $i = 1, \dots, p$, and generate the optimality constraints

$$z_i \geq (-h^i - T^i y)^{\mathrm{T}} \bar{\lambda}_k^i, \quad i = 1, \dots, p. \tag{22}$$

Increase the optimality counter $n_{\mathrm{opt}} = n_{\mathrm{opt}} + 1$. Update the upper bound $\mathrm{UB} = b^{\mathrm{T}} \hat{y}_k + \sum_{i=1}^{p} \rho_i(\hat{y}_k)$ only if necessary, i.e. if the new upper bound is less than the last stored upper bound value. Go to Step 3.

Step 3: Solve the relaxed master problem

$$\begin{aligned} &\min_{y, z_1, \dots, z_p} \; b^{\mathrm{T}} y + \sum_{i=1}^{p} z_i, \\ &\text{s.t.} \quad z_i \geq (-h^i - T^i y)^{\mathrm{T}} \bar{\lambda}_m^i, \quad i = 1, \dots, p, \; m = 1, \dots, n_{\mathrm{opt}}, \\ &\qquad 0 \geq (-h^i - T^i y)^{\mathrm{T}} \bar{u}_n^i, \quad i = 1, \dots, p, \; n = 1, \dots, n_{\mathrm{feas}}, \\ &\qquad Ay + c \succeq_{\mathcal{K}^v} 0, \end{aligned} \tag{23}$$

by any suitable algorithm. Let $(\hat{y}_{k+1}, \hat{z}_{k+1}^1, \dots, \hat{z}_{k+1}^p)$ be the optimal solution. Update the lower bound: $\mathrm{LB} = b^{\mathrm{T}} \hat{y}_{k+1} + \sum_{i=1}^{p} \hat{z}_{k+1}^i$. If $\mathrm{LB} \geq \mathrm{UB} - \epsilon$, stop. Else, increase the iteration counter $k = k + 1$ and go to Step 2.

Observe that our decomposition-based method is a word-by-word extension of the Benders decomposition for linear programs [3] to SDP problems, except that we obtain several (independent) subproblems in place of one in the classical Benders decomposition.

4.2 Theoretical Convergence

Theorem 4.1 (Finite ϵ-Convergence) *Assume that $Y \cap V$ is a nonempty compact set. Then, for any given ϵ, the decomposition-based method for sparse SDP relaxations of polynomial problems terminates in a finite number of steps.*

Proof (This is based on the finite ϵ-convergence proof in [17].) We fix ϵ arbitrarily and suppose that no termination is achieved. Let $\langle z^k, y^k \rangle$ be the sequence of optimal solutions to (23) at successive iterations. We may assume that this sequence, or a subsequence, converges to a point (z^*, y^*) such that $y \in Y \cap V$, since $\langle z^k \rangle$ is a nondecreasing sequence bounded above and the sequence $\langle y^k \rangle$ belongs to the compact set $Y \cap V$. Next, at iteration $k + 1$, for k sufficiently large, the optimality constraints which we would normally generate,

$$z_i^{(k+1)} \geq (-h^i - T^i y^{(k+1)})^{\mathrm{T}} \bar{\lambda}_k^i, \quad i = 1, \dots, p, \tag{24}$$

are satisfied by the current solution due to the accumulation of constraints in (23). In addition, we may assume that the sequence of optimal multiplier vectors $\langle \lambda^i \rangle$ converges to a point λ^{i*} for each subproblem i, $i = 1, \dots, p$. Then, by the continuity of the polynomial function $f(y, \lambda^i) = (-h^i - T^i y)^{\mathrm{T}} \lambda^i$, we have

$$z_i^* \geq (-h^i - T^i y^*)^{\mathrm{T}} \lambda^{i*}, \quad i = 1, \dots, p. \tag{25}$$

In what follows, let us call $\Lambda^i(y)$ the set of optimal solutions to the dual subproblem (11). In other words, for each dual subproblem i, $i = 1, \dots, p$, the set $\Lambda^i(y)$ consists of all points λ^i such that $f(y, \lambda^i) = \rho_i(y)$. Again by the continuity of the function $f(y, \lambda^i)$, we may apply Theorem 1.5 of [18] in order to show that the sets $\Lambda^i(y)$ are upper-semicontinuous mappings at y^*.[8]

