Journal of Symbolic Computation 114 (2023) 246–266

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Duality of sum of nonnegative circuit polynomials and optimal SONC bounds ☆

Dávid Papp

North Carolina State University, Department of Mathematics, Campus Box 8205, Raleigh, NC 27695-8205, USA

ARTICLE INFO

Article history:
Available online 26 April 2022

MSC:
14Q30
90C23
49M29
90C25

Keywords:
Polynomial optimization
Nonnegativity certificates
Circuit polynomials
Convex optimization
Duality
Power cone

ABSTRACT

Circuit polynomials are polynomials with properties that make it easy to compute sharp and certifiable global lower bounds for them. Consequently, one may use them to find certifiable lower bounds for any polynomial by writing it as a sum of circuit polynomials with known lower bounds. Recent work has shown that sums of nonnegative circuit polynomials (or SONC polynomials for short) can be used to compute global lower bounds (called SONC bounds) for polynomials in this manner very efficiently both in theory and in practice, if the polynomial is bounded from below and its support satisfies a certain nondegeneracy assumption. The quality of the SONC bound depends on the circuits used in the computation but finding the set of circuits that yield the best attainable SONC bound among the astronomical number of candidate circuits is a non-trivial task that has not been addressed so far. We propose an efficient method to compute the optimal SONC lower bound by iteratively identifying the optimal circuits to use in the SONC bounding process. The method is derived from a new proof of the result that every SONC polynomial decomposes into SONC polynomials on the same support. This proof is based on convex programming duality and motivates a column generation approach that is particularly attractive for sparse polynomials of high degree and with many unknowns. The method is implemented and tested on a large set of sparse polynomial optimization problems with up to 40 unknowns, of degree up to 60, and up to 3000 monomials in the support. The

☆ This material is based upon work supported by the National Science Foundation under Grant No. DMS-1719828 and Grant No. DMS-1847865.

E-mail address: dpapp@ncsu.edu.

https://doi.org/10.1016/j.jsc.2022.04.015

results indicate that the method is efficient in practice and requires only a small number of iterations to identify the optimal circuits.

1. Introduction

Polynomial optimization, that is, computing the infimum of a polynomial over a basic closed semialgebraic set is a fundamental computational problem in algebraic geometry with a wide range of applications in areas such as discrete geometry, nonlinear dynamical systems, control, extremal combinatorics, power systems engineering, and statistics, to name a few. It is well-known to be an intractable problem; its difficulty stems from the computational complexity of deciding whether a given polynomial is nonnegative, either over $\mathbb{R}^n$ or over a semialgebraic set given by a list of polynomial inequalities (Dickinson and Gijben, 2014; Blekherman et al., 2013). The same problem, coupled with the additional, even more challenging, task of finding a rigorous certificate of nonnegativity (that is verifiable in polynomial time in exact arithmetic) is also a central question in symbolic computation and automated theorem proving (also known as computer-assisted proofs) (Harrison, 2007; Kaltofen et al., 2008; Magron et al., 2015).

Practically scalable approaches to polynomial optimization rely on tractable approximations of cones of nonnegative polynomials. *Inner* approximations based on easily verifiable sufficient conditions of nonnegativity are particularly desirable, as they can yield certificates of nonnegativity or rigorous lower bounds on the infimum, even if one can only compute approximately optimal (but feasible) numerical solutions to the optimization problems solved in the process of generating rigorous certificates (e.g., in hybrid symbolic-numerical methods). Undoubtedly, the most successful of these approximations to date has been sum-of-squares (SOS) cones. Numerical optimization over SOS cones, using semidefinite programming, began at least in the early 2000s (see (Nesterov, 2000; Parrilo, 2000; Lasserre, 2001), and even the earlier work of Shor (1987)) and have given rise to polynomial optimization software such as GloptiPoly 3 (Henrion and Lasserre, 2003), SOSTOOLS (Prajna et al., 2004), YALMIP (Löfberg, 2004), TSSOS (Wang et al., 2021), and alfonso (Papp and Yıldız, 2021).

