Ksenia Bestuzheva[1], Helena Völker, Ambros Gleixner[2]

Strengthening SONC Relaxations with Constraints Derived from Variable Bounds

[1] 0000-0002-7018-7099
[2] 0000-0003-0391-5903

ZIB Report 23-03 (January 2023)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30 84185-0
Telefax: +49 30 84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Strengthening SONC Relaxations with Constraints Derived from Variable Bounds

Ksenia Bestuzheva[1], Helena Völker[2], and Ambros Gleixner[3]

[1]Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany,
`bestuzheva@zib.de`
[2]Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany,
`voelker@zib.de`
[3]Zuse Institute Berlin and HTW Berlin, Germany,
`gleixner@zib.de`

January 19, 2023

Abstract

Nonnegativity certificates can be used to obtain tight dual bounds for polynomial optimization problems. Hierarchies of certificate-based relaxations ensure convergence to the global optimum, but higher levels of such hierarchies can become very computationally expensive, and the well-known sums of squares hierarchies scale poorly with the degree of the polynomials. This has motivated research into alternative certificates and approaches to global optimization. We consider sums of nonnegative circuit polynomials (SONC) certificates, which are well-suited for sparse problems since the computational cost depends on the number of terms in the polynomials and does not depend on the degrees of the polynomials. We propose a method that guarantees that given finite variable domains, a SONC relaxation will yield a finite dual bound. This method opens up a new approach to utilizing variable bounds in SONC-based methods, which is particularly crucial for integrating SONC relaxations into branch-and-bound algorithms. We report on computational experiments with incorporating SONC relaxations into the spatial branch-and-bound algorithm of the mixed-integer nonlinear programming framework SCIP. Applying our strengthening method increases the number of instances where the SONC relaxation of the root node yielded a finite dual bound from 9 to 330 out of 349 instances in the test set.

1 Introduction

Polynomial optimization problems are a topic of active research in the fields of algebraic geometry and nonlinear optimization, with applications in dynamics and control [3, 23, 24], wireless coverage [10, 11], and economics and game theory [37], to name a few. These problems are, in general, nonlinear and nonconvex, making finding the global optimum a difficult task. Furthermore, polynomials with high degrees present a particular challenge since the nonlinearity is more pronounced in such polynomials. Existing optimization techniques often rely specifically on linear or quadratic structures in problems, and even those methods that focus on general polynomial optimization problems often scale poorly with the degree.

Global polynomial optimization methods can be roughly divided into two categories. General-purpose approaches such as spatial branch-and-bound [18, 16] are versatile and can call upon a variety of sophisticated techniques in order to speed up the solving process. However, these algorithms rely on linear or convex relaxations of the problem, and conventional relaxations lack the means to efficiently capture polynomial nonlinearities. Approaches based on nonnegativity certificates [36, 27, 21, 29] are better at leveraging the structure of an entire polynomial, as opposed to its monomial terms, in order to obtain stronger dual bounds. The search for the global optimum, though, requires constructing computationally expensive hierarchies of relaxations. The goal of this paper is to help bridge the existing gap between general-purpose algorithms and certificate-based relaxations by (i) formulating a class of valid constraints to strengthen sums of nonnegative circuit polynomials (SONC) relaxations in the presence of finite variable bounds, thus enabling such relaxations to benefit from decreasing domain sizes in nodes of a branch-and-bound tree, and (ii) developing a branch-and-bound algorithm that solves SONC relaxations next to linear relaxations.

General-purpose solution methods such as spatial branch-and-bound can solve polynomial optimization problems as long as some linear or convex relaxation is available, for example a linear relaxation constructed by outer approximating the power and product terms appearing in a polynomial. The Reformulation-Linearization Technique (RLT) [1, 2] can be employed to strengthen such methods. RLT produces a family of cutting planes in a lifted space, and it has been shown to yield strong relaxations of polynomial problems [35, 34, 12]. The solver RAPOSA [16] implements an RLT-based branch-and-bound algorithm aimed at efficiently solving polynomial optimization problems.

Another approach to polynomial optimization is based on results on nonnegativity of polynomials. Such results usually rely on the cone of polynomials that are representable as sums of squares (SOS), and originate in the works of Shor [36], Nesterov [27], Lasserre [21] and Parrilo [29]. A semidefinite program is solved in order to produce an SOS nonnegativity certificate for $f(x)-\gamma$, seeking to maximize γ. Further developments include improving the computational efficiency of SOS-based methods by exploiting sparsity [38, 39], employing restrictions of the SOS cone [4], as well as developing software for polynomial optimization such as GloptiPoly [17] and SOSTOOLS [28].

The SONC certificate, based on the decomposition of a polynomial into a sum of nonnegative circuit polynomials, was proposed by Iliman and de Wolff [19]. Unlike for SOS certificates, the computational cost of computing a SONC certificate does not depend on the degree of the polynomial. Furthermore, since the cones of SOS and SONC polynomials do not coincide or contain one another [19], for some problems a SONC decomposition yields better bounds than an SOS decomposition.

The cone of sums of arithmetic geometric mean exponentials (SAGE) [8, 9] provides an alternative view of the SONC cone. The membership of a polynomial in the SAGE cone can be decided by solving a convex relative entropy program, and the method was successfully extended to constrained optimization problems [26].

These cones of nonnegative polynomials, however, in general only provide a dual bound. While for SOS polynomials, there is a hierarchy of relaxations that converges to the global optimum [21], solving high levels of this hierarchy can have a prohibitive computational cost. An alternative approach to global optimization via nonnegativity certificates is to combine them with a branch-and-bound algorithm. The first such integrated algorithm was proposed by Seidler [31]. This algorithm branches on signs of variables, so that additional terms can be identified as positive. However, convergence to the global optimum is not guaranteed.

In this work, we continue to investigate the potential for combining SONC-based relaxations with branch-and-bound algorithms. To this end, we first address the difficulties in utilizing variable bounds when applying SONC relaxations. We employ a Lagrangian relaxation approach to build the SONC relaxations of constrained optimization problems and strengthen them by polynomial constraints derived from variable bounds. We construct these constraints in a way that aims at achieving a structure of the exponent set of the Lagrangian function that is well-suited for obtaining a SONC certificate. Adding these constraints enables the root node SONC relaxation to obtain finite dual bounds for 330 out of 349 instances from our test set, whereas the standard SONC relaxation found a finite dual bound only for 9 instances.

The second contribution of this paper is an implementation of SONC relaxations within a general-purpose spatial branch-and-bound algorithm. We implement these relaxations in a relaxator plugin of the MINLP framework SCIP [5], which applies a preprocessing step, adds polynomial-bound constraints and calls the polynomial optimization software POEM [33] in order to obtain SONC certificates. Although our experiments showed that SONC relaxations are not yet competitive with state-of-the-art linear relaxations, we observed a considerable improvement in the root node dual bound on 6 instances, with SONC relaxations closing up to 91% of the gap as compared to linear relaxations.

The rest of the paper has the following structure. Sections 2 and 3 provide a summary of the theory of SONC certificates and their use for polynomial optimization. Section 4 presents the main theoretical contribution of the paper: a new method for incorporating variable bounds into a SONC relaxation. In Section 5, we discuss the implementation of SONC relaxations within the spatial branch-and-bound algorithm of SCIP. Finally, in Section 6, we report the results of our computational experiments.

2 SONC Certificates

Consider a constrained polynomial optimization problem of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x}) = \sum_{\alpha\in\mathcal{A}(f)} f_\alpha \mathbf{x}^\alpha \tag{1a}$$

$$\text{s.t.} \quad g_i(\mathbf{x}) = \sum_{\alpha\in\mathcal{A}(g_i)} g_{i,\alpha}\mathbf{x}^\alpha \geq 0, \qquad i = 1,\dots,m, \tag{1b}$$

where $f_\alpha, g_{i,\alpha} \in \mathbb{R}$ are nonzero coefficients of the polynomials and $\mathcal{A}(f)$, $\mathcal{A}(g_i) \subset \mathbb{N}^n$, $i = 0,\dots,m$, are supports of polynomials f and g_i.