The upper semicontinuity of the set $\Lambda^i(y)$ at y^* implies that $\lambda^{i*} \in \Lambda^i(y^*)$, $i = 1, \dots, p$. The former conclusion yields $z_i^* \geq \rho_i(y^*)$, $\forall i$, or $\sum_{i=1}^p z_i^* \geq \rho(y^*)$, and this in turn implies $b^{\mathrm{T}} y^* + \sum_{i=1}^p z_i^* \geq b^{\mathrm{T}} y^* + \rho(y^*)$. By the upper semicontinuity of function $\rho(y)$, we finally conclude that

$$b^{\mathrm{T}} y^k + \sum_{i=1}^p z_i^k + \epsilon \geq b^{\mathrm{T}} y^k + \rho(y^k), \tag{26}$$

which is equivalent to our termination criterion, i.e. $LB \geq UB - \epsilon$. As a result, our supposition that the termination criterion is not met was proved false. □

Observe that Theorem 4.1 assumes that every fixation of coupling moment variables y yields feasible subproblems. Namely, it is assumed implicitly that only the optimality constraints are added to the relaxed master problem at each iteration. However, in practice not all fixations produce feasible subproblems. To make matters worse, the consecutive addition of feasibility constraints may prevent the procedure from converging. A failure of convergence is also met at the generalized Benders decomposition, for the convergence is based on the assumption that either Y[9] is finite or that optimality constraints are generated for every fixation of y. To overcome such problematic situation, Grothey et al. suggested a procedure, called *feasibility restoration*, that guarantees convergence even in the presence of feasibility constraints [19]. We extended this work such that it applies to semidefinite programming and included it to our decomposition-based method. The feasibility restoration and our modified algorithm are described next.

4.3 Feasibility Restoration

As pointed out earlier, the consecutive addition of feasibility constraints may cause failure of convergence. To rectify this situation, we amend our procedure as follows.

[8] The definition of the upper semicontinuity of a point-to-set mapping can be found in [18].

[9] Y is the set to which the vector variable y belongs.

Recall how the ith subproblem (10) would be formulated for $y=\hat{y}_k$,

$$\min_{y^i} {d^i}^{\mathrm{T}} y^i, \quad \text{s.t.} \quad W^i y^i \succeq_{\mathcal{K}^{m^i}} c^i_k, \tag{27}$$

where

$$c^i_k = (-h^i - T^i \hat{y}_k) \in \mathbb{R}^{{m^i}^2}, \quad i=1,\dots,p.$$

If all subproblems were feasible, the SDP solver would compute a dual optimal solution $\bar{\lambda}^i_k$ for each subproblem. The solver internally would also compute a feasible primal optimal solution $(\bar{y}^i_k, \bar{s}^i_k)$ satisfying the corresponding primal constraints,

$$W^i \bar{y}^i_k - \bar{s}^i_k = c^i_k, \quad \bar{s}^i_k \succeq_{\mathcal{K}^{m^i}} 0.$$

On the other hand, if all subproblems were infeasible, the SDP solver would certify infeasibility by computing a Farkas dual solution $\bar{u}^i_k \succeq_{\mathcal{K}^{m^i}} 0$ such that ${W^i}^{\mathrm{T}} \bar{u}^i_k = 0$ for each subproblem. The solver would still compute a primal solution $(\bar{y}^i_k, \bar{s}^i_k)$ for each subproblem (27), but such a solution would not satisfy the corresponding primal constraints, i.e.

$$W^i \bar{y}^i_k - \bar{s}^i_k \neq c^i_k, \quad \bar{s}^i_k \succeq_{\mathcal{K}^{m^i}} 0.$$

Instead, the latter solution would satisfy a relaxed set of constraints such as

$$W^i \bar{y}^i_k - \bar{s}^i_k = \hat{c}^i_k, \tag{28}$$

$$\bar{s}^i_k \succeq_{\mathcal{K}^{m^i}} 0. \tag{29}$$

The above constraints give rise to the construction of feasible subproblems. In other words, from each (infeasible) subproblem (27), we exploit the primal information to compute the modified right-hand side $\hat{c}^i_k$ using (28). Then, we construct the corresponding relaxed ith subproblem,

