ISSN 1342-2804

Research Reports on Mathematical and Computing Sciences

Solving Polynomial Least Squares Problems via Semidefinite Programming Relaxations

Sunyoung Kim and Masakazu Kojima

August 2007, B–444

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES B: Operations Research

B-444 **Solving polynomial least squares problems via semidefinite programming relaxations**

Sunyoung Kim[⋆] and Masakazu Kojima[†]

August 2007

Abstract.
A polynomial optimization problem whose objective function is represented as a sum of positive and even powers of polynomials, called a polynomial least squares problem, is considered. Methods to transform a polynomial least squares problem to polynomial semidefinite programs to reduce degrees of the polynomials are discussed. Computational efficiency of solving the original polynomial least squares problem and the transformed polynomial semidefinite programs is compared. Numerical results on selected polynomial least squares problems show better computational performance of a transformed polynomial semidefinite program, especially when degrees of the polynomials are larger.

Key words.
Nonconvex optimization problems, polynomial least squares problems, polynomial semidefinite programs, polynomial second-order cone programs, sparsity.

⋆ Department of Mathematics, Ewha W. University, 11-1 Dahyun-dong, Sudaemoon-gu, Seoul 120-750 Korea. The research was supported by Kosef R01-2005-000-10271-0 and KRF-2006-312-C00062.
skim@ewha.ac.kr

† Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. The research was supported by by Grant-in-Aid for Scientific Research (B) 19310096.
kojima@is.titech.ac.jp

1 Introduction

We consider solving a polynomial least squares problem

$$\text{minimize } \sum_{i \in M} f_i(\boldsymbol{x})^{2p_i}, \tag{1}$$

where $f_i(\boldsymbol{x})$ $(i \in M)$ are polynomials in $\boldsymbol{x} \in R^n$, $p_i \in \{1, 2, \dots\}$ $(i \in M)$ and $M = \{1, 2, \dots, m\}$. The problem (1) is a polynomial optimization problem (POP) with an objective function represented as a sum of positive and even powers of polynomials. In particular, if $p_i = 1$ $(i \in M)$, the problem (1) becomes a standard nonlinear least squares problem:

$$\text{minimize } \sum_{i \in M} f_i(\boldsymbol{x})^2. \tag{2}$$

The nonlinear least squares problem (2) has been studied extensively and many methods have been proposed. Popular approaches for nonlinear least squares problems are the Gauss-Newton and the Levenberg-Marquardt methods, which find a local (not global in general) minimum of (2). See, for example, [25]. As opposed to finding a local minimum of (2) in those existing methods, we propose global approaches for a more general form (1) of polynomial least squares problems.

The number of variables, the degree of polynomials, and the sparsity of polynomials of the problem (1) determine its solvability as a POP. Solving the least squares problem (1) using the semidefinite programming (SDP) relaxation proposed by Lasserre [19], which is called the dense SDP relaxation in this paper, is so expensive that only small to some medium-sized problems can be handled, despite the powerful convergence result in theory. A sparse SDP relaxation for solving correlatively sparse POPs was proposed in [31] to overcome this computational difficulty, and shown to be very effective in solving some large-scale POPs. Unconstrained POPs with the correlative sparsity could be solved up to $n = 1000$ by the sparse SDP relaxation in [31]. The convergence result of the sparse SDP relaxation applied to correlatively sparse POPs in [20] supports the use of the sparse SDP relaxation. We should mention that the sparse SDP relaxation provides less accurate solutions than the dense SDP relaxation in general. Exploiting the sparsity of polynomials is, nevertheless, essential when solving large-scale POPs. If the sparsity is not utilized, the size and the degree of polynomial optimization problems that can be solved is limited to small and medium-sized problems.

Most of computational challenges for solving POPs come from the fact that the size of the resulting SDP relaxation problem is too large to handle with SDP solvers such as CSDP [2], SDPA [4], SDPT3 [28], and SeDuMi [27]. Various techniques thus have been introduced to increase the size of problems that can be solved. The sparsity of POPs was utilized to reduce the size of the resulting SDP relaxation problems [14, 31]. Transformation of POPs to easy-to-handle formulations for a certain class of problems was also studied. For instance, it is shown in [13] that second-order cone programming can be used efficiently for a class of convex POPs.

The problem (1) can be transformed to a polynomial SDP, *i.e.* a problem of minimizing a polynomial objective function subject to polynomial matrix inequalities, to improve computational efficiency. Although polynomial SDPs arise in many applications in system

and control theory, their global optimization has not been dealt with extensively. Recently, solving polynomial SDPs with the use of SDP relaxations has been studied in [7, 8, 15]. The aim of this paper is to show how (1) is transformed to various polynomial SDPs and to compare the computational performance of solving the transformed problems with solving the problem (1) itself. We also present an efficient polynomial SDP formulation among them. In both the original and transformed formulations, valid polynomial matrix inequalities are added to construct a polynomial SDP of increased size and the resulting polynomial SDP is linearized, which is then solved by a primal-dual interior-point method. We discuss the effects of the sparsity, the size of SDP blocks, and the size of the coefficient matrix of the linearized SDP on the computational performance.

Solving the original problem is compared with solving a transformed polynomial SDP numerically using SparsePOP [30]. Recent advancement in the study of POPs has accompanied by software packages implementing solution methods for POPs. SOStools [26], GloptiPoly [6], and SparsePOP are developed currently. SparsePOP is a collection of matlab modules utilizing the correlative sparsity structure of polynomials. The size of SDP created by SparsePOP is thus smaller than that of GloptiPoly, which makes it possible to solve larger-sized problems.

This paper is organized as follows: After introducing symbols and notation, we present several ways of formulating the problem (1) as polynomial SDPs in Section 2. In Section 3, a sparse SDP relaxation of a polynomial SDP formulation is described. Section 4 includes comparison of various polynomial SDP formulations in terms of degrees of the polynomials, the sparsity, the size of the resulting SDPs, and the relaxation orders used to solve the polynomial SDPs. In Section 5, numerical experiments are shown. Concluding remarks are presented in Section 6.

2 Various formulations of the polynomial least squares problems

2.1 A sparse POP formulation

Let $\mathbb{R}^n$, $\mathbb{Z}_+$ and $\mathbb{Z}^n_+$ denote the n-dimensional Euclidean space, the set of nonnegative integer numbers and the set of n-dimensional nonnegative integer vectors, respectively. For every $\boldsymbol{\alpha} \in \mathbb{Z}^n_+$ and every $\boldsymbol{x} = (x_1.x_2, \ldots, x_n) \in \mathbb{R}^n$, $\boldsymbol{x}^{\boldsymbol{\alpha}}$ denotes a monomial $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. Let us denote $\mathcal{S}^r$ and $\mathcal{S}^r_+$ the space of $r \times r$ symmetric matrices and the cone of $r \times r$ positive semidefinite symmetric matrices, respectively. We use the notation $\boldsymbol{S} \succeq \boldsymbol{O}$ to mean $\boldsymbol{S} \in \mathcal{S}^r_+$. Let $N = \{1, 2, \ldots, n\}$, $M = \{1, 2, \ldots, m\}$, and $C_i \subseteq N$ $(i \in M)$. The sparsity of polynomials in the polynomial least squares problem (1) is represented using $C_i \subseteq N$. Let $\boldsymbol{x}_{C_i} = (x_j : j \in C_i)$ $(i \in M)$ the column vector variable of the elements x_j, and $\mathbb{R}^{C_i}$ the $\#C_i$-dimensional Euclidean space of the vector variable $\boldsymbol{x}_{C_i}$. We assume that each $f_i(\boldsymbol{x})$ is a polynomial in variables x_j $(j \in C_i)$, and use the notation $f_i(\boldsymbol{x}_{C_i})$ instead of $f_i(\boldsymbol{x})$ $(i \in M)$. Then, (1) can be written as

$$\text{minimize} \sum_{i \in M} f_i(\boldsymbol{x}_{C_i})^{2p_i}. \tag{3}$$

We call (3) a sparse POP formulation of the polynomial least squares problem (1).

2.2 Polynomial SDP formulations of the polynomial least squares problem

A different approach of solving (3) is formulating the problem as a polynomial SDP whose degree is lower than (3). For description of a polynomial SDP, let $\mathcal{F}$ be a nonempty finite subset of $\mathbb{Z}_+^{n'}$ for some $n' \geq n$, $N' = \{1, \ldots, n'\}$, and $\boldsymbol{F}_{\boldsymbol{\alpha}} \in \mathcal{S}^r$ $(\boldsymbol{\alpha} \in \mathcal{F})$. A polynomial $\boldsymbol{F}(\boldsymbol{y}_{C'})$ of $\boldsymbol{y}_{C'} = (y_j : j \in C')$, for some $C' \subseteq N'$, with coefficients $\boldsymbol{F}_{\boldsymbol{\alpha}} \in \mathcal{S}^r$ $(\boldsymbol{\alpha} \in \mathcal{S}^r)$ is written as

$$\boldsymbol{F}(\boldsymbol{y}_{C'}) = \sum_{\boldsymbol{\alpha} \in \mathcal{F}} \boldsymbol{F}_{\boldsymbol{\alpha}} \boldsymbol{y}_{C'}^{\boldsymbol{\alpha}}. \tag{4}$$

We call $\boldsymbol{F}(\boldsymbol{y}_{C'})$ *a symmetric polynomial matrix*, and $\mathcal{F}$ *a support* of $\boldsymbol{F}(\boldsymbol{y}_{C'})$ if $\boldsymbol{F}(\boldsymbol{y}_{C'})$ is represented as (4). Note that each element $F_{k\ell}(\boldsymbol{y}_{C'})$ of $\boldsymbol{F}(\boldsymbol{y}_{C'})$ is a real-valued polynomial in $\boldsymbol{y}_{C'}$ and that $F_{k\ell}(\boldsymbol{y}_{C'}) = F_{\ell k}(\boldsymbol{y}_{C'})$ $(1 \leq k < \ell \leq r)$. When $r = 1$, $\boldsymbol{F}(\boldsymbol{y}_{C'})$ coincides with a real-valued polynomial in $\boldsymbol{y}_{C'}$.