In this formulation, monomials are written as $\mathbf{x}^\alpha := x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$. An exponent α is a *monomial square* if the term $f_\alpha \mathbf{x}^\alpha$ satisfies $f_\alpha \geq 0$ and $\alpha \in (2\mathbb{N})^n$. The *Newton polytope* $\mathrm{New}(f)$ of a polynomial f with support $\mathcal{A}(f)$ is defined to be the convex hull of the exponents of f, that is $\mathrm{New}(f) = \mathrm{conv}(\mathcal{A}(f))$. Let $V(f)$ denote the vertices of $\mathrm{New}(f)$ and let $\Delta(f) = \mathcal{A}(f) \setminus \mathrm{V}(f)$ denote the set of all non-vertex exponents. We will refer to terms that correspond to exponents in $\Delta(f)$ as inner terms of f. Further, we will denote the set of exponents that correspond to monomial square terms as $\mathrm{MoSq}(f)$, and the remaining set of exponents as $\overline{\mathrm{MoSq}}(f) = \mathcal{A}(f) \setminus MoSq(f)$.

SONC certificates, similarly to SOS certificates, utilize a decomposition of a polynomial into a sum of polynomials of a special structure, such that nonnegativity of such polynomials is easy to prove. In the case of SONC, these basic building blocks are circuit polynomials [19]:

Definition 1. *A circuit polynomial is a polynomial of the form*

$$f(\mathbf{x}) = \sum_{\alpha \in V(f)} f_\alpha \mathbf{x}^\alpha + f_\beta \mathbf{x}^\beta, \tag{2}$$

where the vertices $\alpha \in V(f)$ are affinely independent and are monomial squares, that is, $V(f) \subseteq MoSq(f)$.

The exponent β of the inner term of a circuit polynomial can be uniquely written as a convex combination of vertices:

$$\sum_{\alpha \in V(f)} \lambda_\alpha^{(\beta)} = 1 \quad \text{and} \quad \sum_{\alpha \in V(f)} \lambda_\alpha^{(\beta)} \alpha = \beta. \tag{3}$$

The weights $\lambda_\alpha^{(\beta)}$ are referred to as *barycentric coordinates* of β. For a circuit polynomial, one can compute the circuit number

$$\theta_f(\beta) = \prod_{\alpha \in V(f)} \left(\frac{f_\alpha}{\lambda_\alpha^{(\beta)}} \right)^{\lambda_\alpha^{(\beta)}}. \tag{4}$$

Nonnegativity of a circuit polynomial can be decided by comparing the circuit number to the coefficient of the inner term [19]:

Theorem 1. *A circuit polynomial f as given in (2) is nonnegative if and only if*

$$|f_\beta| \leq \theta_f(\beta) \text{ and } \beta \notin (2\mathbb{N})^n \qquad \text{or} \qquad f_\beta \geq -\theta_f(\beta) \text{ and } \beta \in (2\mathbb{N})^n.$$

By utilizing circuit polynomials, Iliman and de Wolff [19] proposed a new class of nonnegative polynomials:

Definition 2. *A polynomial f is a SONC polynomial if it is of the form*

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} c_i f_i(\mathbf{x}) \tag{5}$$

where $c_i \geq 0$ are nonnegative coefficients and f_i are nonnegative circuit polynomials for all $i = 1, \ldots, \ell$.

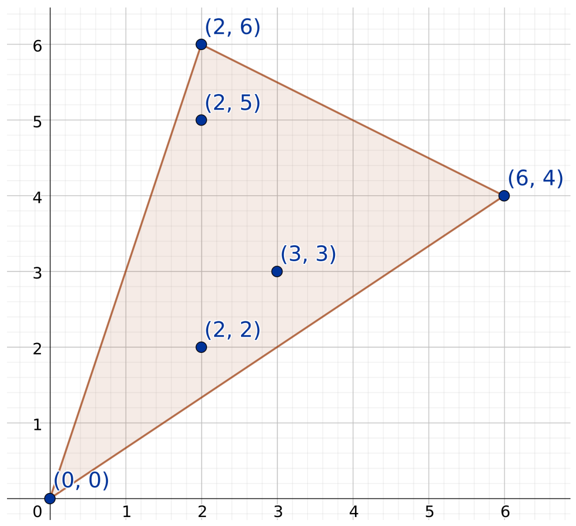

Figure 1: Exponents and the Newton polytope of the polynomial $f(x_1, x_2) = x_1^6x_2^4 + x_1^3x_2^3 + x_1^2x_2^6 + x_1^2x_2^5 + x_1^2x_2^2 + 1$.

Another class of polynomials that is of interest in the context of SONC decompositions is the class of ST-polynomials. Similarly to circuit polynomials, the vertices of ST-polynomials are monomial squares that are affinely independent, or, in other words, form a simplex. The difference is that ST-polynomials can have multiple inner terms.

Definition 3. *A polynomial f is an* ST-polynomial *[20] if it has the form*

$$f(\mathbf{x}) = \sum_{\alpha \in V(f)} f_\alpha \mathbf{x}^\alpha + \sum_{\beta \in \Delta(f)} f_\beta \mathbf{x}^\beta \tag{6}$$

such that New(f) is a simplex whose vertices $\alpha \in V(f)$ are monomial squares, that is, $V(f) \subseteq$ MoSq(f).

For all $\beta \in \Delta(f)$ there exist $\lambda_\alpha^{(\beta)} \geq 0$, $\alpha \in V(f)$, forming the unique convex combination (3). The vertex set $V(f)$ of the simplex is referred to as a *cover* of the inner term β. We will say that β is *covered* by $V(f)$.

Figure 1 shows points corresponding to the exponents of the polynomial $f(x_1, x_2) = x_1^6x_2^4 + x_1^3x_2^3 + x_1^2x_2^6 + x_1^2x_2^5 + x_1^2x_2^2 + 1$. The coloured region represents the Newton polytope. Since the vertices $(6, 4)$, $(2, 6)$ and $(0, 0)$ are even, correspond to terms with positive coefficients and are affinely independent, $f(x_1, x_2)$ is an ST-polynomial. Exponents of the inner terms $(2, 5)$, $(3, 3)$ and $(2, 2)$ can be expressed as unique convex combinations of the vertices.

It is possible to split an ST-polynomial into circuit polynomials by taking the same Newton polytope for each inner term and splitting the coefficients of the monomial squares among the circuit polynomials. The existence of such a decomposition is a proof of nonnegativity.

Theorem 2. *([20, Theorem 3.1]) An ST-polynomial $f(\mathbf{x})$ is a SONC polyno-*

mial if for every $(\beta, \alpha) \in \Delta(f) \times V(f)$ there exist $a_{\beta,\alpha} \geq 0$ such that

$$|f_\beta| \leq \prod_{\alpha \in nz(\beta)} \left(\frac{a_{\beta,\alpha}}{\lambda_\alpha^{(\beta)}} \right)^{\lambda_\alpha^\beta},$$

$$f_\alpha \geq \sum_{\beta \in \Delta(f)} a_{\beta,\alpha}.$$

These $a_{\beta,\alpha}$ are the weights in the following SONC decomposition:

$$f(\mathbf{x}) = \sum_{\beta \in \Delta(f)} \left(\sum_{\alpha \in nz(\beta)} a_{\beta,\alpha} \mathbf{x}^\alpha + f_\beta \mathbf{x}^\beta \right),$$

where $nz(\beta)$ denotes all exponents $\alpha \in V(f)$ that correspond to nonzero barycentric coordinates $\lambda_\alpha^{(\beta)}$.