$$\min_{y^i} {d^i}^{\mathrm{T}} y^i, \quad \text{s.t.} \quad W^i y^i \succeq_{\mathcal{K}^{m^i}} \hat{c}^i_k - \alpha^i \operatorname{vec}(I_{m^i}), \tag{30}$$

which by construction is feasible. By subtracting $\alpha^i \operatorname{vec}(I_{m^i})$ from $\hat{c}^i_k$, where $\alpha^i > 0$ and $I_{m^i} \in \mathbb{R}^{m \times m}$ the identity matrix, we ensure the existence of a strictly feasible solution. Namely, the Slater-type regularity condition is satisfied and hence strong duality between the primal and dual formulations holds [20]. Finally, the dual optimal solutions $\hat{\bar{\lambda}}^i_k$ from the p auxiliary subproblems (30) are used to generate p optimality constraints,

$$z_i \geq (-h^i - T^i y)^{\mathrm{T}} \hat{\bar{\lambda}}^i_k, \quad i=1,\dots,p. \tag{31}$$

To sum up, each time infeasibility is met, not only the p feasibility constraints from the Farkas dual solutions of the original p subproblems (27) are generated, but also the p optimality constraints from the p auxiliary subproblems (30). Such an amendment yields the relaxed master problem below and our modified algorithm is briefly stated in Algorithm 3.

$$\begin{aligned}
&\min_{y,z_1,\dots,z_p} \; b^{\mathrm{T}}y + \sum_{i=1}^{p} z_i,\\
&\text{s.t.}\quad z_i \ge (-h^i - T^i y)^{\mathrm{T}}\bar{\lambda}^i_m, \quad i=1,\dots,p,\; m=1,\dots,n_{\mathrm{opt}},\\
&\qquad\; z_i \ge (-h^i - T^i y)^{\mathrm{T}}\hat{\bar{\lambda}}^i_n, \quad i=1,\dots,p,\; n=1,\dots,n_{\mathrm{feas}},\\
&\qquad\; 0 \ge (-h^i - T^i y)^{\mathrm{T}}\bar{u}^i_n, \quad i=1,\dots,p,\; n=1,\dots,n_{\mathrm{feas}},\\
&\qquad\; Ay + c \succeq_{\mathcal{K}^v} 0.
\end{aligned} \tag{32}$$

Algorithm 3 (Decomposition-based Method with Feasibility Restoration)

Step 1: The same as in Algorithm 2, i.e. initialize.

Step 2: The same as in Algorithm 2, i.e. solve p subproblems (10) for $y := \hat{y}_k$.

Step 2.1: If all the subproblems are infeasible, generate p feasibility constraints from (27). Compute $\hat{c}^i_k$, $i = 1,\dots,p$, and solve p auxiliary subproblems (30) to generate p optimality constraints (31). Add both types of constraints to the relaxed master problem and increase the feasibility counter $n_{\mathrm{feas}} := n_{\mathrm{feas}} + 1$. Go to Step 3.

Step 2.2: The same as in Algorithm 2.

Step 3: The same as in Algorithm 2, except that the amended relaxed master problem (32) is solved in place of the relaxed master problem (23).

5 Implementation

Our method was implemented in C++ and several essential tools were incorporated. To begin with, our program reads as input a file in *GAMS scalar* format. The input files were found in [21]. After the input file is read and parsed, we employ a set of functions from SparsePOP [6] in order to extract the sparsity pattern of the input polynomial problem, as well as to generate the sparse semidefinite relaxation. We also use a set of functions for permuting and factorizing symbolically sparse matrices. This set of functions is part of CHOLMOD [22].

After the sparse semidefinite relaxation is computed and the polynomial sparsity pattern is obtained, several routines were implemented in order to determine the coupling moment variables, the p disjoint sets of independent moment variables and the data that correspond to the relaxed master and each subproblem.

At each iteration of our main algorithm, we solve the SDP subproblems with the CSDP solver [23], which not only outputs the solution when found, but also the Farkas dual solution in case of infeasibility.[10] Finally, in order to solve the SDP relaxed master problem at each iteration, we employ the CSDP solver [23].

[10]All well-known SDP solvers such as DSDP [24], SDPA [25] and SeDuMi [26] compute the Farkas dual solution in case of infeasibility, since it serves as a certificate of infeasibility.