Let $K = \{1, \ldots, m'\} = K_o \cup K_c$ for some $m' \in \mathbb{Z}_+$, $C_i' \subseteq N'$ $(i \in K)$, and let $\boldsymbol{F}_i(\boldsymbol{y}_{C_i'})$ be a symmetric polynomial matrix with $r_i \times r_i$ coefficient matrices $(i \in K_c)$. Then, a polynomial SDP can be described as

$$\text{minimize } \sum_{j \in K_o} g_j(\boldsymbol{y}_{C_j'}) \text{ subject to } \boldsymbol{F}_i(\boldsymbol{y}_{C_i'}) \succeq \boldsymbol{O} \ \ (i \in K_c), \tag{5}$$

We may regard the sparse POP formulation (3) of the polynomial least squares problem as a special case of (5) where we take $n' = n$, $m' = m$, $N' = N$, $K = K_o = M$, $C_i' = C_i$ $(i \in K)$, $g_j(\boldsymbol{y}_{C_j'}) = f_j(\boldsymbol{x}_{C_j})^{p_j}$ $(j \in K_o)$ and $K_c = \emptyset$.

To derive polynomial SDPs which are equivalent to the polynomial least squares problem (3), we utilize a special case of the so-called Schur complement relation:

$$s_1 s_2 \geq \boldsymbol{w}^T \boldsymbol{w}, \ s_1 \geq 0 \ \text{ and } s_2 \geq 0 \text{ if and only if } \begin{pmatrix} s_1 \boldsymbol{I} & \boldsymbol{w} \\ \boldsymbol{w}^T & s_2 \end{pmatrix} \succeq \boldsymbol{O} \tag{6}$$

holds for every $s_1 \in \mathbb{R}$, $s_2 \in \mathbb{R}$ and $\boldsymbol{w} \in \mathbb{R}^k$, where $\boldsymbol{I}$ denotes the $k \times k$ identity matrix. By letting $k = 1$, $s_1 = 1$, $s_2 = t_i$ and $\boldsymbol{w} = f_i(\boldsymbol{x}_{C_i})$, it follows that

$$t_i \geq f_i(\boldsymbol{x}_{C_i})^2 \ \text{ if and only if } \ \begin{pmatrix} 1 & f_i(\boldsymbol{x}_{C_i}) \\ f_i(\boldsymbol{x}_{C_i}) & t_i \end{pmatrix} \succeq \boldsymbol{O}$$

holds for every $i \in M$. Using this equivalence, we can transform the polynomial least squares problem (3) into the following equivalent polynomial SDP:

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j \in M} t_j^{p_j} \\ \text{subject to} & \begin{pmatrix} 1 & f_i(\boldsymbol{x}_{C_j}) \\ f_i(\boldsymbol{x}_{C_j}) & t_i \end{pmatrix} \succeq \boldsymbol{O} \ (j \in M). \end{array}\right\} \tag{7}$$

The problem (7) can be represented in the form of (5) if we let $n' = n + m$, $m' = m$, $N' = \{1, \ldots, n'\}$, $K = K_o = K_c = M$, $C_i' = C_i \cup \{n + i\}$ $(i \in K)$, $g_j(\boldsymbol{y}_{C_j'}) = y_{n+j}^{p_j}$ $(j \in K_o)$ and

$$\boldsymbol{F}_i(\boldsymbol{y}_{C_i'}) = \begin{pmatrix} 1 & f_i(\boldsymbol{x}_{C_i}) \\ f_i(\boldsymbol{x}_{C_i}) & t_i \end{pmatrix} \ (i \in K_c).$$

The equivalence between (3) and the polynomial SDP (7) can be shown as Lemma 2.1.

Lemma 2.1. *The POP (3) is equivalent to the polynomial SDP (7).*

Proof: Suppose that $v = \sum_{i\in M} f_i(\boldsymbol{x}_{C_i})^{2p_i}$. Let $t_i = f_i(\boldsymbol{x}_{C_i})^2$ $(i \in M)$. Then $(\boldsymbol{x}, \boldsymbol{t}) \in \mathbb{R}^{n+m}$ is a feasible solution of the polynomial SDP (7) which attains the objective value v. Conversely, suppose that $(\boldsymbol{x}, \boldsymbol{t}) \in \mathbb{R}^{n+m}$ is a feasible solution of the polynomial SDP (7) with the objective value $v = \sum_{i\in M} t_i^{p_i}$. Then, it follows from $t_i \geq f_i(\boldsymbol{x}_{C_i})^2$ $(i \in M)$ that

$$v = \sum_{i\in M} t_i^{p_i} \geq \sum_{i\in M} f_i(\boldsymbol{x}_{C_i})^{2p_i}.$$

Therefore, we have shown the equivalence of (3) and the polynomial SDP (7). ∎

Using the relation (6) in the same way, we obtain some other polynomial SDP formulations:

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{m} t_j \\ \text{subject to} & \begin{pmatrix} 1 & f_i(\boldsymbol{x}_{C_i})^{p_i} \\ f_i(\boldsymbol{x}_{C_i})^{p_i} & t_i \end{pmatrix} \succeq \boldsymbol{O} \quad (i \in M), \end{array}\right\} \tag{8}$$

$$\left.\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{pmatrix} 1 & 0 & \cdots & 0 & f_1(\boldsymbol{x}_{C_1})^{p_1} \\ 0 & 1 & \cdots & 0 & f_1(\boldsymbol{x}_{C_2})^{p_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & f_m(\boldsymbol{x}_{C_m})^{p_m} \\ f_1(\boldsymbol{x}_{C_1})^{p_1} & f_1(\boldsymbol{x}_{C_2})^{p_2} & \cdots & f_m(\boldsymbol{x}_{C_m})^{p_m} & t \end{pmatrix} \succeq \boldsymbol{O}. \end{array}\right\} \tag{9}$$

As variations of (7), (8) and (9), we also obtain the polynomial SDPs:

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{m} t_j^{2p_j} \\ \text{subject to} & \begin{pmatrix} t_i & f_i(\boldsymbol{x}_{C_i}) \\ f_i(\boldsymbol{x}_{C_i}) & t_i \end{pmatrix} \succeq \boldsymbol{O},\ t_i \geq 0\ (i \in M), \end{array}\right\} \tag{10}$$

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{m} t_j^{2} \\ \text{subject to} & \begin{pmatrix} t_i & f_i(\boldsymbol{x}_{C_i})^{p_i} \\ f_i(\boldsymbol{x}_{C_i})^{p_i} & t_i \end{pmatrix} \succeq \boldsymbol{O},\ t_i \geq 0\ (i \in M), \end{array}\right\} \tag{11}$$

$$\left.\begin{array}{ll} \text{minimize} & t^2 \\ \text{subject to} & \begin{pmatrix} t & 0 & \cdots & 0 & f_1(\boldsymbol{x}_{C_1})^{p_1} \\ 0 & t & \cdots & 0 & f_1(\boldsymbol{x}_{C_2})^{p_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & t & f_m(\boldsymbol{x}_{C_m})^{p_m} \\ f_1(\boldsymbol{x}_{C_1})^{p_1} & f_1(\boldsymbol{x}_{C_2})^{p_2} & \cdots & f_m(\boldsymbol{x}_{C_m})^{p_m} & t \end{pmatrix} \succeq \boldsymbol{O}, \\ & t \geq 0. \end{array}\right\} \tag{12}$$

Intuitively the formulating the problem (3) as (10), (11) and (12) does not seem to have advantages in comparison with (7), (8) and (9), respectively, because the degree of the

objective function is doubled and more auxiliary variables t_i ($i \in M$) and t are contained in the diagonal of polynomial matrix inequality constraints. In Section 4, we show that the size of the SDP relaxation of (10) is the same as the size of the SDP relaxation of (7), but the number of nonzeros in the coefficient matrix is slightly larger and the accuracy attained is worse than the one by the relaxation problem of (7) through numerical results.

We can rewrite the polynomial SDPs (10), (11) and (12) as the following polynomial second order cone programs (SOCPs):

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{m} t_j^{2p_j} \quad \text{subject to} \quad (t_i, f_i(\boldsymbol{x}_{C_i})) \in \mathcal{K}^2 \quad (i \in M). \end{array}\right\} \tag{13}$$

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{m} t_j^{2} \quad \text{subject to} \quad (t_i, f_i(\boldsymbol{x}_{C_i})^{p_i}) \in \mathcal{K}^2 \quad (i \in M). \end{array}\right\} \tag{14}$$

$$\text{minimize} \quad t^2 \quad \text{subject to} \quad (t, f_1(\boldsymbol{x}_{C_1})^{p_1}, \ldots, f_m(\boldsymbol{x}_{C_m})^{p_m}) \in \mathcal{K}^{1+m}. \tag{15}$$

Here $\mathcal{K}^2$ and $\mathcal{K}^{1+m}$ denote 2- and $(m+1)$-diensional SOCP cones. We may replace the objective function t^2 of the last SOCP (15) by t.

$$\text{minimize} \quad t \quad \text{subject to} \quad (t, f_1(\boldsymbol{x}_{C_1})^{p_1}, \ldots, f_m(\boldsymbol{x}_{C_m})^{p_m}) \in \mathcal{K}^{1+m}. \tag{16}$$

When all polynomials $f_i(\boldsymbol{x}_{C_i})$ ($i \in M$) are linear and $p_i = 1$ ($i \in M$), the problem (16) is, in fact, a linear SOCP that can be directly solved by a primal-dual interior-point method without using any relaxation technique. In such a case, solving (16) is more efficient than solving all the other formulations (7) – (15). Also for some special cases of polynomial least squares problems with all $f_i(\boldsymbol{x}_{C_i})$ ($i \in M$) linear and each $p_i = 2^{q_i}$ for some $q_i = 0, 1, \ldots$, they can be transformed to linear SOCPs. See [13] for more details.