3 Polynomial Optimization via SONC

In this section we restate known results on SONC polynomials and optimization methods based on SONC relaxations. Polynomial optimization problems can be stated as nonnegativity problems, since one can write the problem of minimizing $f(\mathbf{x})$ equivalently as

$$\sup\{\gamma \in \mathbb{R} : \quad f(\mathbf{x}) - \gamma \geq 0\}. \tag{7}$$

This problem, however, is as hard as the original problem. Requiring instead that some certificate of nonnegativity exists for $f(\mathbf{x}) - \gamma$ results in a relaxation of the original problem. If a SONC decomposition is used as a nonnegativity certificate, then the relaxation is

$$\sup\{\gamma \in \mathbb{R} : f(\mathbf{x}) - \gamma \text{ is SONC}\}. \tag{8}$$

3.1 Lower Bounds for Unconstrained Optimization over an ST-Polynomial

This section shows how to write the problem of finding the optimal SONC decomposition of an unconstrained polynomial optimization problem as a convex optimization problem. We begin by recalling the notion of geometric programs (GPs).

Definition 4. *[7] A monomial is defined as a function $q : \mathbb{R}^n \to \mathbb{R}$ of the form $q(\mathbf{x}) = c x_1^{a_1} \cdot \ldots \cdot x_n^{a_n}$ with $c > 0$. A posynomial is defined as a sum of monomials. A geometric program (GP) is an optimization problem of the form*

$$\begin{array}{lll} \text{minimize} & p_0(\mathbf{x}) & \\ \text{subject to} & p_i(\mathbf{x}) \leq 1, & \textit{for } i = 1, \ldots, r, \\ & q_j(\mathbf{x}) = 1, & \textit{for } j = 1, \ldots, s, \end{array} \tag{9}$$

where $p_0, \ldots, p_r$ are posynomials and $q_1, \ldots, q_s$ are monomials.

Applying a logarithmic transformation to a GP results in a convex optimization problem [6] which is equivalent to the GP; in particular, applying a reverse transformation to an optimal solution of the transformed problem gives an optimal solution of the original GP.

Let f be an ST-polynomial and assume that $\Delta(f) \cap \mathrm{MoSq}(f) = \emptyset$. This can be achieved by disregarding all monomial squares that are not vertices of $\mathrm{New}(f)$. This change preserves the validity of a lower bound on the optimal solution f^* since monomial squares are always nonnegative [15]. The following theorem formulates a GP based on the nonnegativity conditions from Theorem 2.

Theorem 3. *[15, Corollary 2.7] Let f be an ST-polynomial and introduce variables $a_{\beta,\alpha} > 0$ for every $\beta \in \Delta(f)$ and $\alpha \in V(f)$. Then the SONC lower bound on f is given by $f_{SONC} = f_0 - \gamma$, where γ is the solution of the following GP:*

$$\text{minimize} \quad \sum_{\substack{\beta \in \Delta(f) \\ \lambda_0^{(\beta)} \neq 0}} \lambda_0^{(\beta)} \cdot |f_\beta|^{\frac{1}{\lambda_0^{(\beta)}}} \cdot \prod_{\substack{j \in nz(\beta) \\ \alpha \neq 0}} \left(\frac{\lambda_\alpha^{(\beta)}}{a_{\beta,\alpha}} \right)^{\frac{\lambda_\alpha^{(\beta)}}{\lambda_0^{(\beta)}}} \tag{10a}$$

$$\text{subject to} \quad \sum_{\beta \in \Delta(f)} \frac{a_{\beta,\alpha}}{f_\alpha} \leq 1 \textit{ for all } \alpha \in V(f) \setminus 0, \tag{10b}$$

$$|f_\beta| \prod_{\alpha \in nz(\beta)} \left(\frac{\lambda_\alpha^{(\beta)}}{a_{\beta,\alpha}} \right)^{\lambda_\alpha^{(\beta)}} \leq 1 \textit{ for all } \beta \in \Delta(f) \textit{ with } \lambda_0^{(\beta)} = 0. \tag{10c}$$

3.2 Lower Bounds for Constrained Optimization with an ST-Polynomial Lagrangian Function

Dressler et al. [15] extend the method described above to the constrained case by utilizing Lagrangian relaxations. Consider an optimization problem of the form (1). Then the Lagrangian function of the problem has the form

$$L(\mathbf{x}, \mu) = f(\mathbf{x}) - \sum_{i=1}^{m} \mu_i g_i(\mathbf{x}), \tag{11}$$

where $\mu_i \geq 0$, $i = 1, \ldots, m$ are Lagrangian multipliers. Let $\mu_0 := 1$ and $g_0 := -f$. The Lagrangian then becomes

$$L(\mathbf{x}, \mu) = -\sum_{i=0}^{m} \mu_i g_i(\mathbf{x}). \tag{12}$$

The coefficients of the polynomial (12) depend on μ, and term cancellation may lead to some of the monomials vanishing for certain values of μ. Therefore, the definitions of exponent sets need to account for this. Thus, the support $\mathcal{A}(L)$ is defined as the union of supports of individual polynomials defining the objective and the constraints: $\mathcal{A}(L) = \cup_{i=0}^{m} \mathcal{A}(g_i)$. The definitions of the vertices of the Newton polytope and the exponents corresponding to inner terms are analogous: $V(L) = \cup_{i=0}^{m} V(g_i)$ and $\Delta(L) = \cup_{i=0}^{m} \Delta(g_i)$.

We assume that $\Delta(L)$ does not contain exponents corresponding to monomial square terms. If this is not the case, then, since monomial square terms are always nonnegative, we can disregard monomial square inner terms and still obtain a valid lower bound.

Further, we assume that L is an ST-polynomial, and write it in the form

$$L(\mathbf{x},\mu) = \sum_{\alpha\in V(L)} L(\mu)_\alpha \mathbf{x}^\alpha + \sum_{\beta\in\Delta(L)} L(\mu)_\beta \mathbf{x}^\beta. \tag{13}$$

For a fixed μ, the problem of optimizing the Lagrangian reduces to a similar GP that is solved in the unconstrained case. Let $\gamma(\mu)$ denote the optimal value of problem (10) for a given μ. The goal is to find the best possible lower bound γ^* over all nonnegative μ, that is, $\gamma^* = \sup\{\gamma(\mu) : \mu \in \mathbb{R}^n_+\}$. However, directly writing this in the form (10) does not produce a GP, and additional relaxation steps are necessary in order to obtain a GP [15].

As before, let $\lambda_\alpha^{(\beta)}$ denote the barycentric coordinates for each $\beta \in \Delta(L)$ with respect to the vertices $\alpha \in V(L)$. For each $\beta \in \Delta(L)$, we introduce new variables $a_{\beta,\alpha} > 0$ for each $\alpha \in V(L)$, and $b_\beta \geq 0$. For each constraint $g_i(\mathbf{x})$, $i = 1,\dots,m$, we denote by $g_{i,\alpha}$ the coefficient corresponding to the monomial $\mathbf{x}^\alpha$. Then optimizing the Lagrangian over μ and $\mathbf{x}$ is equivalent to solving the following optimization problem with variables μ, a_β and b_β:

$$\begin{aligned}
\underset{\substack{\mu\geq 0,\\ a_\beta>0, b_\beta\geq 0}}{\text{minimize}} \quad & \sum_{i=1}^m \mu_i g_{i,0} + \sum_{\substack{\beta\in\Delta(L)\\ \lambda_0^{(\beta)}\neq 0}} \lambda_0^{(\beta)} \cdot b_\beta^{\frac{1}{\lambda_0^{(\beta)}}} \cdot \prod_{\substack{\alpha\in nz(\beta)\\ \alpha\neq 0}} \left(\frac{\lambda_\alpha^{(\beta)}}{a_{\beta,\alpha}}\right)^{\frac{\lambda_\alpha^{(\beta)}}{\lambda_0^{(\beta)}}} && \text{(14a)}\\
\text{subject to} \quad & \sum_{\beta\in\Delta(L)} a_{\beta,\alpha} \leq L(\mu)_\alpha \text{ for all } \alpha\neq 0, && \text{(14b)}\\
& \prod_{\alpha\in nz(\beta)} \left(\frac{a_{\beta,\alpha}}{\lambda_\alpha^{(\beta)}}\right)^{\lambda_\alpha^{(\beta)}} \geq b_\beta \text{ for all } \beta\in\Delta(L) \text{ with } \lambda_0^{(\beta)} = 0, && \text{(14c)}\\
& |L(\mu)_\beta| \leq b_\beta \text{ for all } \beta\in\Delta(L) \text{ with } \lambda_0^{(\beta)} \neq 0. && \text{(14d)}
\end{aligned}$$

As before, $nz(\beta)$ denotes all exponents $\alpha \in V(L)$ with non-zero barycentric coordinates in the convex combination that forms β, i.e. $nz(\beta) = \{\alpha \in V(L) \mid \lambda_\alpha^{(\beta)} \neq 0\}$. This optimization problem was stated in [20] and the following statements were shown, see [20, Theorem 5.1 and 5.2].