Furthermore, we compare our results with the results obtained if `CSDP` were used in place of our decomposition-based method for solving the computed sparse SDP relaxation of the POP. To compute the sparse SDP relaxations, we employ a set of functions from `SparsePOP`. However, the `SparsePOP` solver is mainly implemented in Matlab and calls `SeDuMi` to solve the computed sparse SDP relaxations. We considered it more convenient to have everything implemented in C++; for this reason, we replicated `SparsePOP` usage in C++, using `CSDP` in place of `SeDuMi`. From this point onward, we refer to our C++ version of `SparsePOP` as `SparsePOP/CSDP`. Hence, let us refer to the optimal objective value computed by our decomposition-based method as p^*_ω and to the one computed by `SparsePOP/CSDP` as $p^*_{\omega,\mathrm{b}mrk}$. Similarly, let us call the optimal solution vector produced by our method as x^* and the one computed by `SparsePOP/CSDP` as $x^*_{\mathrm{b}mrk}$. To evaluate the accuracy of our solution against the benchmark solution, the following metrics were used:

$$\epsilon_{p^*} = \frac{|\, p^*_{\omega,\mathrm{b}mrk} - p^*_\omega \,|}{\max\{1, |\, p^*_{\omega,\mathrm{b}mrk} \,|\}}, \qquad \epsilon_{x^*} = \max\left\{ \frac{|\, x^*_{i,\mathrm{b}mrk} - x^*_i \,|}{\max\{1, |\, x^*_{i,\mathrm{b}mrk} \,|\}} \right\}, \tag{33}$$

where x^*_i ($x^*_{i,\mathrm{b}mrk}$) corresponds to the ith element of the vector x^* ($x^*_{\mathrm{b}mrk}$).

6 Computational Experience

Let us consider Example 3.1. Applying our method to its first and second sparse SDP relaxations, i.e. $\omega = 1$ and $\omega = 2$ respectively, we get the convergent bounds presented in Fig. 2. Recall that the global optimal solution of this example is $p^* = -213$, with its first SDP relaxation giving a lower bound equal to $p^*_1 = -214$ and its second relaxation giving the global optimal solution, i.e. $p^*_2 = p^* = -213$. Figure 3 is another graphical example of convergent bounds computed by our method and it corresponds to test problem st_e21. In particular, we tested our method to a collection

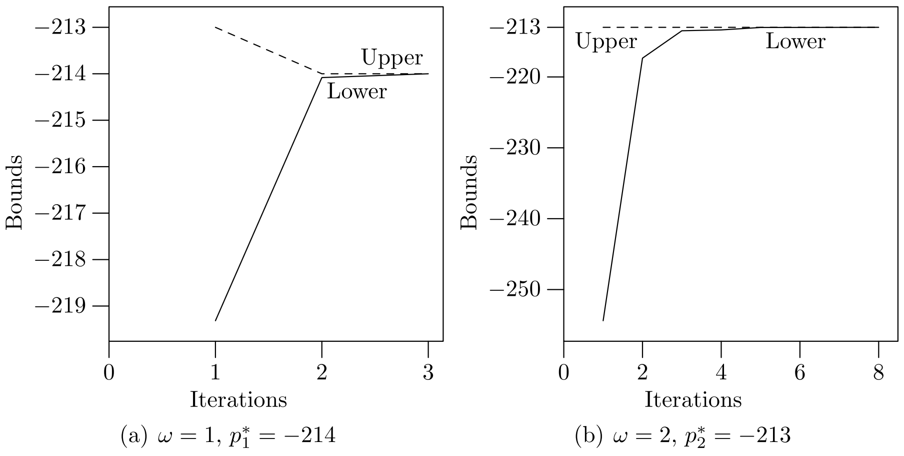
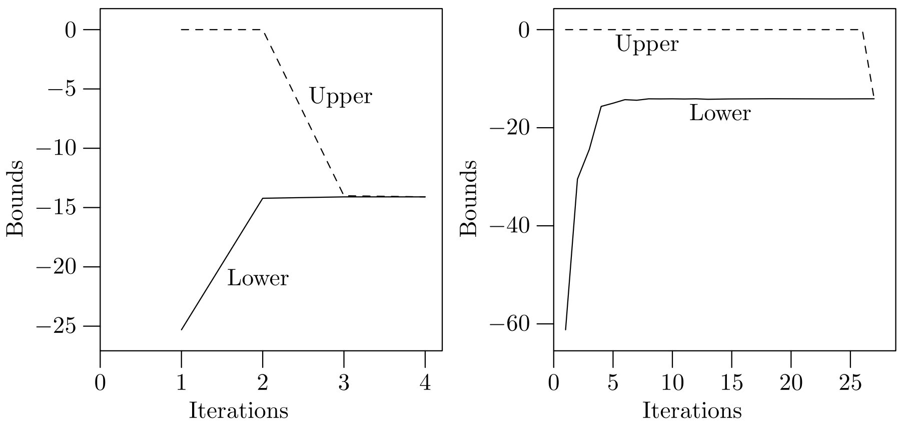