In general cases where some of $f_i(\boldsymbol{x}_{C_i})$s are nonlinear polynomials, (13), (14) and (15) become polynomial (but not linear) SOCPs. The sparse SDP relaxation method proposed by Kojima *et al.* [16, 17] can be applied to such SOCPs. A basis of the Euclidean space where the underlying second-order cone lies is chosen in their method, and different choices of basis induce different SDP relaxation problems. When the standard Euclidean basis consisting of the unit coordinate vectors is chosen, the SDP relaxation problems induced from the SOCPs (13), (14) and (15) can be shown to be the same as those induced from (10), (11) and (12), respectively, by applying the SDP relaxation method [15] described in Section 3. Therefore, we will not consider the polynomial SOCP formulations (13), (14) and (15) in the subsequent discussion, and we focus on the polynomial SDP formulations (7) – (12). We show in Section 4 that the polynomial SDP formulation (7) is more efficient than all others.

3 A sparse SDP relaxation of the polynomial SDP

We briefly describe the sparse SDP relaxation [15, 31] of the sparse POP formulation (3) and all polynomial SDP formulations (7) – (12) of the polynomial least squares problem (3). Consider (5) to deal with them simultaneously. For example, (5) represents (3) if $n' = n$, $m' = m$, $N' = N$, $K = K_o = M$, $C'_i = C_i$ ($i \in K$), $g_j(\boldsymbol{y}_{C'_j}) = f_j(\boldsymbol{x}_{C_j})^{p_j}$ ($j \in K_o$) and $K_c = \emptyset$,

and (7) if $n' = n + m$, $m' = m$, $N' = \{1, \ldots, n'\}$, $K = K_o = K_c = M$, $C'_i = C_i \cup \{n + i\}$ $(i \in K)$, $g_j(\boldsymbol{y}_{C'_j}) = y^{p_j}_{n+j}$ $(j \in K_o)$ and

$$\boldsymbol{F}_i(\boldsymbol{y}_{C'_i}) = \begin{pmatrix} 1 & f_i(\boldsymbol{x}_{C_i}) \\ f_i(\boldsymbol{x}_{C_i}) & t_i \end{pmatrix} \ (i \in K_c).$$

The sparsity of polynomials in (5) is first considered with a graph $G(N', E)$ representing the sparsity structure of (5). More specifically, a graph $G(N', E)$ is constructed such that a pair $\{k,\ \ell\}$ with $k \neq \ \ell$ selected from the node set N' is an edge or $\{k,\ \ell\} \in E$ if and only if $k \in C'_i,\ \ell \in C'_i$ for some $i \in K$. We call the graph $G(N', E)$ *a correlative sparsity pattern (csp) graph*. Each C'_i is a clique of $G(N', E)$ $(i \in K)$. The next step is to generate a chordal extension $G(N', E')$ of $G(N', E)$. (For the definition and basic properties of chordal graphs, we refer to [1]). For simplicity of notation, we assume that $C'_1, \ldots, C'_m$ form the set of maximal cliques of a chordal extension $G(N', E')$ of the underlying csp graph $G(N', E)$ of the polynomial SDP (5); if this is not the case, we replace C'_i by a maximal clique containing C'_i. For more details, see [31].

For every $C \subset N'$ and $\psi \in \mathbb{Z}_+$, we define

$$\mathcal{A}^C_\psi \ = \ \left\{ \boldsymbol{\alpha} \in \mathbb{Z}^n_+ : \alpha_j = 0 \text{ if } j \notin C,\ \sum_{i \in C} \alpha_i \leq \psi \right\}.$$

Depending on how a column vector of the monomials $\boldsymbol{y}^{\boldsymbol{\alpha}}$ is chosen, the sparse relaxation [31] or the dense relaxation [19] is derived. The dense relaxation is obtained using a column vector $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{N'}_\psi)$ that contains all the possible monomials $\boldsymbol{y}^{\boldsymbol{\alpha}}$ of degree up to ψ. Selecting a column vector $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^C_\psi)$ of the monomials $\boldsymbol{y}^{\boldsymbol{\alpha}}$ $(\boldsymbol{\alpha} \in \mathcal{A}^C_\psi)$ where elements $\boldsymbol{y}^{\boldsymbol{\alpha}}$ $(\boldsymbol{\alpha} \in \mathcal{A}^C_\psi)$ are arranged in lexicographically increasing order of $\boldsymbol{\alpha}$'s leads to the sparse SDP relaxation if we take $C \subset N'$ with a small cardinality or the dense SDP relaxation if we take $C = N'$. The first element of the column vector $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^C_\psi)$ is always $\boldsymbol{y}^{\boldsymbol{0}} = 1$ since $\boldsymbol{0} \in \mathcal{A}^C_\psi$. The size of $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{N'}_\psi)$ of the dense relaxation is $\begin{pmatrix} n' + \psi \\ \psi \end{pmatrix}$, and the size of $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^C_\psi)$ of the sparse relaxation is $\begin{pmatrix} \#C + \psi \\ \psi \end{pmatrix}$. As a result, the size of $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{N'}_\psi)$ of the dense relaxation is always larger than that of $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^C_\psi)$ of the sparse relaxation unless $C = N'$.

Let $\omega_0 = \lceil \deg(\sum_{j \in M} g_j(\boldsymbol{y}_{C'_j}))/2 \rceil$, $\omega_i = \lceil \deg(\boldsymbol{F}_i(\boldsymbol{y}_{C'_i}))/2 \rceil$ for every $i \in K_c$, and

$$\omega_{\max} = \max\{\omega_i : i \in \{0\} \cup K_c\}. \tag{17}$$

Then the polynomial SDP (5) is transformed into an equivalent polynomial SDP

$$\left.\begin{array}{ll} \text{minimize} & \sum_{j \in K_o} g_j(\boldsymbol{y}_{C'_j}) \\ \text{subject to} & \boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_i}_{\omega-\omega_i}) \boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_i}_{\omega-\omega_i})^T \otimes \boldsymbol{F}_i(\boldsymbol{y}_{C'_i}) \succeq \boldsymbol{O} \ (i \in K_c), \\ & \boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_j}_{\omega}) \boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_j}_{\omega})^T \succeq \boldsymbol{O} \ (j \in K) \end{array}\right\} \tag{18}$$

with some *relaxation order* $\omega \geq \omega_{\max}$, where $\otimes$ denotes the Kronecker product of the two matrices $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_i}_{\omega-\omega_i}) \boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_i}_{\omega-\omega_i})^T$ and $\boldsymbol{F}_i(\boldsymbol{y}_{C'_i})$.

The matrices $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_i}_{\omega-\omega_i})\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_i}_{\omega-\omega_i})^T$ $(i \in K_c)$ and $\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_j}_{\omega})\boldsymbol{u}(\boldsymbol{y}, \mathcal{A}^{C'_j}_{\omega})^T$ $(j \in K)$ are positive semidefinite symmetric matrices of rank one for any $\boldsymbol{y}$, and the element in the upper-left corner of the matrices is 1. The equivalence between the polynomial SDP (5) and the polynomial SDP (18) is therefore shown.

Since the objective function of the polynomial SDP (18) is a real-valued polynomial and the left hand side of the matrix inequality constraints of the polynomial SDP (18) are real symmetric polynomial matrices, we can rewrite the polynomial SDP (18) as

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \tilde{c}_0(\boldsymbol{\alpha})\boldsymbol{y}^{\boldsymbol{\alpha}} \\ \text{subject to} & \boldsymbol{L}_i(\boldsymbol{0},\omega) - \displaystyle\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{L}_i(\boldsymbol{\alpha},\omega)\boldsymbol{y}^{\boldsymbol{\alpha}} \succeq \boldsymbol{O}\ (i \in K_c), \\ & \boldsymbol{M}_j(\boldsymbol{0},\omega) - \displaystyle\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{M}_j(\boldsymbol{\alpha},\omega)\boldsymbol{y}^{\boldsymbol{\alpha}} \succeq \boldsymbol{O}\ (j \in K). \end{array}\right\}$$

for some $\widetilde{\mathcal{F}} \subset \mathbb{Z}^n_+\backslash\{\boldsymbol{0}\}$, $\tilde{c}_0(\boldsymbol{\alpha}) \in \mathbb{R}$ $(\boldsymbol{\alpha} \in \widetilde{\mathcal{F}})$ and real symmetric matrices $\boldsymbol{L}_i(\boldsymbol{\alpha},\omega)$, $\boldsymbol{M}_j(\boldsymbol{\alpha},\omega)$ $(\boldsymbol{\alpha} \in \widetilde{\mathcal{F}} \cup \{\boldsymbol{0}\},\ i \in K_c,\ j \in K)$. Note that the size of the matrices $\boldsymbol{L}_i(\boldsymbol{\alpha},\omega)$, $\boldsymbol{M}_j(\boldsymbol{\alpha},\omega)$ $(\boldsymbol{\alpha} \in \widetilde{\mathcal{F}} \cup \{\boldsymbol{0}\},\ i \in K_c,\ j \in K)$ and the number of monomials $\boldsymbol{y}_{\boldsymbol{\alpha}}$ $(\boldsymbol{\alpha} \in \widetilde{\mathcal{F}})$ are determined by the relaxation order ω. Each monomial $\boldsymbol{y}^{\boldsymbol{\alpha}}$ is replaced by a single real variable $z_{\boldsymbol{\alpha}}$, and we have an SDP relaxation problem of the polynomial SDP (5), called *a sparse SDP relaxation*:

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \tilde{c}_0(\boldsymbol{\alpha})z_{\boldsymbol{\alpha}} \\ \text{subject to} & \boldsymbol{L}_i(\boldsymbol{0},\omega) - \displaystyle\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{L}_i(\boldsymbol{\alpha},\omega)z_{\boldsymbol{\alpha}} \succeq \boldsymbol{O}\ (i \in K_c), \\ & \boldsymbol{M}_j(\boldsymbol{0},\omega) - \displaystyle\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{M}_j(\boldsymbol{\alpha},\omega)z_{\boldsymbol{\alpha}} \succeq \boldsymbol{O}\ (j \in K). \end{array}\right\} \tag{19}$$

Here $\boldsymbol{y}^{\boldsymbol{0}} = z_{\boldsymbol{0}} = 1$. We mention that the dense SDP relaxation is obtained if we take $C'_i = N'$ $(i \in K_c)$ and $C'_j = N'$ $(j \in K)$ in (18).