Theorem 4. *Let L be an ST-polynomial of the form (13). Assume that every coefficient $L(\mu)_\alpha$ consists of only one summand for each $\alpha \in V(L)$ and is strictly positive. Moreover, assume that for each $\beta \in \Delta(L)$ the coefficient $L(\mu)_\beta$ has only positive terms and $g_{i,0} \geq 0$ for all $i = 1,\dots,m$. Then the problem (14) is a GP for $\mu > 0$.*

The optimization problem (14) yields a lower bound for problem (1). More precisely, if γ^* is the solution of the GP (14) and f^* is the solution of the original problem (1), then we have

$$f_0 - \gamma^* \leq f^*.$$

However, Theorem 4 places considerable restrictions on the coefficients. Hence, a relaxation of this problem was introduced in [15]. The key step is to redefine the objective (14a) and the constraint (14d) in order to obtain a GP under weaker assumptions.

First, we split up the coefficients $L(\mu)_\beta$ into their positive and their negative parts, defining $L(\mu)_\beta = L(\mu)^+_\beta - L(\mu)^-_\beta$ with

$$L(\mu)^+_\beta = \sum_{\substack{i=1,\dots,m \\ g_{i,\beta}>0}} \mu_i g_{i,\beta} \quad \text{and} \quad L(\mu)^-_\beta = -\sum_{\substack{i=1,\dots,m \\ g_{i,\beta}<0}} \mu_i g_{i,\beta},$$

where $g_{i,\beta}$ denotes the coefficient corresponding to $\mathbf{x}^\beta$ of g_i. Using this decomposition, constraint (14d) can be changed to $\max\{L(\mu)^+_\beta, L(\mu)^-_\beta\} \leq b_\beta$. Further, in the objective function, instead of all $g_{i,0}$ for $i = 1, \dots, m$, we only consider those coefficients of the monomial $\mathbf{x}^0$ that are positive in the corresponding g_i and thus negative in $L(\mathbf{x}, \mu)$, resulting in $g^+_{i,0} = \max\{g_{i,0}, 0\}$. This produces the following optimization problem:

$$\underset{\substack{\mu\geq 0, \\ a_\beta>0, b_\beta\geq 0}}{\text{minimize}} \quad \sum_{i=1}^{m} \mu_i g^+_{i,0} + \sum_{\substack{\beta\in\Delta(L) \\ \lambda_0^{(\beta)}\neq 0}} \lambda_0^{(\beta)} \cdot b_\beta^{\frac{1}{\lambda_0^{(\beta)}}} \cdot \prod_{\substack{\alpha\in nz(\beta) \\ \alpha\neq 0}} \left(\frac{\lambda_\alpha^{(\beta)}}{a_{\beta,\alpha}}\right)^{\frac{\lambda_\alpha^{(\beta)}}{\lambda_0^{(\beta)}}} \tag{15a}$$

$$\text{subject to} \quad \sum_{\beta\in\Delta(L)} a_{\beta,\alpha} \leq L(\mu)_\alpha \text{ for all } \alpha \neq 0, \tag{15b}$$

$$\prod_{\alpha\in nz(\beta)} \left(\frac{a_{\beta,\alpha}}{\lambda_\alpha^{(\beta)}}\right)^{\lambda_\alpha^{(\beta)}} \geq b_\beta \text{ for all } \beta \in \Delta(L) \text{ with } \lambda_0^{(\beta)} = 0, \tag{15c}$$

$$L(\mu)^+_\beta \leq b_\beta \text{ for all } \beta \in \Delta(L), \tag{15d}$$

$$L(\mu)^-_\beta \leq b_\beta \text{ for all } \beta \in \Delta(L). \tag{15e}$$

This optimization problem is a relaxation of (14). The following theorem [15] shows that it is a GP under weaker assumptions.

Theorem 5. *Let L be an ST-polynomial of the form (13). Assume that for every $\alpha \in V(L) \setminus 0$ the coefficient $L(\mu)_\alpha$ has exactly one strictly positive term. Then the optimization problem (15) is a geometric program for $\mu > 0$. Moreover, the GP provides a lower bound on the solution of the original optimization problem (1). Denoting the solution of (15) by γ_{SONC} and the solution of (14) by γ^*, we get*

$$f_0 - \gamma_{SONC} \leq f_0 - \gamma^* \leq f^*, \tag{16}$$

where f^* is the optimal solution of (1).

3.3 Lower Bounds for Arbitrary Constrained Polynomial Optimization Problems

The relaxation methods described so far in this section are designed for ST-polynomials. Recall that, by Definition 3, an ST-polynomial satisfies two conditions: the vertices of the Newton polytope must correspond to monomial square terms and form a simplex. The first condition is a necessary condition for nonnegativity of a polynomial [30]. The second condition, however, may not hold for some nonnegative polynomials, and one can obtain a SONC certificate by applying an extended technique based on SONC certificates.

Consider a polynomial f that satisfies the first condition and violates the second condition. The idea of the extended method [32, 15], is to write f as a sum of ST-polynomials. To this end, the algorithm presented in [32] covers all terms of f that are not monomial squares by monomial square terms by solving a linear program for each non-monomial-square term. Then it splits the coefficients of the terms of f between the new polynomials so that summing these polynomials up yields f. The coefficients can either be split evenly, though this does not guarantee an optimal splitting, or one can solve a modified version of the SONC optimization problem in order to find an optimal coefficient distribution.

4 Polynomial-Bound Constraints

As mentioned in the previous section, a necessary condition for the nonnegativity of a polynomial is that all vertices $V(f)$ are monomial squares, that is, $V(f) \subseteq \text{MoSq}(f)$. Therefore, when working with an arbitrary polynomial, it may be necessary to reformulate the lower bounding problem so that each term that is not a monomial square is covered by even exponents. Moreover, a relaxation that is solved as part of a branch-and-bound algorithm must be able to utilize tighter variable bounds in the nodes of the tree in order to obtain lower bounds of improving quality.

We assume that each variable has finite bounds. However, directly adding a term corresponding to a variable bound, that is, $(x_i - l_i)$ or $(u_i - x_i)$, to the Lagrangian function does not lead to an improvement in the SONC bound since x_i is not a monomial square, and is therefore treated by the SONC relaxation as a 'negative' term.

In order to circumvent this issue, we derive polynomial constraints on x_i that are valid with respect to the bounds on x_i. The goal is to find a polynomial $h(x_i)$ that is nonnegative on $[l_i, u_i]$ such that, when added to the Lagrangian, $h(x_i)$ adds monomial squares to it, but not new non-monomial-square terms.

Let L' denote the Lagrangian function after the replacement of variable bounds with polynomial-bound constraints. The choice of such a polynomial is not unique and must take the current structure of the exponent set of L into account.