Fig. 2 Convergent bounds for Example 3.1 (Bex2_1_2)



(a) $\omega = 1$, $p_1^* = -14.1$ (b) $\omega = 2$, $p_2^* = -14.1$

Fig. 3 Convergent bounds for test problem st_e21

of polynomial optimization problems taken from [21] and some indicative results are presented in Table 3.[11] For a detailed list of numerical results, the reader is referred to [27]. The first two columns of Table 3 show the problem dimensions, i.e. n is the number of polynomial variables and m is the number of constraints, while column ω records the order of the sparse SDP relaxation used. The following three columns hold the number of iterations and the values of metrics stated in (33).

Remarks Our code should be able to compute the value that the benchmark solver `SparsePOP/CSDP` computes. Table 3 demonstrates that this is true in all cases with good accuracy. However, as the theory of SDP relaxation technique implies, the solution of the relaxation is not always the global optimal solution of the POP. Consequently, the same applies to our solution. However, as we increase the relaxation order, we will be able to compute a better and better approximation of the global optimal solution of the POP.

Moreover, several numerical issues were met. The most important one was the difficulty in computing a starting feasible point. It was observed that, when the starting point made all subproblems infeasible, then in most of the cases the relaxed master problem at the first iteration was unbounded and our procedure was terminated. As a remedy to this problem, an interesting algorithm is intended to be incorporated as a Phase 1 between the preprocess phase and the decomposition-based method. The algorithm will be a modification of a similar Phase 1 algorithm discussed in [28]. The latter algorithm is also iterative and requires the solution of an optimization problem and the computation of an approximate analytic center at each iteration. Other numerical issues were met when `CSDP` did not make any progress in solving one or more subproblems or was stuck at the edge of dual feasibility. In such cases, our procedure was terminated. We would like to examine whether or not the use of another SDP

[11]Note that test problem Bex2_1_2 is Example 3.1.

Table 3 Decomposition-based method on test problems

Problem	n	m	ω	Iters	ϵ_{p^*}	ϵ_{x^*}
Bex2_1_2	6	2	1	3	1.24e–09	1.54e–07
Bex2_1_2	6	2	2	8	1.2e–08	2.25e–07
Bex2_1_3	13	9	1	6	7.35e–09	1.1e–08
Bex2_1_8	24	10	1	4	1.17e–08	0.0023
Bex3_1_1	8	6	2	165	8.92e–06	0.00232
Bex3_1_1	8	6	3	152	7.45e–06	0.00101
Bex5_2_2_case1	9	6	1	9	6.04e–09	1.39e–06
Bex5_3_2	22	16	1	3	1.4e–10	0.667
Bex5_4_2	8	6	2	114	1.74e–07	2.1e–05
Bex9_1_1	13	12	1	10	5.04e–09	0.013
Bex9_1_1	13	12	2	99	7.56e–05	0.594
Bex9_1_2	10	9	1	12	1.13e–09	0.00121
Bex9_1_4	10	9	1	11	8.93e–06	0.000119
Bex9_1_5	13	12	1	2	2.33e–09	0.0808
Bex9_1_5	13	12	2	24	5.17e–09	0.0451
Bex9_1_8	14	12	1	5	6.79e–11	0.0257
Bex9_2_1	10	9	1	2	1.028e–07	9.192e–06
Bex9_2_2	10	11	1	4	7.27e–06	8.03e–05
Bex9_2_4	8	7	1	3	6.73e–10	0.0441
Bex9_2_4	8	7	2	36	3.41e–09	4.33e–06
Bex9_2_5	8	7	1	1	2.43e–09	3.73e–07
Bex9_2_8	6	5	1	1	1.62e–09	2.42e–08
Bex9_2_8	6	5	2	6	6.94e–09	3.72e–06
Balkyl	14	7	3	121	9.76e–05	0.0067
Bhaverly	12	9	1	8	2.06e–05	0.000218
Bst_e05	5	3	1	2	9.61e–06	0.432
Bst_e05	5	3	2	143	4.42e–06	0.000258
Bst_e07	10	7	1	7	1.49e–09	3.7e–08
Bst_e07	10	7	2	83	1.78e–07	0.000507
st_e21	6	6	1	4	3.88e–09	1.83e–09
st_e21	6	6	2	25	1.12e–08	2.93e–08
Bst_bpaf1a	10	10	1	6	4.68e–09	1.1e–05
Bst_bpaf1b	10	10	1	7	2.04e–09	1.72e–08
st_glmp_kk90	5	7	2	4	8.66e–09	3.44e–09
st_glmp_kk90	5	7	3	5	4.93e–08	7.1e–09