More recently, a number of alternatives and extensions to SOS have been proposed to address difficulties often encountered when using SOS techniques for polynomials with either a large number of unknowns or a high degree. Methods exploiting sparsity or symmetry of polynomials have been proposed by many researchers including Kojima et al. (2004), Lasserre (2006), Wang et al. (2021), and Riener et al. (2013). Ghasemi and Marshall (2012) suggest an approach using geometric programming for nonnegativity certification, a more efficient convex optimization approach than semidefinite programming commonly used in SOS optimization. Cones of SONC (sums of nonnegative circuit) polynomials (Iliman and de Wolff, 2016) are another family of subcones of nonnegative polynomials that neither contain SOS cones nor are they contained by them, and thus, in principle have the potential to provide better bounds than SOS while promising to be faster than SOS optimization (Seidler and de Wolff, 2018; Magron and Wang, 2021). The nonnegativity of SONC polynomials is established via the AM/GM inequality; also related to this is the notation of SAGE functions (sums of AM/GM exponentials) (Chandrasekaran and Shah, 2016). Through further approximations of the SONC cone, a linear programming approach has also been proposed in (Dressler et al., 2020) to address the computational challenges that arise with the SONC and SAGE cones. In this work, we focus on SONC polynomials, specifically on the problem of *computing optimal SONC lower bounds* efficiently, without approximations.

Some, but not all, of these approaches can be paired with symbolic computing approaches or implemented in rational arithmetic (or as a hybrid numeric-symbolic method) in order to compute rigorously certifiable rational lower bounds for polynomials. In SOS setting, we point to Magron and Safey El Din (2018), Papp and Yıldız (2019a), and Davis and Papp (2021) for examples of different approaches; exact SONC decompositions are computed, e.g., in Magron et al. (2019).

The three main contributions of this paper are the following. Following a brief review on the necessary background on SONC polynomials in Section 2, we provide a new conic optimization formulation for determining whether a polynomial is SONC in Section 3; this formulation is somewhat smaller and simpler than the relative entropy programming formulation used in previous work on SONC polynomials. Using this formulation and convex programming duality, we prove in Section 4 that every SONC polynomial f can be written as a sum of nonnegative circuit polynomials supported on the support of f and a sum of monomial squares. This was also shown (independently, and under slightly different assumptions) by Wang (2019) and later Murray et al. (2021), using different methods, and by Katthän et al. (2020) in the setting of AG functions. In Section 5 we propose an algorithm, motivated by our proof of this result, to iteratively identify the circuits that appear in the optimal SONC decomposition. An implementation of this approach is discussed in Sections 6 and 7, where we demonstrate that the approach can be used to find the optimal SONC lower bound on sparse polynomials with up to 3000 monomials in minutes. We conclude with a discussion on possible extensions and open questions in Section 8.

2. Preliminaries

Recall the following notation and definitions. For vectors $\mathbf{z}$ and $\boldsymbol{\alpha}$ of dimension n, $\mathbf{z}^{\boldsymbol{\alpha}}$ is a shorthand for the monomial $\prod_{i=1}^{n} z_i^{\alpha_i}$. For an n-variate polynomial f given by $f(\mathbf{z}) = \sum_{\boldsymbol{\alpha}\in\operatorname{supp}(f)} f_{\boldsymbol{\alpha}}\mathbf{z}^{\boldsymbol{\alpha}}$ with $f_{\boldsymbol{\alpha}} \neq 0$, the (finite) set of exponents $\operatorname{supp}(f)$ is called the *support* of f. The *Newton polytope* of f is $\operatorname{New}(f) \stackrel{\text{def}}{=} \operatorname{conv}(\operatorname{supp}(f))$, the closed convex hull of the support. A polynomial is a *monomial square* if it can be written as $c\mathbf{z}^{\boldsymbol{\alpha}}$ with $c > 0$ and $\boldsymbol{\alpha} \in (2\mathbb{N})^n$. We are now ready to define the central objects of this paper.