Let A' be a matrix defining exponents introduced in polynomial-bound constraints, where an entry α'_{ji} denotes the exponent for variable j in the ith polynomial-bound constraint, and a column α'_{*i} defines the exponent vector corresponding to the ith polynomial-bound constraint. We propose to use polynomial constraints of the form

$$\mathbf{x}^{\alpha'_{*i}} \leq \max\{|l_i|, |u_i|\}^{\alpha'_{ii}},$$

where $\alpha'_{ji} = 0$ if $j \neq i$, and refer to them as *polynomial-bound constraints.*

The choice of exponents A' will aim at ensuring that all vertices of the Newton polytope $\text{New}(L')$ are monomial squares. In other words, this requires a set of exponents such that these exponents together with the even exponents $\alpha \in V(L)$ provide a valid cover for each $\beta \in \overline{\text{MoSq}}(L)$.

Let $\mathcal{A}' = \{\alpha_{*1}, \dots, \alpha_{*n}\} \subset (2\mathbb{N})^n$ denote the set of new exponents, and let $V(L')$ denote the set of all vertices of the Newton polytope of L after the transformation of variable bounds into polynomial-bound constraints. We require that the vertices $V(L')$ provide a cover for all $\beta \in \overline{\text{MoSq}}(L)$, that is,

$$\forall\ \beta \in \overline{\text{MoSq}}(L)\ \exists\ \lambda^{(\beta)} \in \mathbb{R}_+^{n+1} : \beta = \sum_{\alpha \in V(L')} \lambda_\alpha^{(\beta)} \alpha \text{ and } \sum_{\alpha \in V(L')} \lambda_\alpha^{(\beta)} = 1. \quad (17)$$

The following theorem provides a choice of the set $\mathcal{A}'$ that ensures that the above requirement holds:

Theorem 6. *Let $A' \in \mathbb{N}^{n\times n}$ be a matrix defining the exponents in polynomial-bound constraints, and let $\mathcal{A}' = \{\alpha_{*1}, \dots, \alpha_{*n}\}$, where $\alpha'_{ji} = 0$ if $j \neq i$, and*

$$\alpha'_{ii} \geq (n + (n \mod 2)) \cdot \max_{\beta \in \overline{MoSq}(L)} ||\beta||_\infty.$$

Then $\mathcal{A}' \subset (2\mathbb{N})^n$ and (17) holds, that is, all terms $\beta \in \overline{MoSq}(L)$ are covered by some subset of the exponents $V(L')$.

Proof. The condition $\mathcal{A}' \in (2\mathbb{N})^n$ holds since $(n + (n \mod 2))$ is an even number. We now need to prove the existence of barycentric coordinates $\lambda^{(\beta)}$ satisfying (17).

For each $\beta \in \overline{\text{MoSq}}(L)$, consider the following barycentric coordinates:

$$\begin{aligned}
\lambda_{\alpha'_{i*}}^{(\beta)} &= \frac{\beta_i}{\alpha'_{ii}} && \text{for } i = 1, \dots, n,\\
\lambda_0^{(\beta)} &= 1 - \sum_{\alpha' \in \mathcal{A}'} \lambda_{\alpha'}^{(\beta)},\\
\lambda_\alpha^{(\beta)} &= 0 && \text{for } \alpha \in V(L).
\end{aligned}$$

Then β can be written as

$$\begin{aligned}
\beta = \begin{pmatrix} \lambda_{\alpha'_{1*}}^{(\beta)} \alpha'_{11} \\ \vdots \\ \lambda_{\alpha'_{n*}}^{(\beta)} \alpha'_{nn} \end{pmatrix} &= \lambda_0^{(\beta)} \mathbf{0} + \sum_{\alpha \in V(L)} 0 \cdot \alpha + \sum_{\alpha' \in \mathcal{A}'} \lambda_{\alpha'}^{(\beta)} \alpha' = \\
&= \lambda_0^{(\beta)} \mathbf{0} + \sum_{\alpha \in V(L)} \lambda_\alpha^{(\beta)} \alpha + \sum_{\alpha' \in \mathcal{A}'} \lambda_{\alpha'}^{(\beta)} \alpha'.
\end{aligned}$$

Thus, the condition $\beta = \sum_{\alpha \in V(L')} \lambda_\alpha^{(\beta)} \alpha$ of (17) is fulfilled. It remains to show that $\lambda_\alpha^{(\beta)} \geq 0$ for all $\alpha \in V(L')$, and $\sum_{\alpha \in V(L')} \lambda_\alpha^{(\beta)} = 1$.

The condition $\sum_{\alpha \in V(L')} \lambda_\alpha^{(\beta)} = 1$ and the nonnegativity of all components of λ except for $\lambda_0^{(\beta)}$ follow directly from the definition of λ. To prove that $\lambda_0^{(\beta)} \geq 0$, observe that $\alpha'_{ii} \geq (n + (n \mod 2)) \cdot \max_{\beta \in \overline{\text{MoSq}}(L)} ||\beta||_\infty$ and $\lambda_{\alpha'_{i*}}^{(\beta)} = \dfrac{\beta_i}{\alpha'_{ii}}$, and therefore

$$\lambda_{\alpha'_{i*}}^{(\beta)} \leq \frac{\beta_i}{(n + (n \mod 2)) \cdot \max_{\beta \in \overline{\text{MoSq}}(L)} ||\beta||_\infty} \leq \frac{1}{n + (n \mod 2)}.$$

This implies that

$$\sum_{\alpha' \in \mathcal{A}'} \lambda_{\alpha'}^{(\beta)} \le \frac{n}{n + (n \mod 2)} \le 1$$

$$\Rightarrow \lambda_0^{(\beta)} = 1 - \sum_{\alpha' \in \mathcal{A}'} \lambda_{\alpha'}^{(\beta)} \ge 0.$$

□

The above theorem provides a sufficient condition on the exponents α' in order to guarantee a cover for all inner terms. Since the cost of computing a SONC certificate does not depend on the degree of a polynomial, introducing terms with high degrees is not an issue. The practical choice of exponent values will be further discussed in Section 6.

5 SONC Relaxations in a Branch-and-Bound Algorithm

We have implemented an experimental algorithm that combines an LP-based branch-and-bound algorithm with SONC relaxations. The goal was to assess the potential of integrating the branch-and-bound and SONC-based approaches to constrained polynomial optimization. Our algorithm solves SONC relaxations in some nodes of the branch-and-bound tree and, if they yield a better dual bound than the standard LP relaxations, uses this bound.

5.1 Software: SCIP and POEM

We used the polynomial optimization software POEM in order to solve SONC relaxations. POEM mainly employs SONC nonnegativity certificates, although it also supports SAGE and SOS certificates for unconstrained problems. To solve SONC relaxations, it constructs a geometric problem and applies a logarithmic transformation in order to obtain a convex problem, which it then passes to the solver ECOS [14] via CVXPY [13]. The main focus of POEM is currently on unconstrained optimization, but it also provides functionality for constrained problems.

The general-purpose solver SCIP [5] provided the branch-and-bound algorithm and called POEM in order to solve SONC relaxations. SCIP is a Constraint Integer Programming (CIP) solver, which means that its design allows it to handle any problems where fixing all integer variables results in a linear or nonlinear program. In particular, SCIP can solve problems belonging to two important subclasses of the CIP problem class: mixed-integer linear programs (MILPs) and mixed-integer nonlinear programs (MINLPs). SCIP implements a spatial branch-and-bound algorithm which, by default, solves linear programming (LP) relaxations at each node of the branch-and-bound tree.

SCIP has a plugin-based structure, wherein plugins implementing various components of the solving process such as presolving techniques, primal heuristics, domain propagation techniques, cutting plane separation, and others techniques, are coordinated by the core of the solver. Users can add new plugins.

For the purposes of our implementation, we created a new relaxator plugin. Relaxator plugins enable the solving of custom relaxations instead of default LP relaxations. When called, a relaxator plugin can return a dual bound or report infeasibility of the current node. Additionally, relaxators may provide primal solution candidates, reduce variable domains and add branching candidates and cutting planes.