We call each $\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{L}_i(\boldsymbol{\alpha},\omega)z_{\boldsymbol{\alpha}}$ *a localizing matrix* , and each $\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{M}_j(\boldsymbol{\alpha},\omega)z_{\boldsymbol{\alpha}}$ *a moment matrix* in (19). If $\boldsymbol{F}_i(\boldsymbol{y}_{C'_i})$ is $r_i \times r_i$, then the size (= the number of rows = the number of columns) of the localizing matrix $\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{L}_i(\boldsymbol{\alpha},\omega)z_{\boldsymbol{\alpha}}$ is

$$\left(\begin{array}{c} \#C'_i + \omega - \omega_i \\ \omega - \omega_i \end{array}\right) r_i$$

$(i \in K_c)$. Similarly, the size of the moment matrix $\sum_{\boldsymbol{\alpha}\in\widetilde{\mathcal{F}}} \boldsymbol{M}_j(\boldsymbol{\alpha},\omega)z_{\boldsymbol{\alpha}}$ is

$$\left(\begin{array}{c} \#C'_j + \omega \\ \omega \end{array}\right)$$

$(j \in K)$. Since the sizes of the localizing and moment matrices affect very much computational performance, their sizes of the various formulations in Section 2 are compared in Section 4.

The SDP relaxation problem (19) is solved by SeDuMi in our numerical experiment whose results are reported in Section 5. The problem is formulated as the dual standard form

$$\text{maximize } \boldsymbol{b}^T\boldsymbol{y} \quad \text{subject to} \quad \boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} \succeq \boldsymbol{0}. \tag{20}$$

Here each column index of $\boldsymbol{A}^T$ (hence each row index of $\boldsymbol{A}$) corresponds to an $\boldsymbol{\alpha} \in \widetilde{\mathcal{F}}$, $\boldsymbol{y}$ the column vector of $z_{\boldsymbol{\alpha}}$ ($\boldsymbol{\alpha} \in \widetilde{\mathcal{F}}$) and $\boldsymbol{b}$ the column vector of $-\tilde{c}_0(\boldsymbol{\alpha})$ ($\boldsymbol{\alpha} \in \widetilde{\mathcal{F}}$). Note that the coefficient matrices $\boldsymbol{L}(\boldsymbol{\alpha}, \omega)$, $\boldsymbol{M}(\boldsymbol{\alpha}, \omega)$ ($\boldsymbol{\alpha} \in \widetilde{\mathcal{F}} \cup \{\boldsymbol{0}\}$), which is called SDP blocks in the numerical results in Section 5, are reshaped into column vectors and arranged in $\boldsymbol{c}$ and $\boldsymbol{A}^T$. Computational performance of solving (20) with SeDuMi depends on the size of the coefficient matrix $\boldsymbol{A}$, the sparsity of the coefficient matrix $\boldsymbol{A}$ and the size of SDP blocks. The most time-consuming part in primal-dual interior-point methods is solving the Schur complement matrix that is constructed from $\boldsymbol{A}$. We note that the size of the Schur complement matrix coincides with the number of rows of $\boldsymbol{A}$ and that its sparsity is determined by the sparsity of $\boldsymbol{A}$. For details on the relationship between the Schur complement matrix and $\boldsymbol{A}$, we refer to [12]. Whether formulating polynomial SDPs with a small number of large-sized SDP constraints is a better approach than formulating polynomial SDPs with a large number of small-sized SDP constraints should be decided based on the size of the coefficient matrix $\boldsymbol{A}$, the sparsity of the coefficient matrix $\boldsymbol{A}$ and the size of SDP blocks.

4 Comparison of various formulations

There exist several advantages of formulating the problems (3) as polynomial SDPs. We compare the maximum degree of polynomials, the minimum relaxation oder defined by (17), the ability to exploit the sparsity, the size of the moment matrices, and the size of the localizing matrices of the various formulation presented in Secion 2.

As seen in Section 3, the maximum of the degree of the objective function and the degrees of polynomial SDP constraints, determine the minimum relaxation order which is denoted as $\omega_{\max}$ in (17). We usually choose the value $\omega_{\max}$ for the relaxation order ω when an SDP relaxation problem (19) of the given polynomial SDP (or POP) (5) is constructed. The chosen value $\omega_{\max}$ may not be large enough to get an accurate optimal solution in some cases. If a solution of desired accuracy is not obtained after the application of SparsePOP, then ω is increasesd by 1 and solve the SDP relaxation problem with the updated relaxation order ω again. This does not guarantee attaining an optimal solution in theory, but a solution of better accuracy is usually obtained in practice. In view of computational efficiency, however, taking a smaller value for the relaxation order ω works more efficiently than a large value because the size of the SDP relaxation problem grows very rapidly as we take a increasingly large value for the relaxation order ω. It is thus important to have a smaller minimum relaxation order $\omega_{\max}$ that leads to a smaller size of the starting SDP relaxation problem. In Table 1, the maximum degree of polynomials and the minimum relaxation order for the formulations (3) and (7) – (12) are summarized. The following notation is used.

$$\begin{aligned} \bar{\delta} &= \max\{p_i \deg(f_i(\boldsymbol{x}_{C_i})) \ \ (i \in M)\}, \\ \hat{\delta} &= \max\{\deg(f_i(\boldsymbol{x}_{C_i})) \ (i \in M), \ p_i \ (i \in M)\}. \end{aligned}$$

formulation	max. degree	the min. relaxation order $\omega_{\max}$ in (17)
(3)	$2\bar{\delta}$	$\omega^{(3)}_{\max} = \bar{\delta}$
(7) & (10)	$\hat{\delta}$	$\omega^{(7)}_{\max} = \lceil \hat{\delta}/2 \rceil$
(8) & (11)	$\bar{\delta}$	$\omega^{(8)}_{\max} = \lceil \bar{\delta}/2 \rceil$
(9) & (12)	$\bar{\delta}$	$\omega^{(9)}_{\max} = \lceil \bar{\delta}/2 \rceil$

Table 1: Comparison of the maximum degree of polynomials and the relaxation order of the various formulations.

In Table 1, the sparse POP formulation (3) has the largest maximum degree of polynomials among the formulations, and the sparse polynomial SDP formulations (7) and (10) have the smallest maximum degree. In particular, the maximum degree $2\bar{\delta}$ in (3) is at least twice larger than the other formulations. Since the smallest relaxation order that can be taken is roughly the half of the maximum degree of polynomials, we see that the minimum relaxation order for the sparse polynomials SDP formulations (7) and (10) is the smallest. This is the main advantage of (7) and (10) in comparison with (3).

Table 2 shows how the relaxation order ω, the degree of polynomials $f_i(\boldsymbol{x}_{C_i})$, p_i $(i \in M)$ and the size of maximal cliques C_i $(i \in M)$ determine the maximum size of moment matrices and the size of localizing matrices. We use the following notation:

$$\begin{aligned} \gamma_{\max} &= \max\{\#\ C_j\ (j \in K)\}, \\ \hat{\eta}_i &= \lceil \deg(f_i(\boldsymbol{x}_{C_i}))/2 \rceil\ (i \in M), \\ \bar{\eta}_i &= \lceil p_i \deg(f_i(\boldsymbol{x}_{C_i}))/2 \rceil\ (i \in M), \\ \bar{\eta} &= \lceil \bar{\delta}/2 \rceil = \max\{\bar{\eta}_i\ (i \in M)\}. \end{aligned}$$

In addition, $\omega^{(3)}, \omega^{(7)}, \omega^{(8)}, \omega^{(9)}$ indicate the relaxation order used for (3), (7) & (10), (8) & (11) and (9) & (12), respectively.

formulation	exploiting sparsity	the max. size of moment matrices	the size of localizing matrices
(3)	◯	$\begin{pmatrix} \gamma_{\max} + \omega^{(3)} \\ \omega^{(3)} \end{pmatrix}$	N/A
(7) & (10)	◯	$\begin{pmatrix} \gamma_{\max} + 1 + \omega^{(7)} \\ \omega^{(7)} \end{pmatrix}$	$\begin{pmatrix} \#C_i + 1 + \omega^{(7)} - \hat{\eta}_i \\ \omega^{(7)} - \hat{\eta}_i \end{pmatrix} \times 2$
(8) & (11)	◯	$\begin{pmatrix} \gamma_{\max} + 1 + \omega^{(8)} \\ \omega^{(8)} \end{pmatrix}$	$\begin{pmatrix} \#C_i + 1 + \omega^{(8)} - \bar{\eta}_i \\ \omega^{(8)} - \bar{\eta}_i \end{pmatrix} \times 2$
(9) & (12)	×	$\begin{pmatrix} n + 1 + \omega^{(9)} \\ \omega^{(9)} \end{pmatrix}$	$\begin{pmatrix} n + 1 + \omega^{(9)} - \bar{\eta} \\ \omega^{(9)} - \bar{\eta} \end{pmatrix} \times (m+1)$

Table 2: Comparison of various formulations. N/A: not applicable.

Recall that the relaxation order $\omega^{(3)}, \omega^{(7)}, \omega^{(8)}, \omega^{(9)}$ must satisfy

$$\omega^{(k)} \geq \omega^{(k)}_{\max}\ (k = 3, 7, 8, 9),$$

and that

$$\omega^{(7)}_{\max} \leq \omega^{(8)}_{\max} = \omega^{(9)}_{\max} < \omega^{(3)}_{\max}.$$

Hence, if we take $\omega^{(k)} = \omega^{(k)}_{\max}$ $(k = 3, 7, 8, 9)$ for the starting SDP relaxation for the formulations, the largest size of moment matrices of (7) and (10) is the smallest among the largest size of moment matrices produced from the formulations, and the largest size of moment matrices of (3) is the largest although (3) does not involve any localizing matrices. We confirm again that the sparse SDP formulations (7) and (10) have an clear advantage over the sparse POP formulation (3) and the other sparse SDP formulations (8) & (11).