solver could reduce some of these numerical problems. In addition, there were some cases where, although the subproblems were either solved to optimality or infeasibility was detected and the corresponding cuts were added, little progress did happen in the increment of the lower bound toward the optimal solution of the relaxation. On the contrary, the upper bound tended to reach the optimal solution of the relaxation quite early in the process. Nevertheless, the slow progress in the lower bound slowed down the convergence between the lower and upper bounds. What is more, in cases where feasibility cuts were only added at some point onward, convergence was either extremely slow or not possible.[12] This does not contradict finite ϵ-convergence shown in Theorem 4.1, since its proof is based on adding optimality constraints only. Such a situation also appears in [17]. In practice, the addition of feasibility cuts does not usually prevent the procedure from converging. However, in cases where convergence appears to be very slow, the *feasibility restoration* is essential to ensure convergence.

7 Discussion

In this work we deal with polynomial problems with a sparsity pattern, which has been shown to be inherited into their sparse SDP relaxations. By exploiting this sparsity pattern, we apply a decomposition method to the sparse SDP relaxations aiming at solving polynomials problems. At each iteration of our algorithm, we solve p subproblems and the relaxed master problem. The subproblems give a sequence of upper bounds and the relaxed master problems give a sequence of lower bounds. Our procedure terminates when these bounds are very close.

Many ideas have been raised from this work and remain to be examined. Firstly, a difficulty often met was the computation of a starting point to make all subproblems feasible. The remedy to this situation is the incorporation of a Phase 1 algorithm, such as the one discussed in [28]. This is an iterative algorithm and requires the solution of an optimization problem and the computation of an approximate analytic center at each iteration. We intend to extend it for the purposes of our work. Next, to improve the performance of our algorithm we intend to exploit the fact that the subproblems are solved independently at each iteration. At the moment, this property has been developed in a sequential algorithm. We intend to explore how our algorithm could be converted into a parallel algorithm hoping to gain in terms of efficiency and time. A similar work has been carried out in [29, 30]. Lastly, a very interesting extension of our algorithm is to separate it from the POP framework and amend it accordingly so as to make it directly applicable to a sparse SDP problem. This would yield a decomposition algorithm for sparse SDPs and would potentially be very useful in large-scale semidefinite programming. To achieve the separation of the decomposition-based method from the POP, we should aim at detecting and extracting the sparsity pattern of the SDP problem independently of any POP it may approximate. If the SDP sparsity pattern can be detected independently, then the decomposition-based method is ready for application in sparse semidefinite programming.

[12]The maximum number of iterations was reached and the procedure was terminated. The maximum number of iterations was set to 500.