Further, since polynomial problems are nonlinear problems, the handling of nonlinear constraints in SCIP is relevant to our implementation. Nonlinear expressions in SCIP are represented by expression graphs, where the nodes represent operations and the arcs represent the flows of computation. SCIP constructs an extended formulation of the problem in a lifted space in order to construct LP relaxations. It stores this extended formulation alongside the original formulation of the problem.

5.2 Algorithm and Implementation

We implement SONC relaxations as a relaxator plugin in SCIP via the Python interface PySCIPOpt[1] [22]. The main steps that the relaxator performs are the following:

- it inspects the structure of the problem in order to decide if a SONC relaxation should be applied to it;
- reverts the creation of an objective variable to represent the nonlinear part of the objective function;
- converts the problem into a matrix representation, where each polynomial is given by a vector of term coefficients and a matrix comprised of exponent vectors for each monomial term;
- if needed, adds polynomial-bound constraints;
- passes the problem to POEM, which solves the SONC relaxation and returns either a dual bound or a failure status;
- and reports the results back to SCIP.

The relaxator is called in the root node and in every 10 nodes of the branch-and-bound tree. In the following, we will further explain some steps of the algorithm.

Problem inspection First, the relaxator analyses the expressions in nonlinear constraints in order to determine whether they are polynomials in the original formulation of the problem. The SONC relaxation is only solved when at least half of constraints are polynomial constraints. Linear constraints are not counted as polynomial constraints, but are added to the relaxation in the same way as polynomial constraints are. The relaxator does not use constraints that cannot be represented as polynomial constraints.

[1]Development branch SONCRelax at https://github.com/scipopt/PySCIPOpt

Objective variable removal SCIP handles nonlinear objective functions by reformulating the problem so that to move the nonlinear part of the objective function into a constraint. That is, SCIP applies the following reformulation:

$$\begin{array}{lcl} \min f(\mathbf{x}) + \mathbf{c}^T\mathbf{x} & & \min y + \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \rightarrow & \text{subject to} \\ g(\mathbf{x}) \geq 0, & & y - f(\mathbf{x}) \geq 0, \quad g(\mathbf{x}) \geq 0. \end{array}$$

This, however, introduces a non-square monomial (y) which is not covered by any monomial squares since y does not participate in any other monomial terms. Therefore, the relaxator reverses this reformulation by detecting variables that are present only in the objective and one constraint (which imposes a restriction on how much the variable can decrease), removing such variables and moving the rest of the corresponding constraints to the objective function.

Polynomial-bound constraints The relaxator calls POEM to compute the vertices of the Newton polytope of the Lagrangian function. If the vertices are monomial squares and form a simplex, that is, the polynomial is an ST-polynomial, then the algorithm will preserve the existing structure of the polynomial. The relaxator may still add polynomial-bound constraints $(\max\{|\ell|, |u|\})^{\alpha'} - \mathbf{x}^{\alpha'} \geq 0$, but it uses only those exponents α' that belong to the original exponent set $\mathcal{A}(L)$. Thus, no new monomial terms are introduced, and the Lagrangian function remains an ST-polynomial.

If the Lagrangian function is not an ST-polynomial, then either some of the vertices are non-monomial-squares or the vertices are affinely dependent, or both. If some vertices are non-monomial-squares, then the necessary condition for the nonnegativity of the polynomial is violated, and the addition of polynomial-bound constraints is necessary in order to obtain a finite dual bound by applying a nonnegativity certificate. Adding such constraints, however, does not guarantee that the vertices of the new polynomial form a simplex.

POEM in its current implementation is unable to solve relaxations of constrained optimization problems where the vertices of the Lagrangian function polynomial are affinely dependent. Recall that in the unconstrained case, a non-ST-polynomial is split up into several ST-polynomials, and then dual bounds are computed for each of those polynomials and combined to obtain the dual bound on the entire polynomial. In the constrained case, however, the polynomial to be optimized has variable coefficients that depend on the Lagrangian multipliers μ. If the same variable μ_j is involved in several ST-polynomials obtained by such splitting, then, in general, it will have different values in solutions of the corresponding problems. In order to obtain a valid bound, however, the value of μ_j must be similar for all ST-polynomials. Ensuring this is by itself a challenging task, and is not done in the current implementation.

Therefore, given a polynomial with affinely dependent monomial square vertices of the Newton polytope, we always use the terms obtained from polynomial-bound constraints as the cover for all inner points. These vertices always form a simplex that covers all inner points. Other vertices are disregarded so that to preserve this property. This is a valid relaxation, since monomial square vertices can only increase the dual bound. However, bound quality is worsened when such vertices are discarded.

Thus, the steps for extending the Lagrangian via polynomial-bound constraints are the following:

- If $L(\mathbf{x}, \mu)$ is an ST-polynomial, add only such polynomial-bound constraints that do not introduce new monomials;
- If $L(\mathbf{x}, \mu)$ is not an ST-polynomial, add polynomial-bound constraints with exponents as in Theorem 6 and discard all other vertices of the Newton polytope.

6 Computational Results

In this section, we evaluate the effect of polynomial-bound constraints on bounds yielded by SONC relaxations, in particular the numbers of instances for which a SONC relaxation of the root node yields a finite dual bound. Then we compare three choices of exponents of polynomial-bound constraints, and finally, we assess the impact of SONC relaxations on the performance of a spatial branch-and-bound algorithm.

To this end, we run SCIP with standard SONC relaxations, with SONC relaxations enhanced by polynomial-bound constraints, and default SCIP. We distinguish the following termination statuses of the SONC relaxator:

- *optimal*: the relaxator successfully computed a finite dual bound.
- *infeasible*: infeasibility of the node subproblem was detected.
- *unsolvable*: no valid decomposition of the Lagrangian into circuit polynomials could be computed, or none of the optimization problems (14) and (15) was a GP, or ECOS failed to compute a solution of the GP due to a numerical error.
- *interrupted*: the relaxator was interrupted due to a time limit.
- *did not run*: the relaxator was not called since SCIP trivially solved the instance.

We ran the tests on a development version of SCIP on a cluster of 2.60 GHz Intel Xeon E5-2660 processors with 128 GB memory per node. The time limit was 3600 seconds and the optimality gap tolerance was 0.01%. The relaxation handler was called with a frequency of 10, i.e. the relaxator was executed on subproblems in depth levels that are multiples of 10 of the branch and bound tree. To solve the GP, POEM called the solver ECOS via CVXPY.

We use the same test set that was used by González-Rodríguez et al. [16]. The test set contains 349 instances chosen from two different sources. 180 instances come from the set of polynomial instances randomly generated by Dalkiran and Sherali [12], and 169 polynomial programming instances with continuous variables were chosen from MINLPLib [25].

In all instances, variables are bounded in the original problem formulation. The only exception is the variable that is introduced in order to rewrite a nonlinear objective function as a linear objective function, and is often unbounded. However, the relaxator removes this variable from the formulation when the linear objective is transformed back into nonlinear form, as described in Section 5.

6.1 Finiteness of Bounds

We compare the performance of standard SONC relaxations to the performance of SONC relaxations with added polynomial-bound constraints of the form

$$x_i^{\alpha_i} \leq \max\{|l_i|, |u_i|\}^{\alpha_i},$$

where $\alpha_i = 2 \cdot \max\{\beta_i \mid \beta \in \Delta(L)\} + 4$. The choice of the exponent will be discussed in more detail in the next subsection.

Table 1 compares results produced when using the relaxation enhanced with polynomial-bound constraints with the results produced when using the standard SONC relaxation. It reports the statuses of the relaxator in the root node.

Test Run	*Solution Status of the Relaxation*				
	optimal	*infeasible*	*unsolvable*	*interrupted*	*did not run*
Standard SONC	9	4	329	2	5
SONC + PB	330	0	0	14	5

Table 1: Effect of polynomial-bound (PB) constraints on relaxator status in the root node.

For most instances, the standard SONC relaxation fails to find a lower bound. This is due to the fact that for most instances the Lagrangian function does not have a Newton polytope with even vertices. Adding the polynomial-bound constraints increased the number of instances where a finite lower bound was found from 9 to 330, as the bound constraints always provide a valid cover.