Let us now compare (7) & (10) and (8) & (11) further. When $p_i = 1$ $(i \in M)$, there is no difference in these two pairs of formulations; (7) $\equiv$ (8) and (10) $\equiv$ (11). Suppose that $p_i = 2$ $(i \in M)$. Then, $2\hat{\delta} = \bar{\delta}$. It follows that $2\omega^{(7)}_{\max} - 1 \leq \omega^{(8)}_{\max}$. Consequently, the size of the starting SDP relaxation in the sparse polynomial SDP formulations (7) and (10) is smaller than that in the sparse polynomial SDP formulations (8) and (11).

The sparsity of polynomials in the formulations (9) and (12) can not be exploited, thus, the maximum size of moment matrix and the size of the localizing matrices are expected to become larger than (7), (10), (8) and (11) unless $\gamma_{\max} = n$.

The pairs of polynomial SDP formulations (7) & (10), (8) & (11), (9) & (12) are equivalent in the maximum degree, the maximum size of moment matrices, and the size of localizing matrices as indicated in Table 2. Their computational accuracy is, however, different. In fact, (7), (8), and (9) provide higher accuracy than their counterpart. As an example, the comparison of numerical accuracy for the Broyden tridiagonal function between (7) and (10) is shown in Table 3. We see that (7) results in smaller relative errors. Notice that the size of $\boldsymbol{A}$ for (7) is equivalent to that for (10). See Table 4 for the notation used in Table 3.

Polynomial SDP formulation (7)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
100	2	4158×26877	46269	12(8.9)	4.6e-10	19.2
150	2	6258×40427	69619	12(9.0)	1.0e-10	23.3
200	2	8358×53977	92969	12(9.0)	2.4e-9	34.2
Polynomial SDP formulation (10)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
100	2	4158×26877	48751	12(8.9)	1.4e-8	16.8
150	2	6258×40427	73351	12(9.0)	2.1e-8	23.3
200	2	8358×53977	97951	12(9.0)	2.0e-8	31.8

Table 3: Numerical results of the Broyden tridiagonal function. $x_1 \geq 0$ is added.

As observed with the size of the moment and localizing matrices in Table 2, computational accuracy in Table 3, the relaxation order in Table 1, we use (7) to compare with (3) numerically in Section 5.

5 Numerical results

We compare numerical results of the spare POP formulation (3) and the sparse polynomial SDP formulation (7) (PSDP) of several polynomial least squares problems from [3, 5, 9, 18, 23, 24, 29]. Problems for the numerical tests are randomly generated problems, the Broyden tridiagonal function, the generalized Rosenbrock function, the chained Wood function, the Broyden banded function, the Watson function, the partition problem described in [9], and polynomial least squares problems using the cyclic-n polynomial and the economic-n polynomial from [29]. All the problems were solved by Matlab codes using SparsePOP [30] and SeDuMi [27] on the hardware Power Mac G5 of 2.5 GHz with 2GB memory. The notation in Table 4 is used for the description of numerical experiments.

n	the number of variables
sizeA	the size of the coefficient matrix $\boldsymbol{A}$ of the SDP relaxation problem in the SeDuMi input format (20)
ω	the relaxation order
#nz	the number of nonzeros in the coefficient matrix $\boldsymbol{A}$
sdpBl	the maximum size (average size) of SDP blocks in the coefficient matrix $\boldsymbol{A}$
rel.err	the relative error of SDP and POP/PSDP objective values
cpu	the cpu time to solve SDP by SeDuMi in seconds

Table 4: Notation

A smaller value of the starting relaxation order $\omega = \omega_{\max}$ given by (17) for the sparse PSDP formulation (7) than the sparse POP formulation (3), as shown in Section 4, does not always mean better performance of (7). Also the relaxation order $\omega = \omega_{\max}$ may not be large enough to get optimal solutions with high accuracy. In such a case, increasing the relaxation order, which gives an impact on numerical performance, and solving the problem again is necessary. Note that no theoretical result on the speed of the convergence is known, although the convergence of the SDP relaxation of increasing size to the optimal value of the POP was proved by [19].

We show the effects of the size of the coefficient matrix $\boldsymbol{A}$ of the SDP relaxation problem in the SeDuMi input format (20) (sizeA), the number of nonzero elements of $\boldsymbol{A}$ (#nz), and the size of SDP blocks of $\boldsymbol{A}$ (sdpBl) on numerical performance. In the numerical experiments comparing the sparse POP formulation (3) and the sparse polynomial SDP formulation (7), we observe that the formulation that leads to larger sizeA, #nz and sdpBl takes longer cpu time to find an optimal solution except the generalized Rosenbrock function. Among the three factors, sizeA and #nz affect the computational efficiency more than sdpBl as will be seen in Table 12. It should be mentioned that the three factors may not completely determine computational efficiency particularly when cpu time is very small, for instance, less than 5 seconds, for small-sized problems. SeDuMi usually takes a fixed amount of cpu time regardless of the size of SDP, and finding an approximate solution of lower accuracy may take shorter than obtaining an approximate solutions of higher accuracy. This will be observed in some of the numerical tests on the generalized Rosenbrock function and a few tests on the partition problem using transformation.

We begin with randomly generated unconstrained POPs with artificial correlative sparsity that show a clear advantage of the sparse PSDP formulation (7) against the saprse POP formulation (3). As described in Section 3, the correlative sparsity affects sizeA, #nzA and sdpBl. With a given clique size $2 \leq c \leq n$, define cliques

$$C_i = \{j \in N : i \leq j \leq i + c - 1\} \ (i = 1, 2, \ldots, n - c + 1),$$

where $N = \{1, 2, \ldots, n\}$. We then generate a vector $\boldsymbol{g}_i$ $(i = 1, 2, \ldots, n-c+1)$ using random number generator in the interval (-1, 1) for coefficients. Let

$$f_i(\boldsymbol{x}) = \boldsymbol{g}_i^T \boldsymbol{u}(\boldsymbol{x}, \mathcal{A}_{d_i}^{C_i}) \ (i = 1, \ldots, n - c + 1),$$

where d_i denotes the degree of $f_i(\boldsymbol{x})$. Then, we consider

$$\text{minimize} \ \sum_{i=1}^{n-c+1} f_i(\boldsymbol{x}_{C_i})^{2p_i} + \sum_{i=1}^{n} x_i^2. \tag{21}$$

where $\sum_{i=1}^{n} x_i^2$ is added in order to avoid multiple number of optimal solutions.

Tables 5 and 6 show numerical results for varying n, the size c of cliques, p_i and $d_i = \deg f_i(\boldsymbol{x}_{C_i})$ $(i = 1, 2, \ldots, n - c + 1)$. The notation "deg" denotes the degree of the polynomial objective function of (21). For all tested cases, sizeA, #nzA, and sdpBl of the sparse PSDP formulation (7) are smaller than those of the sparse POP formulation (3), providing optimal solutions faster. See Table 1 for differences in $\omega = \omega_{\max}$. In Table 6, we took $d_i = 2$ and $p_i = 2$ $(i = 1, 2, \ldots, n - c + 1)$, so that the degree of the polynomial objective function of (21) is $2 \times 2 \times 2 = 8$. In (3), we need to take the relaxation order ω not less than $\omega_{\max} = 4$, while we can take the starting relaxation order $\omega_{\max} = 1$ in (7). Actually, $\omega = 4$ is used for (3) while $\omega = 1$ is used for (7). This provideds big differences in sizeA, #nzA, and sdpBl. As a result, cpu time for (7) is much smaller than that of (3).

The Broyden tridiagonal function [23] is

$$\begin{aligned} f(\boldsymbol{x}) \ &= \ ((3 - 2x_1)x_1 - 2x_2 + 1)^2 + \sum_{i=2}^{n-1} ((3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1)^2 \\ &\quad + ((3 - 2x_n)x_n - x_{n-1} + 1)^2 . \end{aligned}$$

The numerical results of the Broyden tridiagonal function are shown in Table 7. The sparse PSDP formulation (7) requires the relaxation order 2 to get accurate optimal solutions. The sizeA, #nzA and sdpBl for the sparse PSDP formulation (7) with $\omega = 2$ are larger than those for the sparse POP formulation (3), taking longer to get an optimal solution. An inequality constraint $x_1 \geq 0$ is added to avoid numerical difficulty arising from multiple number of solutions.

The generalized Rosenbrock function [24] is written as

$$f(\boldsymbol{x}) = 1 + \sum_{i=2}^{n} \left\{ 100 \left(x_i - x_{i-1}^2 \right)^2 + (1 - x_i)^2 \right\}.$$

In Table 8, we notice that sizeA, #nzA, sdpBl of (7) are smaller than those of (3). Although (7) took longer cpu time, the accuracy shown in the column of rel.err is better than (3).

The sparse POP formulation (3)										
p_i	d_i	deg	c	n	$\omega = \omega_{\max}$	sizeA	#nzA	sdpBl	rel.err	cpu
1	2	4	3	30	2	574×4848	5270	10(5.9)	5.1e-9	2.4
1	2	4	3	50	2	974×8248	8950	10(5.9)	4.0e-9	3.1
1	2	4	3	100	2	1974×16748	18150	10(6.0)	9.2e-9	5.5
1	2	4	3	200	2	3974×33748	36550	10(6.0)	9.2e-9	8.1
1	3	6	5	50	3	6005×91985	103138	35(21.4)	3.2e-8	35.2
1	3	6	5	100	3	12305×188235	210538	35(21.5)	4.7e-9	73.7
1	3	6	5	200	3	49601×889858	977662	56(32.5)	9.3e-9	764.6
The sparse PSDP formulation (7)										
p_i	d_i	deg	c	n	$\omega = \omega_{\max}$	sizeA	#nzA	sdpBl	rel.err	cpu
1	2	4	3	30	1	175×1150	1482	4(2.5)	5.5e-9	0.4
1	2	4	3	50	1	295×1950	2522	4(2.5)	7.5e-9	0.7
1	2	4	3	100	1	595×3950	5122	4(2.5)	4.4e-5	1.5
1	2	4	3	200	1	1195×7950	10322	4(2.5)	2.1e-5	2.8
1	3	6	5	50	2	2011×13042	18395	6(4.1)	2.7e-8	5.8
1	3	6	5	100	2	4111×26592	37595	6(4.1)	1.2e-8	13.1
1	3	6	5	200	2	8311×53692	75995	6(4.1)	3.6e-8	20.6

Table 5: Numerical experiments with the randomly generated problem (21) of degree 4 and 6.