Definition 1. We say that a polynomial f is a *circuit polynomial* if its support can be written as $\operatorname{supp}(f) = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r, \boldsymbol{\beta}\}$, where the set $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r\}$ is affinely independent and $\boldsymbol{\beta} = \sum_{i=1}^{r} \lambda_i \boldsymbol{\alpha}_i$ with some $\lambda_i > 0$ satisfying $\sum_{i=1}^{r} \lambda_i = 1$. In other words, $\boldsymbol{\beta}$ lies in the convex hull of the $\boldsymbol{\alpha}_i$, and the scalars λ_i are the corresponding barycentric coordinates of $\boldsymbol{\beta}$. The support set of a circuit polynomial is called a *circuit*. The exponent $\boldsymbol{\beta}$ is referred to as the *inner exponent* of the circuit, while the $\boldsymbol{\alpha}_i$ are the *outer exponents*.

Note that the affine independence condition on the exponents implies that the barycentric coordinates λ_i are unique and strictly positive. Given a circuit C, $NC(C)$ denotes the set of nonnegative circuit polynomials supported on C. The vector of barycentric coordinates of the inner exponent is denoted by $\boldsymbol{\lambda}(C)$.

Our starting point is the well-known characterization of nonnegative circuit polynomials (Iliman and de Wolff, 2016):

Proposition 2. *Let f be an n-variate circuit polynomial satisfying $f(\mathbf{z}) = \sum_{i=1}^{r} f_{\boldsymbol{\alpha}_i}\mathbf{z}^{\boldsymbol{\alpha}_i} + f_{\boldsymbol{\beta}}\mathbf{z}^{\boldsymbol{\beta}}$ for some real coefficients $f_{\boldsymbol{\alpha}_i}$ and $f_{\boldsymbol{\beta}}$ and suppose that $\boldsymbol{\beta} = \sum_{i=1}^{r} \lambda_i\boldsymbol{\alpha}_i$ with some $\lambda_i > 0$ satisfying $\sum_{i=1}^{r} \lambda_i = 1$. Then f is nonnegative if and only if $\boldsymbol{\alpha}_i \in (2\mathbb{N})^n$ and $f_{\boldsymbol{\alpha}_i} > 0$ for each i, and at least one of the following two alternatives holds:*

1. *$\boldsymbol{\beta} \in (2\mathbb{N})^n$ and $f_{\boldsymbol{\beta}} \geq 0$, or*
2. *$|f_{\boldsymbol{\beta}}| \leq \prod_{i=1}^{r} \left(\frac{f_{\boldsymbol{\alpha}_i}}{\lambda_i}\right)^{\lambda_i}$.*

It has been shown by Dressler et al. (2017) that the inequality in the second alternative in Proposition 2 is convex in the coefficients of f, moreover, it can be represented using $O(r)$ number of affine and relative entropy cone constraints; see also (Chandrasekaran and Shah, 2016) for more on relative entropy programming. In this work, we use conic constraints involving the generalized power cone and its dual to represent nonnegative circuit polynomials, which has the advantage of requiring only a single cone constraint per circuit. Additionally, power cone constraints are computationally easier to handle than relative entropy constraints.

The *(generalized) power cone* with *signature* $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_r)$ is the convex cone defined as

$$\mathcal{P}_{\boldsymbol{\lambda}} \stackrel{\text{def}}{=} \left\{ (\mathbf{v}, z) \in \mathbb{R}^r_+ \times \mathbb{R} \;\middle|\; |z| \leq \mathbf{v}^{\boldsymbol{\lambda}} \right\}. \tag{1}$$

It can be shown that $\mathcal{P}_{\boldsymbol{\lambda}}$ is a proper (closed, pointed, full-dimensional) convex cone for every $\boldsymbol{\lambda} \in]0,1[^r$, and that its dual cone (with respect to the standard inner product) is the following (Chares, 2009):

$$\mathcal{P}^*_{\boldsymbol{\lambda}} \stackrel{\text{def}}{=} \left\{ (\mathbf{v}, z) \in \mathbb{R}^r_+ \times \mathbb{R} \;\middle|\; |z| \leq \prod_{i=1}^r \left(\frac{v_i}{\lambda_i} \right)^{\lambda_i} \right\}.$$