The relaxator, however, no longer detected infeasibility on the 4 instances that the standard SONC relaxation identified as infeasible. This might have been caused either by the change in the constant term resulting from the addition of polynomial-bound constraints, or the relaxator not using the monomial squares that, after the addition of polynomial-bound constraints, lie in the interior of the Newton polytope.

For 12 more instances, the relaxator was interrupted due to a time limit. The reason for this is that with polynomial-bound constraints, it attempted to solve more GPs than the version without the polynomial-bound constraints, since the latter often terminated after establishing that the Newton polytope had non-monomial-square vertices.

6.2 Choice of Exponents in Polynomial-Bound Constraints

This section analyses the effect that the choice of exponent matrix $A' \in \mathbb{N}^{n \times n}$ in polynomial-bound constraints has on performance and dual bound quality. As before, the matrices are diagonal, and we compare the following values of the diagonal entries:

$$\alpha_{ii}^{(n0)} = (n + (n \mod 2)) \cdot \max\{\beta_i \mid \beta \in \Delta(L)\}, \tag{18}$$

$$\alpha_{ii}^{(n4)} = (n + (n \mod 2)) \cdot \max\{\beta_i \mid \beta \in \Delta(L)\} + 4, \tag{19}$$

$$\alpha_{ii}^{(4)} = 2 \cdot \max\{\beta_i \mid \beta \in \Delta(L)\} + 4, \tag{20}$$

for $i = 1, \ldots, n$ and denote the respective matrices by $A^{(n0)}$, $A^{(n4)}$ and $A^{(4)}$.

The matrix $A^{(n0)}$ defines the minimal exponents that satisfy the conditions of Theorem 6, thus guaranteeing a valid cover. However, for some instances, using $A^{(n0)}$ creates degenerate points, that is, inner points that lie on a face of the Newton polytope. A known issue in POEM can cause the relaxator to terminate without successfully finding a dual bound, even though there exists a valid cover. To avoid this issue, we increase the value of the nonzero component by 4, which corresponds to the matrix $A^{(n4)}$. If the exponents defined by $A^{(n4)}$ are vertices of the Newton polytope, then no inner points are degenerate. This is the case for any instance where the Lagrangian function is not an ST-polynomial.

Both with $A^{(n0)}$ and $A^{(n4)}$, the degree increases linearly with the number of variables. Although the complexity of SONC bound computation does not depend on the degree, very high degrees can lead to numerical issues, particularly when variable bounds have large absolute values. This motivates our use of the values $A^{(4)}$. Although this choice does not satisfy the conditions of Theorem 6 and thus does not in general guarantee a valid cover, it is independent of n and leads to a much smaller increase in the degree. For $n = 1, 2$, the matrices $A^{(n4)}$ and $A^{(4)}$ are equivalent, and $A^{(4)}$ guarantees a valid cover for each non-monomial-square term. For $n > 2$, if the inner terms are close to the axes, that is, the monomials are sparse, then $A^{(4)}$ is still likely to provide a valid cover.

Table 2 reports the numbers of instances where the root node SONC relaxation terminated with a given status for a given exponent choice. The exponent matrix $A^{(n0)}$ leads to 13 instances where the SONC relaxation was unsolvable due to degenerate points, or by numerical issues caused by the value $\max\{|l_i|, |u_i|\}^{\alpha_{ii}}$ becoming too large. With $A^{(n4)}$, there are 9 unsolvable instances due to such numerical issues. The use of $A^{(4)}$ results in polynomial-bound constraints of lower degrees and alleviates these numerical issues while still providing a valid cover for all instances in our test set. Further, the use of exponents $A^{(4)}$ results in the smallest number of timeouts, leading to the SONC relaxation producing valid finite dual bounds for the largest number of instances when $A^{(4)}$ is used.

Exponents	Solution Status				
	optimal	*infeasible*	*unsolvable*	*interrupted*	*did not run*
$A^{(n0)}$	308	0	13	23	5
$A^{(n4)}$	314	0	9	21	5
$A^{(4)}$	330	0	0	14	5

Table 2: Comparison of solution statuses for different exponent choices.

There is only a small difference in the root node dual bounds and the run times when using different exponents. For all but three instances, the relative differences between absolute values of the bound are below 1%. On the remaining instances, there is no clear best choice.

Table 3 shows the shifted geometric means of times and the numbers of instances where a given exponent yielded the best time. Given time t for a given exponent and sorted times corresponding to the other two exponents, $t_1 \leq t_2$, the time t is considered to be best if $t/t_1 \leq 1.01$ and $t/t_2 \leq 0.9$.

$A^{(n4)}$ leads to the fastest performance, followed by $A^{(n0)}$ and $A^{(4)}$. However, the difference is small, and since the relaxator yields a valid dual bound for more instances when using $A^{(4)}$, we employ this as the default choice in the rest of our evaluations.

exponent	mean time	number of best
$A^{(n0)}$	122.51	25
$A^{(n4)}$	120.29	26
$A^{(4)}$	123.04	20

Table 3: Comparison of run time with different exponent choices.

6.3 Effect of SONC Relaxations on Branch-and-Bound Performance

Table 4 compares the overall performance of default SCIP with the performance of SCIP with the SONC relaxator. It shows the geometric means of the solving time (shift 1 second) and nodes (shift 100 nodes), and the numbers of instances where the LP relaxation (shown in the row "Off") or the SONC relaxation (shown in the row "On") yielded the better dual bound in the root node. On the remaining 38 instances, there was no difference between the LP and SONC dual bounds.

Overall, SONC relaxations are currently not competitive with the LP relaxations employed by default. The geometric mean of the run time increases drastically when SONC relaxator is enabled, and SONC bounds are stronger than LP bounds on only few instances. The increase in the number of nodes is considerable smaller than the slowdown, which indicates that the high computational cost of SONC relaxations compared to LP relaxations and the lack of warm starting in the SONC relaxator are the main factors leading to the slowdown. Further, the current lack of presolving, reduction and branching methods aimed at improving the bounds yielded by SONC relaxations puts them at a disadvantage when compared to LP relaxations.

Relaxator	Time	Nodes	Bound improvements
Off	25.29	27.04	305
On	123.04	34.60	6

Table 4: Solver performance with and without the relaxator.

Table 5 provides a more detailed analysis for the 6 instances where enabling the SONC relaxator lead to an improvement in the root node dual bound. For two more instances, the SONC relaxator produced a tighter bound than the LP relaxation in non-root nodes, but these instances are not included in the table. For each instance in Table 5, we report the LP and SONC dual bound, the primal bound provided on the MINLPLib website[2], the percentage of absolute gap closed, as well as information on the instance itself such as the highest polynomial degree in the instance and the numbers of variables and constraints, disregarding the objective variable and the objective constraint that are introduced in order to move the nonlinear part of the objective into a constraint.

Despite the small number of such instances, the improvement in the root node dual bound is considerable for each instance. All instances but one are unconstrained optimization problems involving high polynomial degrees, and on these instances SONC relaxations close between 76.1 and 91.8% of the gap. This is most likely due to the SONC approach for unconstrained optimization being more mature than the SONC approach for constrained optimization, as well as the fact that LP relaxations of problems with low-degree polynomials, especially quadratic problems, are very well developed; specialized relaxations such as SONC may be more beneficial for problems with high-degree polynomials.

The improvement on waterund01 is smaller at 11.3%, but the fact that we can at all observe an improvement here, however, is already notable since waterund01 is a quadratic instance, and the benefits of SONC relaxations are more pronounced for problems involving high degrees.