The sparse POP formulation (3)										
p_i	d_i	deg	c	n	$\omega = \omega_{\max}$	sizeA	#nzA	sdpBl	rel.err	cpu
2	2	8	3	30	4	3404×65048	76990	35(24.8)	3.3e-8	26.1
2	2	8	3	50	4	5804×110308	130210	35(24.9)	1.3e-7	45.7
2	2	8	3	100	4	11804×223458	263260	35(24.9)	1.2e-7	92.6
The sparse PSDP formulation (7)										
p_i	d_i	deg	c	n	$\omega = \omega_{\max}$	sizeA	#nzA	sdpBl	rel.err	cpu
2	2	8	3	30	1	347×1896	2228	5(3.0)	1.5e-8	0.6
2	2	8	3	50	1	587×3216	3788	5(3.0)	1.2e-8	1.2
2	2	8	3	100	1	1187×6516	7688	5(3.0)	1.6e-8	2.2

Table 6: Numerical experiments with the randomly generated problem (21) of degree 8.

The sparse POP formulation (3)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
200	2	3974×19819	19621	10(10.0)	8.9e-8	8.4
500	2	9974×49819	49321	10(10.0)	1.5e-6	11.7
1000	2	19974×99819	98821	10(10.0)	1.5e-6	22.5
The sparse PSDP formulation (7)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
200	1	997×4188	4984	4(3.0)	1.0e+0	0.8
500	1	2497×10488	12484	4(3.0)	1.0e+0	3.1
1000	1	4997×20988	24984	4(3.0)	1.0e+0	5.9
200	2	8358×53977	92969	12(9.0)	2.4e-9	34.2
500	2	20958×135277	233069	12(9.0)	3.7e-7	67.4
1000	2	41958×270777	466569	12(9.0)	2.4e-7	165.2

Table 7: Numerical results of the Broyden tridiagonal function.

The sparse POP formulation (3)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
200	2	1988×7156	6957	6(6.0)	5.1e-5	1.9
500	2	4988×17956	17457	6(6.0)	1.6e-4	4.1
1000	2	9988×35956	34957	6(6.0)	2.1e-4	8.0
The sparse PSDP formulation (7)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
200	1	995×4570	4175	3(2.2)	5.3e-5	2.1
500	1	2495×11470	10475	3(2.2)	5.3e-7	4.8
1000	1	4995×22970	20975	3(2.2)	1.1e-6	9.9

Table 8: Numerical results of the generalized Rosenbrock function.

The difference in cpu time, however, is small. An inequality constraint $x_1 \geq 0$ is added as for the Broyden tridiagonal function.

The chained Wood function [3] is

$$\begin{aligned} f(\boldsymbol{x}) &= 1 + \sum_{i \in J} \big(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 + 90(x_{i+3} - x_{i+2}^2)^2 + (1 - x_{i+2})^2 \\ &\quad + 10(x_{i+1} + x_{i+3} - 2)^2 + 0.1(x_{i+1} - x_{i+3})^2\big), \end{aligned}$$

where $J = \{1, 3, 5, \ldots, n-3\}$ and n is a multiple of 4. In Table 9, the sparse PSDP formulation (7) takes longer to converge, and results in less accurate solutions for the tested n's except $n = 1000$. We notice that sizeA, #nzA, sdpBl are larger in (7) than those of (3).

The sparse POP formulation (3)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
100	2	449×1241	1142	4(3.5)	8.1e-6	1.3
200	2	899×2491	2292	4(3.5)	5.3e-6	0.8
400	2	1799×4991	4592	4(3.5)	1.2e-5	1.4
1000	2	4499×12491	11492	4(3.5)	3.4e-5	3.8
The sparse PSDP formulation (7)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
100	1	248×2891	1470	7(5.0)	6.5e-5	0.8
200	1	498×5841	2970	7(5.0)	1.8e-4	1.2
400	1	998×11741	5970	7(5.0)	3.9e-4	2.2
1000	1	4494×22954	21956	7(5.0)	1.8e-6	10.2

Table 9: Numerical results of the chained Wood function

The Broyden banded function [23] is written as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \left(x_i(2 + 5x_i^2) + 1 - \sum_{j \in J_i} (1 + x_j)x_j \right)^2$$

where $J_i = \{j \mid j \neq i, \max(1, i-5) \leq j \leq \min(n, i+1)\}$. Note that the number of terms in $\left(x_i(2 + 5x_i^2) + 1 - \sum_{j \in J_i}(1 + x_j)x_j\right)^2$ can be varied by changing J_i. We let

$$f_i(\boldsymbol{x}) \equiv \left(x_i(2 + 5x_i^2) + 1 - \sum_{j \in J_i} (1 + x_j)x_j \right)^2,$$

and vary the number of variables in $f_i(\boldsymbol{x})$ to investigate the performance of the sparse POP formulation (3) and the sparse PSDP formulation (7). The numerical results of the Broyden banded function are shown in Table 10. We used the relaxation order 3 for (7) because the relaxation order 2 did not provide accurate optimal solutions. The sparse PSDP formulation (7) provides accurate values indicated in the column of rel.err and performs better in terms of cpu time. The numbers shown in the columns of sizeA, #nzA, and sdpBl of (7) are smaller than those of (3).

The sparse POP formulation (3)							
k	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
5	7	3	1715×14400	14399	120(120.0)	6.0e-9	71.8
5	10	3	4091×57600	57596	120(120.0)	8.3e-8	351.2
5	15	3	8546×128025	128017	165(125.6)	2.9e-7	1158.5
The sparse PSDP formulation (7)							
k	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
5	7	3	2029×13702	20998	45(22.7)	2.3e-9	20.6
5	10	3	4130×28362	42858	45(27.3)	1.1e-8	46.8
5	15	3	8158×58099	85034	66(31.8)	1.5e-8	174.5

Table 10: Numerical experiments with Broyden banded functions

We now change J_i to observe the effects of the number of variables in each $f_i(\boldsymbol{x})$ upon sdpBl and sparsity of $\boldsymbol{A}$, and the performance of the two formulations. Because the number of indices in J_i determines the number of variables that appear in $f_i(\boldsymbol{x})$, we use varying k in $J_i = \{j \mid j \neq i, \max(1, i-k) \leq j \leq \min(n, i+1)\}$ to change the number of variables in $f_i(\boldsymbol{x})$. Table 11 shows the numerical results for $k = 3$. Notice that the sparse PSDP formulation (7) gives optimal solutions faster than the sparse POP formulation (3). We see smaller differences in sdpBl and the cpu time in Table 11 than in Table 10; sdpBl of (7) is about half of that of (3). We notice that sizeA and #nzA of (7) are smaller than those of (3).

The sparse POP formulation (3)							
k	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
3	7	3	965×9408	9405	56(56.0)	8.9e-9	5.4
3	10	3	1931×19600	19595	84(61.6)	4.8e-8	21.1
3	30	3	6761×81536	81510	56(56.0)	1.7e-7	46.8
3	100	3	24401×301056	300960	56(56.0)	5.5e-7	200.5
The sparse PSDP formulation (7)							
k	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
3	7	3	1387×8924	13624	28(19.1)	3.2e-9	7.9
3	10	3	2412×16096	24023	36(21.2)	1.8e-9	18.2
3	30	3	7761×48850	75790	28(22.2)	6.8e-9	44.9
3	100	3	27431×172610	267870	28(23.0)	1.1e-7	142.9

Table 11: Broyden banded functions with $k = 3$

With $k = 1$, as shown in Table 12, the sparse POP formulation (3) gives faster results than the sparse PSDP formulation (7), however, the accuracy of optimal solutions by (7) is higher than (3). Note that sizeA and #nzA of (3) are smaller than those of (7) though sdpBl of (3) is bigger than that of (7). This indicates that cpu time is more affected by sizeA and #nzA than sdpBl.

The sparse POP formulation (3)							
k	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
1	30	3	1595×11200	11172	20(20.0)	6.9e-8	1.9
1	100	3	5515×39200	39102	20(20.0)	1.3e-7	6.7
The sparse PSDP formulation (7)							
k	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
1	30	3	2778×16048	24584	15(13.1)	2.3e-9	11.4
1	100	3	9498×54828	84084	15(13.3)	9.9e-9	30.5

Table 12: Broyden banded functions with $k = 1$

The Watson function [18] is described as

$$\begin{aligned} f_i(\boldsymbol{x}) &= \sum_{j=1}^{m}(j-1)x_j y_i^{j-2} - (\sum_{j=1}^{m} x_j y_i^{j-1})^2 - 1)^2 - 1 \quad (i = 1, ...29) \\ f_{30}(\boldsymbol{x}) &= x_1, \quad f(\boldsymbol{x})_{31} = x_2 - x_1^2 - 1. \end{aligned}$$

The numerical results of the Watson function are shown in Table 13. Note that the difference in cpu time between the sparse POP formulation (3) with $m = 7$ and $\omega = 2$ and the sparse PSDP formulation (7) with $m = 7$ and $\omega = 1$ is small, and the rel.err of (3) is smaller than (7). For $n = 7$ and $\omega = 2$, (7) obtains more accurate optimal solution than (3) with $m = 7$ and $\omega = 2$ while taking more cpu time. We see that smaller sizeA and #nzA of (3) result in shorter cpu time. In the case of $n = 10$, (7) resulted in a smaller relative error with $\omega = 2$ than (3) with $\omega = 2$ and 3. In the case of $\omega = 4$ of (3), the size of $\boldsymbol{A}$ of the sparse POP formulation (3) was too large to handle, stopping in out of memory