(This has also been independently shown by Dressler et al. (2021).) This means that the second alternative in Proposition 2 can be written simply as a single cone constraint (and without additional auxiliary variables):

$$|f_{\boldsymbol{\beta}}| \leq \prod_{i=1}^r \left(\frac{f_{\boldsymbol{\alpha}_i}}{\lambda_i} \right)^{\lambda_i} \iff \left((f_{\boldsymbol{\alpha}_1}, \ldots, f_{\boldsymbol{\alpha}_r}), f_{\boldsymbol{\beta}} \right) \in \mathcal{P}^*_{\boldsymbol{\lambda}}. \tag{2}$$

Note that the cone depends on the circuit $C = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r, \boldsymbol{\beta}\}$ only through its signature $\boldsymbol{\lambda}(C)$.

We say that a polynomial is a *sum of nonnegative circuit polynomials*, or *SONC* for short, if it can be written as a sum of monomial squares and nonnegative circuit polynomials. SONC polynomials are obviously nonnegative by definition. Since the nonnegativity of a circuit polynomial can be easily verified using Proposition 2, the nonnegativity of a SONC polynomial can be certified by providing an explicit representation of the polynomial as a sum of monomial squares and nonnegative circuit polynomials. Such a certificate is called a *SONC decomposition*. As long as the number of circuits is sufficiently small, a SONC decomposition can be verified efficiently. From (the conic version of) Carathéodory's theorem (Rockafellar, 1970, Corollary 17.1.2) it is clear that every SONC polynomial f can be written as a sum of at most $|\operatorname{supp}(f)|$ nonnegative circuit polynomials, therefore, a "short" SONC decomposition exists. Additionally, it is straightforward that only circuits supported on the Newton polytope need to be considered. (See the proof of Theorem 5 below.) However, the number of circuits supported on the Newton polytope of a polynomial can be astronomical even for polynomials with a relatively small support set (see also Example 6), and it is not clear which of these circuits will be needed in a SONC decomposition. This motivates the search for algorithms that can identify the relevant circuits and compute short SONC decompositions.

3. SONC decompositions and optimization over power cones

Suppose we are given a polynomial $f(\mathbf{z}) = \sum_{\boldsymbol{\alpha} \in \operatorname{supp}(f)} f_{\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}}$ by its support and its coefficients in the monomial basis, and that we are given a set of circuits $\mathcal{C} = \{C^1, \ldots, C^N\}$. We shall assume, without loss of generality, that $\mathbf{0} \in \operatorname{supp}(f)$ and that $\operatorname{supp}(f) \subseteq \bigcup_{j=1}^N C^j$.

Let $\mathcal{S}(\mathcal{C})$ be the set of polynomials that can be written as a sum of nonnegative circuit polynomials whose support is a circuit belonging to $\mathcal{C}$ and of monomial squares supported on $\operatorname{supp}(f)$. Using Proposition 2 and Equation (2), one may see that deciding whether f belongs to $\mathcal{S}(\mathcal{C})$ amounts to solving a conic optimization problem. We shall give the details of this optimization problem next.

Let V be the vertices of $\operatorname{New}(f)$, and consider the following optimization problem, whose decision variables are the nonnegative coefficients $\boldsymbol{\gamma}$ indexed by V:

$$\begin{aligned} &\underset{\boldsymbol{\gamma} \in \mathbb{R}^V_+}{\text{minimize}} \quad \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} \\ &\text{subject to} \quad (\mathbf{z} \mapsto f(\mathbf{z}) + \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}}) \in \mathcal{S}(\mathcal{C}). \end{aligned} \tag{3}$$

It is immediate that f has a desired SONC decomposition if and only if the optimal objective function value of this problem is 0 and if this infimum is attained.