References

1. Lasserre, J.B.: Convergent SDP-relaxations in polynomial optimization with sparsity. SIAM J. Optim. **17**(3), 822–843 (2006)
2. Waki, H., Kim, S., Kojima, M., Muramatsu, M.: Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. SIAM J. Optim. **17**(1), 218–242 (2006) (electronic)
3. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Comput. Manag. Sci. **2**(1), 3–19 (2005). Reprinted from Numer. Math. **4**, 238–252 (1962)
4. Ramana, M., Goldman, A.J.: Some geometric results in semidefinite programming. J. Glob. Optim. **7**(1), 33–50 (1995)
5. Kim, S., Kojima, M., Toint, P.: Recognizing underlying sparsity in optimization. Technical Report, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology (2006)
6. Waki, H., Kim, S., Kojima, M., Muramatsu, M.: SparsePOP: a sparse semidefinite programming relaxation of polynomial optimization problems. ACM Trans. Math. Softw. **35**(2) (2008)
7. Alizadeh, F.: Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J. Optim. **5**(1), 13–51 (1995)
8. Pardalos, P.M., Wolkowicz, H. (eds.): Topics in Semidefinite and Interior-point Methods. Fields Institute Communications, vol. 18. American Mathematical Society, Providence (1998)
9. Ramana, M.V., Pardalos, P.M.: Semidefinite programming. In: Interior Point Methods of Mathematical Programming. Appl. Optim., vol. 5, pp. 369–398. Kluwer Academic, Dordrecht (1996)
10. Pataki, G.: The geometry of semidefinite programming. In: Handbook of Semidefinite Programming. Internat. Ser. Oper. Res. Management Sci., vol. 27, pp. 29–65. Kluwer Academic, Dordrecht (2000)
11. Yajima, Y., Ramana, M.V., Pardalos, P.M.: Cuts and semidefinite relaxations for nonconvex quadratic problems. In: Generalized Convexity and Generalized Monotonicity, Karlovassi, 1999. Lecture Notes in Econom. and Math. Systems, vol. 502, pp. 48–70. Springer, Berlin (2001)
12. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM J. Optim. **11**(3), 796–817 (2000/01)
13. Parrilo, P.A.: Semidefinite programming relaxations for semialgebraic problems. Math. Program., Ser. B **96**(2), 293–320 (2003). Algebraic and geometric methods in discrete optimization
14. Laurent, M.: Sums of squares, moment matrices and optimization over polynomials. In: Emerging Applications of Algebraic Geometry. IMA Vol. Math. Appl., vol. 149, pp. 157–270. Springer, New York (2009)
15. Floudas, C.A., Pardalos, P.M.: A collection of test problems for constrained global optimization algorithms. Lecture Notes in Computer Science, vol. 455. Springer, Berlin (1990)
16. LP_SOLVE Description (2008). tech.groups.yahoo.com/group/lp_solve
17. Geoffrion, A.M.: Generalized Benders decomposition. J. Optim. Theory Appl. **10**, 237–260 (1972)
18. Meyer, R.: The validity of a family of optimization methods. SIAM J. Control Optim. **8**, 41–54 (1970)
19. Grothey, A., Leyffer, S., Mckinnon, K.I.M.: A note on feasibility in benders decomposition (1999)
20. Helmberg, C., Rendl, F., Vanderbei, R.J., Wolkowicz, H.: An interior point method for semidefinite programming. SIAM J. Optim. **6**(2), 342–361 (1996)
21. GLOBAL Lib: www.gamsworld.org/global/globallib/globalstat.htm (2008)
22. Davis, T.A.: User Guide for CHOLMOD: a sparse Cholesky factorization and modification package. Technical Report, Dept. of Computer and Information Science and Engineering, University of Florida (2006)
23. Borchers, B.: CSDP, A C library for semidefinite programming. Optim. Methods Softw. **11**, 613–623 (1999)
24. Benson, S.J., Ye, Y.: DSDP5: Software for semidefinite programming. Technical Report, Mathematics and Computer Science Division, Argonne National Laboratory (2005)
25. Fujisawa, K., Kojima, M., Nakata, K., Yamashita, M.: SDPA (SemiDefinite Programming Algorithm) User's Manual—Version 5.01. Technical Report, Tokyo Institute of Technology (1995)
26. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11/12**(1–4), 625–653 (1999)
27. Kleniati, P.M.: Decomposition Schemes for Polynomial Optimization, Semidefinite Programming and Applications to Nonconvex Portfolio Decisions. Ph.D. Thesis, Imperial College London (2009)

28. Zhao, G.: A log-barrier method with Benders decomposition for solving two-stage stochastic linear programs. Math. Program., Ser. A **90**(3), 507–536 (2001)
29. Sivaramakrishnan, K.K., Plaza, G., Terlaky, T.: A conic interior point decomposition approach for large scale semidefinite programming (2005)
30. Sivaramakrishnan, K.K.: A parallel interior point decomposition algorithm for block angular semidefinite programs. Comput. Optim. Appl. (2008)