The sparse POP formulation (3)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
7	2	329×2836	3276	36(9.9)	9.7e-4	4.1
7	3	791×21008	30072	36(36.0)	6.6e-5	32.7
10	2	1000×8756	9955	66(13.6)	3.4e-2	43.1
10	3	3002×97460	141009	66(66.0)	1.1e-1	1049.9
10	4	-	out of	memory	-	
The sparse PSDP formulation (7)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
7	1	66×2156	5011	8(4.8)	1.2e-1	3.1
7	2	4850×82744	328364	44(16.2)	7.6e-6	405.3
10	1	96×3829	8934	11(6.2)	1.0e+0	2.4
10	2	10862×217743	975265	77(23.8)	1.1e-5	3104.5

Table 13: Watson function

A difficult unconstrained optimization problem known as NP-complete is partitioning an integer sequence $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$. That is, if there exists $\boldsymbol{x} \in \{\pm 1\}^n$ such that $\boldsymbol{a}^T\boldsymbol{x} = 0$,

then the sequence can be partitioned. It can be formulated as

$$\min f(\boldsymbol{x}) = (\boldsymbol{a}^T\boldsymbol{x})^2 + \sum_{i=1}^{n}(x_i^2 - 1)^2. \tag{22}$$

Numerical results for several sequences of $\boldsymbol{a}$ are shown in [9]. We tested the sequences of $\boldsymbol{a}$ of large dimension among the problems included in [9]. Tables 14 and 15 show the numerical results for the sequences of dimension 10 and 11, respectively in [9]. The sparse PSDP formulation (7) in Tables 14 and 15 finds approximate solutions faster than the sparse POP formulation (3). Smaller values are displayed for sizeA and #nzA of (7) than those of (3). The solutions obtained by (7) for both sequence $\boldsymbol{a}$'s resulted in higher accuracy than the solutions in [9].

The sparse POP formulation (3)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
10	2	1000×8756	9955	66(13.6)	1.2e+0	37.8
10	3	3002×97460	141009	66(66.0)	1.2e+0	936.7
solution	(1.0000 -0.9996 1.0000 0.9991 0.9991 0.9991 -0.9997 0.9991 0.9991 -0.6099)					
10	3	-	out of	memory	-	
The sparse PSDP formulation (7)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
10	1	76×357	371	11(2.4)	9.5e-1	0.3
10	2	1158×8597	11934	67(5.4)	8.3e-2	65.5
solution	(1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 -0.8442)					
10	3	-	out of	memory	-	

Table 14: Numerical results for the problem of partitioning integer sequence $\boldsymbol{a} = (1, 2, 3, 20, 5, 6, 7, 10, 11, 77)$

The sparse POP formulation (3)						
n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
11	2	1364×11958	13530	78(14.9)	1.0e+0	95.5
11	3	4367×148644	215556	78(78.0)	1.0e+0	3490.3
solution	(1.0000 -0.9999 1.0000 -0.9998 -0.9998 -0.9998 -1.0000 -0.9998 -0.9998 0.7792 -1.0000)					
The sparse PSDP formulation (7)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
11	1	89×414	430	12(2.4)	1.0e+0	0.3
11	2	1543×11362	15594	79(5.5)	4.8e-2	169.4
solution	(1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 -0.8832 1.0000)					

Table 15: Numerical results for the problem of partitioning integer sequence $\boldsymbol{a} = (1, 2, 3, 20, 5, 6, 7, 10, 11, 77, 3)$

Solving the problem (22) of large dimension can be time-consuming because it does not appear to have any sparsity in (22). However, if the technique proposed in [11] is applied,

it can be solved efficiently. More precisely, let

$$P = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & a_n \\ 0 & a_2 & \cdots & a_{n-1} & a_n \\ 0 & 0 & \cdots & a_{n-1} & a_n \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & 0 & 0 & a_n \end{bmatrix},$$

and $P\boldsymbol{x} = \boldsymbol{y}$. Then, $\boldsymbol{x} = P^{-1}\boldsymbol{y}$, or,

$$\begin{aligned} x_i &= \frac{y_i - y_{i+1}}{a_i} \quad \text{for } i = 1, \ldots, n-1, \\ x_n &= \frac{y_n}{a_n} \end{aligned}$$

Consequently, (22) becomes

$$\min g(\boldsymbol{x}) = f(P^{-1}\boldsymbol{y}) = y_1^2 + \sum_{i=1}^{n-1} \left\{ \left(\frac{y_i - y_{i+1}}{a_i} \right)^2 - 1 \right\}^2 + \left\{ \left(\frac{y_n}{a_n} \right)^2 - 1 \right\}^2. \qquad (23)$$

We notice that cpu time in Table 16 and 17 is much smaller than than of Table 14 and 15, although the accuracy has deteriorated slightly. With the transformation, the sparse PSDP formulation (7) performs better than the sparse POP formulation (3) in finding approximate solutions with smaller relative errors. The formulation (7) with $m = 10$ and $\omega = 2$ has larger numbers in sizeA, #nzA and sdpBl than (3) with $m = 10$ and $\omega = 3$ taking longer to obtain a lower bound as shown in Table 16. Similar result is displayed in Table 17. The formulation (3) with $m = 10$ and $\omega = 2$ in Table 16 and (3) with $m = 11$ and $\omega = 2$ in Table 17 take slightly shorter cpu time than (7) with $m = 10$ and $\omega = 1$ in Table 16 and (7) with $m = 11$ and $\omega = 1$ in Table 17, respectively, but the rel.err is larger in (3). We also note that the difference in cpu time is small. In these cases, sizeA, #nzA, and sdpBl do not serve as the deciding factors for cpu time.

The sparse POP formulation (3)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
10	2	94×672	743	6(3.9)	1.3e+1	1.0
10	3	140×1304	1645	6(6.0)	1.3e+1	1.6
solution	not found					
The sparse PSDP formulation (7)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
10	1	40×213	265	3(2.4)	9.9e-1	2.1
10	2	263×2527	4174	9(5.2)	7.5e-2	13.4
solution	(-1.0052 -1.0030 -1.0043 -1.0279 -1.0072 -1.0087 -1.0101 -1.0143 -1.0157 0.8597)					

Table 16: Numerical results for the problem of partitioning integer sequence $\boldsymbol{a} = (1, 2, 3, 20, 5, 6, 7, 10, 11, 77)$ using the transformation

The sparse POP formulation (3)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
11	2	97×688	761	6(3.8)	1.3e+1	1.4
11	3	145×1339	1687	6(5.7)	1.3e+1	1.5
solution	not found					
The sparse PSDP formulation (7)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
11	1	44×234	292	3(2.4)	9.6e-1	2.4
11	2	290×2786	4619	9(5.3)	4.3e-2	13.8
solution	(1.0059 1.0030 1.0030 1.0217 1.0053 1.0065 1.0076 1.0109 1.0120 -0.8954 1.0000)					

Table 17: Numerical results for the problem of partitioning integer sequence $\boldsymbol{a} = (1, 2, 3, 20, 5, 6, 7, 10, 11, 77, 3)$ using the transformation

For additional test problems of partitioning sequences, we generated integer sequences randomly as follows. Let u and ν be positive integers, and let r be a random number in $(0, 1)$. Then, we create $a_i = \lceil r \cdot u \rceil$ for $i = 1, \ldots, \nu$ and compute $s = \sum_{i=1}^{\nu} a_i$. Next, $a_{\nu+1}, \ldots, a_m$ are generated such that $\sum_{i=\nu+1}^{m} a_i = s$. More precisely, $a_{\nu+1}, \ldots, a_{m-1}$ are computed by $a_i = \lceil r \cdot u \rceil$, and $a_m = s - \sum_{i=\nu+1}^{m-1} a_i$. Note that u decides the magnitude of a_i and ν the number of elements in the sequence. Table 18 displays the numerical results for a randomly generated integer sequence. In this case, increasing relaxation order did not result in higher accuracy in both of the sparse POP formulation (3) and the sparse PSDP formulation (7). Errors involved in the transformation may have caused the large relative error. We note, however, the signs of solution values are correct. The rel.err and cup time of (7) are smaller than (3). In Table 19, we see a big difference in cpu time between (3) and (7). The accuracy of the sparse POP formulation is slightly better.

The sparse POP formulation (3)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
13	2	124×888	980	6(3.9)	2.1e+1	0.9
	(-1.3190 -1.3151 1.2849 1.2988 1.4303 1.4421 -1.1039 1.3206 -1.0170 1.6722 -1.3672 -2.0442 1.0000)					
The sparse PSDP formulation (7)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
13	1	52×276	346	3(2.4)	7.6e-1	0.5
solution	(-0.9951 -0.9989 0.7459 0.9940 0.9986 0.9987 -0.9951 0.5032 -0.9900 0.9985 -0.9987 -0.9994 0.9999)					

Table 18: Numerical results for the problem of partitioning randomly generated integer sequence $\boldsymbol{a} = (3\ 1\ 2\ 1\ 1\ 1\ 1\ 3\ 3\ 2\ 1\ 3\ 4)$, $u = 3$, $\nu = 8$ using the transformation.

We use polynomial systems in [29] to solve the following problem:

$$\min \quad \sum_{i=1}^{n} f_i(\boldsymbol{x})^2 \quad \text{subj. to} \quad l_i \le x_i \le u_i \tag{24}$$

The sparse POP formulation (3)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
15	2	3875×33896	37720	136(19.9)	2.1e-2	2869.6
	(1.0000 -1.0000 1.0000 1.0000 1.0000 0.9998 -1.0000 -0.9999					
	-1.0000 0.9999 -1.0000 -1.0000 1.0000 0.9999 -0.9999)					
The sparse PSDP formulation (7)						
m	ω	sizeA	#nzA	sdpBl	rel.err	cpu
15	1	151×682	706	16(2.4)	7.9e-1	1.0
solution	(1.0000 -0.9998 0.9999 0.9996 0.9997 -0.9984 -0.9999 -0.3342					
	-0.9998 0.9998 -0.9999 -0.9998 0.9999 0.9997 -0.9995)					

Table 19: Numerical results for the problem of partitioning randomly generated integer sequence $\boldsymbol{a} = (\ 3\ 1\ 2\ 1\ 1\ 1\ 1\ 3\ 3\ 2\ 1\ 3\ 3\ 3\ 4)$, $u = 3$, $k = 9$.

where $f_i : \mathbb{R}^n \to \mathbb{R}$ represents ith equation of polynomial system, l_i and u_i denote lower and upper bounds for x_i, respectively. Many numerical methods exist for solving a system of polynomial $\boldsymbol{f}(\boldsymbol{x}) = 0$. One of the most successful methods is the polyhedral homotopy continuation method [21], which provides all isolated complex solutions of $\boldsymbol{f}(\boldsymbol{x}) = 0$. When one or some of isolated real solutions in a certain interval are to be found, it is more reasonable to formulate the problem as (24). We must say, however, that any comparison of the presented method with the polyhedral continuation method is not of our interest; the state-of-art software package [22] for the polyhedral homotopy continuation method computes all complex solutions of economic-n and cyclic-n polynomial systems much faster than the presented method that computes a single solution of (24). The main concern here is comparing the sparse POP formulation (3) with the sparse PSDP formulation (7) through polynomial systems.