Instance	LP	SONC	Primal	Gap closed	Degree	NVars	NConss
ex4_1_1	-385.308	-97.8766	-7.4873	76.1%	6	1	0
ex4_1_4	-256	-27	0.0	89.5%	4	1	0
ex4_1_6	-3125	-250	7.0	91.8%	6	1	0
ex4_1_7	-166.636	-44.1665	-7.5	77.0%	4	1	0
mathopt5_8	-41.1406	-8.2371	-0.6861	81.3%	6	1	0
waterund01	-307.557	-262.889	86.8333	11.3%	2	40	38

Table 5: Instances with root node dual bound improvements when using SONC relaxations.

7 Conclusion

In this paper, we have developed a method to guarantee that, given finite variable domains, the SONC relaxation returns a finite dual bound. This method enables the SONC relaxation to make use of tighter variable bounds in the branch-and-bound tree. Our experiments showed that for most instances in our test set, the standard SONC relaxation failed to find a finite dual bound, whereas the SONC relaxation with added polynomial-bound constraints found such a bound for all instances except the ones where the relaxator exceeded the time limit. Thus, our approach drastically improves the applicability of SONC relaxations to constrained polynomial optimization problems.

Further, we implemented an experimental algorithm that solves SONC relaxations of node subproblems in a branch-and-bound tree. In its current version, this algorithm is not competitive with the LP-based branch-and-bound methods. This is not surprising since it lacks a full integration of SONC relaxations with

[2]https://www.minlplib.org

the branch-and-bound algorithm, whereas the integration of LP relaxations has been the topic of many years of research. Moreover, the treatment of constraints and variable bounds in SONC relaxations needs further refinement.

However, our experiments showed that our approach has potential, since for some instances, the SONC relaxation yielded dual bounds that closed up to 91.8% of the root node gap when compared to the LP relaxation. Further work would be focused on designing a SONC-based branch-and-bound algorithm where reductions and branching decisions are aimed at strengthening SONC relaxations, and further improving the capabilities of SONC relaxations to make use of bounds and constraints.

Acknowledgments

The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (BMBF grant numbers 05M14ZAM, 05M20ZBM).

Declarations

Competing interests

The authors have no competing interests to declare that are relevant to the content of this article.

Availability of data

Datasets generated and analysed during the current study are available at `https://github.com/scipopt/PySCIPOpt` and `https://www.minlplib.org/`. The authors are in the process of making the remaining data publicly available.

References

[1] Warren P Adams and Hanif D Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986.

[2] Warren P Adams and Hanif D Sherali. Linearization strategies for a class of zero-one mixed integer programming problems. *Operations Research*, 38(2):217–226, 1990.

[3] Amir Ali Ahmadi. *Algebraic relaxations and hardness results in polynomial optimization and Lyapunov analysis*. PhD thesis, Massachusetts Institute of Technology, 2011.

[4] Amir Ali Ahmadi and Anirudha Majumdar. DSOS and SDSOS optimization: more tractable alternatives to sum of squares and semidefinite optimization. *SIAM Journal on Applied Algebra and Geometry*, 3(2):193–230, 2019.

[5] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. ZIB Report 21-41, Zuse Institute Berlin, 2021.

[6] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and engineering*, 8(1):67–127, 2007.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*, volume 9, pages 457–483. Cambridge University Press, Cambridge, 2004.

[8] Venkat Chandrasekaran and Parikshit Shah. Relative entropy relaxations for signomial optimization. *SIAM Journal on Optimization*, 26(2):1147–1173, 2016.

[9] Venkat Chandrasekaran and Parikshit Shah. Relative entropy optimization and its applications. *Mathematical Programming*, 161(1):1–32, 2017.

[10] Clayton W Commander, Panos M Pardalos, Valeriy Ryabchenko, Oleg Shylo, Stan Uryasev, and Grigoriy Zrazhevsky. Jamming communication networks under complete uncertainty. *Optimization Letters*, 2(1):53–70, 2008.

[11] Clayton W Commander, Panos M Pardalos, Valeriy Ryabchenko, Stan Uryasev, and Grigoriy Zrazhevsky. The wireless network jamming problem. *Journal of Combinatorial Optimization*, 14(4):481–498, 2007.

[12] Evrim Dalkiran and Hanif Sherali. Rlt-pos: Reformulation-linearization technique-based optimization software for solving polynomial programming problems. *Mathematical Programming Computation*, 8, 02 2016.

[13] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[14] Alexander Domahidi, Eric Chu, and Stephen Boyd. ECOS: An SOCP solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

[15] Mareike Dressler, Sadik Iliman, and Timo de Wolff. An approach to constrained polynomial optimization via nonnegative circuit polynomials and geometric programming, 2016.

[16] Brais González-Rodríguez, Joaquín Ossorio-Castillo, Julio González-Díaz, Ángel M González-Rueda, David R Penas, and Diego Rodríguez-Martínez. Computational advances in polynomial optimization: RAPOSa, a freely available global solver. *Journal of Global Optimization*, pages 1–28, 2022.

[17] Didier Henrion, Jean-Bernard Lasserre, and Johan Löfberg. Gloptipoly 3: moments, optimization and semidefinite programming. *Optimization Methods & Software*, 24(4-5):761–779, 2009.

[18] Reiner Horst and Hoang Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, Berlin, 2013.

[19] Sadik Iliman and Timo de Wolff. Amoebas, nonnegative polynomials and sums of squares supported on circuits. *Res. Math. Sci.*, 3:Paper No. 9, 35, 2016.

[20] Sadik Iliman and Timo de Wolff. Lower bounds for polynomials with simplex Newton polytopes based on geometric programming. *SIAM J. Optim.*, 26(2):1128–1146, 2016.

[21] Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

[22] Stephen Maher, Matthias Miltenberger, João Pedro Pedroso, Daniel Rehfeldt, Robert Schwarz, and Felipe Serrano. PySCIPOpt: Mathematical programming in Python with the SCIP Optimization Suite. In *International Congress on Mathematical Software*, pages 301–307. Springer, 2016.

[23] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 4054–4061. IEEE, 2013.

[24] John Mattingley and Stephen Boyd. Real-time convex optimization in signal processing. *IEEE Signal processing magazine*, 27(3):50–61, 2010.

[25] A library of mixed-integer and continuous nonlinear programming instances. https://www.minlplib.org, 2022-10-14.

[26] Riley Murray, Venkat Chandrasekaran, and Adam Wierman. Signomial and polynomial optimization via relative entropy and partial dualization. *Mathematical Programming Computation*, 13(2):257–295, 2021.

[27] Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, pages 405–440. Springer, Boston, 2000.

[28] Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, Pablo Parrilo, Matthew Peet, and Declan Jagt. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*. http://arxiv.org/abs/1310.4716, 2021. Available from https://github.com/oxfordcontrol/SOSTOOLS.

[29] Pablo A Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003.

[30] Bruce Reznick. Extremal PSD forms with few terms. *Duke mathematical journal*, 45(2):363–374, 1978.

[31] Henning Seidler. Improved lower bounds for global polynomial optimisation. *arXiv preprint arXiv:2105.14124*, 2021.

[32] Henning Seidler and Timo de Wolff. An experimental comparison of SONC and SOS certificates for unconstrained optimization. *arXiv preprint arXiv:1808.08431*, 2018.

[33] Henning Seidler and Timo de Wolff. POEM: Effective methods in polynomial optimization, version 0.2.1.0(a). `http://www.iaa.tu-bs.de/AppliedAlgebra/POEM/index.html`, July 2019.

[34] Hanif D Sherali, Evrim Dalkiran, and Leo Liberti. Reduced RLT representations for nonconvex polynomial programming problems. *Journal of Global Optimization*, 52(3):447–469, 2012.

[35] Hanif D Sherali and Cihan H Tuncbilek. New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operations Research Letters*, 21(1):1–9, 1997.

[36] Naum Z Shor. Class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6):731–734, 1987.

[37] Bernd Sturmfels. *Solving systems of polynomial equations*. Number 97. American Mathematical Society, Rhode Island, 2002.

[38] Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.

[39] Yang Zheng and Giovanni Fantuzzi. Sum-of-squares chordal decomposition of polynomial matrix inequalities. *Mathematical Programming*, pages 1–38, 2021.