Values given for lower bounds l_i and upper bounds u_i for variables x_i $(i = 1, 2, \ldots, n)$ are crucial to have the convergence to an optimal value. See Section 5.6 of [31]. When appropriate values for the bounds are not known in advance, we simply assign a very large number and a very smaller number, for instance, 1.0e+10 and -1.0e+10, to the bounds and solve (24). If an optimal value of desired accuracy is not obtained, then the attained optimal solution values are used for the lower and upper bounds after perturbing the values slightly. Then, the problem is solved again.

The two formulations are compared numerically in Table 20. We use $f_i(\boldsymbol{x})$ from the corresponding polynomial whose name is described in the first column [29]. The number in the column "iter" indicates the number of times that the problem is solved with updated lower and upper bounds; 1 means initial application of the sparse POP formulation (3) or the sparse PSDP formulation (7) for the problem (24). The initial bounds for the variables were given as $[-5, 5]$ for the tested problems. As shown in Table 20, (7) outperforms (3) in obtaining optimal solutions in less cpu time. In cyclic-6, (3) resulted in out of memory because sizeA was too large to handle.

The sparse POP formulation (3)									
Prob.	iter	deg	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
eco-6	1	3	6	3	506×11852	16549	38(28.8)	2.9e-2	7.2
eco-6	2	3	6	3	506×11852	16549	38(28.8)	2.2e-13	4.0
eco-8	1	3	8	3	1441×29566	41709	66(46.6)	1.9e-11	98.3
eco-10	1	3	10	3	3382×63586	89715	102(68.8)	5.5e-9	1319.3
cyclic-5	1	5	5	5	3002×228258	307632	252(137.5)	8.3e-14	1789.0
cyclic-6	1	6	6	6	-	out of	memory	-	-
The sparse PSDP formulation (7)									
Prob.	iter	deg	n	ω	sizeA	#nzA	sdpBl	rel.err	cpu
eco-6	1	3	6	2	265×2511	4244	14(6.9)	5.8e-3	1.7
eco-6	2	3	6	2	265×2511	4307	14(6.9)	3.7e-9	1.4
eco-8	1	3	8	2	529×4713	8267	18(8.5)	3.7e-9	3.9
eco-10	1	3	10	2	867×7908	14258	22(10.1)	4.0e-9	7.9
cyclic-5	1	5	5	3	2771×50700	83077	84(42.1)	3.3e-9	148.8
cyclic-6	1	6	6	3	2187×54084	148153	72(38.3)	5.8e-2	230.6

Table 20: Polynomials

6 Concluding remarks

We have discussed various ways of formulating polynomial least problems as polynomial SDPs, and presented an efficient polynomial SDP formulation after comparing the degree of polynomials, and the sizes of the moment and the localizing matrices. Solving the polynomial SDP is expected to provide the computational efficiency over solving the given form of polynomial least squares problem because the degree of polynomials in the former formulation is smaller than the degree of polynomials in the latter.

Numerical tests performed on various test problems show that the size of the coefficient matrix $\boldsymbol{A}$, the number of nonzero elements of $\boldsymbol{A}$ and the size of SDP blocks of $\boldsymbol{A}$ are important factors on computational efficiency. Overall performance of the polynomial SDP formulation is shown to be better than the POP formulation except a few cases.

We finally note that our discussion on formulating polynomial least squares problem (1) as a polynomial SDP can be extended to a constrained problem of the form:

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i\in M} f_i(\boldsymbol{x})^{2p_i} \\ \text{subject to} & g_j(\boldsymbol{x}) \geq 0 \ (j = 1, \ldots, \widehat{m}), \end{array}\right\} \tag{25}$$

where $f_i(\boldsymbol{x})$ and $g_j(\boldsymbol{x})$ are polynomials in $\boldsymbol{x} \in \mathbb{R}^n$.

References

[1] J. R. S. Blair and B. Peyton, "An introduction to chordal graphs and clique trees", in *Graph Theory and Sparse Matrix Computation*, A. George, J. R. Gilbert and J. W. H. Liu, eds., Springer-Verlag, New York, 1993, pp. 1-29.

[2] B. Borchers, "SDPLIB 1.2, a library of semidefinite programming test problems", *Optimization Methods and Software*, **11** & **12** (1999) 683-690.

[3] A. R. Conn, N. I. M. Gould and P. L. Toint, "Testing a class of methods for solving minimization problems with simple bounds on the variables", *Math.Comp.*, **50** (1988) 399–430

[4] K. Fujisawa, M. Kojima, K. Nakata (1995) SDPA (SemiDefinite Programming Algorithm) user's manual, Version 5.0, Research Report B-308, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[5] N. I. M. Gould, D. Orban and Ph. L. Toint, (2003) "Cuter, a Constrained and Unconstrained Testing Environment, revisited", *TOMS*, **29** 373-394.

[6] D. Henrion and J. B. Lasserre, "GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi", Laboratoire d'Analyse et d'Architecture des Syst'emes, Centre National de la Recherche Scientifique, 7 Avenue du Colonel Roche, 31 077 Toulouse, cedex 4, France, February 2002.

[7] D. Henrion and J. B. Lasserre, Convergent relaxations of polynomial matrix inequalities and static output feedback, *IEEE Trans. Automat. Cont.* **51** (2), 192-202 (2006).

[8] C. W. Hol, C. W. Scherer, Sums of squares relaxations for polynomial semi-definite programming, In: B. De Moor, B. Motmans (eds), Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems, Leuven,, Belgium, 5-9 July, 1-10 (2004).

[9] D. Jibetean and M. Laurent, "Semidefinite approximation for global unconstrained polynomial optimization," *SIAM J. Optim,*, **16**, 2 (2005) 490-514.

[10] S. Kim, M. Kojima and H.Waki, "Generalized Lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems", *SIAM J. Optim.*, **15** (2005) 697-719 .

[11] S. Kim, M. Kojima and Ph.L. Toint, "Recognizing underlying sparsity," esearch Report B-428, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[12] K. Kobayashi, S. Kim, and M. Kojima (2006) "Correlative sparsity in primal-dual interior point methods for LP, SOCP and SDP", Research Report B-434, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[13] K. Kobayashi, S. Kim, and M. Kojima (2007) "Sparse second order cone programming approaches for convex opimitization problems", Research Report B-440, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[14] M. Kojima, S. Kim and H. Waki, "Sparsity in sums of squares of polynomials", *Math. Program.*, **103** (2005) 45-62.

[15] M. Kojima, Sums of squares relaxations of polynomial semidefinite programs, Research Report B-397, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan (2003).

[16] M. Kojima and M. Muramatsu, An extension of sums of squares relaxations to polynomial optimization problems over symmetric cones, *Mathematical Programming*, **110**, (2007) 315-326.

[17] M. Kojima and M. Muramatsu, A note on sparse SOS and SDP relaxations for polynomial optimization problems over yymmetric cones, *Computational Optimization and Applications* to appear.

[18] J. S. Kowalik and M. R. Osborne, *Methods for unconstrained optimization problems,* Elseview North-Halland, New York, (1968).

[19] J. B. Lasserre, "Global optimization with polynomials and the problems of moments", *SIAM Journal on Optimization*, **11** (2001) 796–817.

[20] J. B. Lasserre: Convergent SDP-relaxations in polynomial optimization with sparsity, *SIAM Journal on Optimization*, **17**, 3 (2006) 822-843.

[21] T. Y. Li, "Solving polynomial systems by polyhedral homotopies", *Taiwan Journal of Mathematics* **3** (1999) 251–279.

[22] T. Y. Li, "HOM4PS in Fortran", http://www.mth.msu.edu/ li/

[23] J. J. More, B. S. Garbow and K. E. Hillstrom, "Testing Unconstrained Optimization Software", *ACM Trans. Math. Soft.*, **7**, (1981) 17–41

[24] S. G. Nash, "Newton-Type Minimization via the Lanczos method", *SIAM J.Numer. Anal.*,**21** (1984) 770–788.

[25] J. Nocedal and S. J. Wright (2006) *Numerical Optimization*, Springer.

[26] S. Prajna, A. Papachristodoulou and P. A. Parrilo, "SOSTOOLS: Sum of Squares Optimization Toolbox for MATLAB – User's Guide", Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125 USA, 2002.

[27] F. J. Sturm (1999) " Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones", *Optimization Methods and Software* **11-12**, 625–653.

[28] K. Toh, M. J. Todd, R. H. Tütüntü (1998) SDPT3 — a MATLAB software package for semidefinite programming, Dept. of Mathematics, National University of Singapore, Singapore.

[29] Test suite of polynomial systems "http://www.math.uic.edu/ jan".

[30] H. Waki, S. Kim, M. Kojima and M. Muramatsu, "SparsePOP : a Sparse Semidefinite Programming Relaxation of Polynomial Optimization Problems", Research report B-414, Dept. of Math. & Computing Sciences, Tokyo Institute of Technology, March 2005.

[31] H. Waki, S. Kim, M. Kojima and M. Muramatsu, (2006) "Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity", SIAM Journal on Optimization **17** (1) 218-242.