Making the SONC decomposition of the polynomial in the constraint explicit, problem (3) can also be written as follows:

$$\begin{aligned}
\underset{\boldsymbol{\gamma}, p_1, \ldots, p_N, \boldsymbol{\delta}}{\text{minimize}} \quad & \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} \\
\text{subject to} \quad & f(\mathbf{z}) + \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}} \equiv \underbrace{\sum_{j=1}^{N} p_j(\mathbf{z})}_{\substack{\text{nonnegative} \\ \text{circuit polynomials}}} + \underbrace{\sum_{\boldsymbol{\alpha} \in \operatorname{supp}(f) \cap (2\mathbb{N})^n} \delta_{\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}}}_{\text{monomial squares}} \\
& p_j \in NC(C^j) \qquad j = 1, \ldots, N \\
& \gamma_{\boldsymbol{\alpha}} \geq 0 \qquad \boldsymbol{\alpha} \in V \\
& \delta_{\boldsymbol{\alpha}} \geq 0 \qquad \boldsymbol{\alpha} \in \operatorname{supp}(f) \cap (2\mathbb{N})^n .
\end{aligned} \tag{4}$$

In computation, the polynomials required to be identical (by the first constraint) need to be represented by their coefficients in some basis, reducing the constraint to a system of $|\operatorname{supp}(f)|$ linear equations. It is convenient to use the monomial basis, in which case, by way of Proposition 2 and Eq. (2), the cone constraints $p_j \in NC(C^j)$ can be written as cone constraints involving $\mathcal{P}^*_{\boldsymbol{\lambda}(C^j)}$. The details of this formulation are given next; they are straightforward, but in order to write the formulation out explicitly, we need to introduce some additional notation.

Let us partition $\operatorname{supp}(f)$ into $S_{\text{even}} \overset{\text{def}}{=} \operatorname{supp}(f) \cap (2\mathbb{N})^n$ and $S_{\text{odd}} \overset{\text{def}}{=} \operatorname{supp}(f) \setminus (2\mathbb{N})^n$. Now, $f(\mathbf{z}) + \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}}$ is SONC if and only if there exist nonnegative circuit polynomials $p_1, \ldots, p_N$ supported on $C^1, \ldots, C^N$, respectively and coefficients $\delta_{\boldsymbol{\alpha}} \geq 0$ for each $\boldsymbol{\alpha} \in S_{\text{even}}$ such that $p_1(\mathbf{z}) + \cdots + p_N(\mathbf{z}) + \sum_{\boldsymbol{\alpha} \in S_{\text{even}}} \delta_{\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}} = f(\mathbf{z}) + \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} \mathbf{z}^{\boldsymbol{\alpha}}$.

For each $j \in \{1, \ldots, N\}$, let $\mathbf{A}^j \in \{0, 1\}^{\operatorname{supp}(f) \times C^j}$ be the matrix whose rows and columns are indexed by the support of f and the circuit C^j respectively, and whose $(\boldsymbol{\alpha}, \boldsymbol{\alpha})$-th element is 1 for every $\boldsymbol{\alpha} \in C^j (\subseteq \operatorname{supp}(f))$. All other elements of $\mathbf{A}^j$ are 0. In what follows, $\mathbf{A}^j_{\boldsymbol{\alpha},\cdot}$ denotes the row of $\mathbf{A}^j$ indexed by the exponent vector $\boldsymbol{\alpha}$. Noting that $V \subseteq S_{\text{even}}$, we can now write the optimization problem (4) in the monomial basis as follows:

$$\begin{aligned}
\underset{\boldsymbol{\gamma}, \mathbf{x}_1, \ldots, \mathbf{x}_N}{\text{minimize}} \quad & \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} \\
\text{subject to} \quad & \sum_{j=1}^{N} \mathbf{A}^j_{\boldsymbol{\alpha},\cdot} \mathbf{x}_j - \gamma_{\boldsymbol{\alpha}} \leq f_{\boldsymbol{\alpha}} \qquad \boldsymbol{\alpha} \in V \\
& \sum_{j=1}^{N} \mathbf{A}^j_{\boldsymbol{\alpha},\cdot} \mathbf{x}_j \leq f_{\boldsymbol{\alpha}} \qquad \boldsymbol{\alpha} \in S_{\text{even}} \setminus V \\
& \sum_{j=1}^{N} \mathbf{A}^j_{\boldsymbol{\alpha},\cdot} \mathbf{x}_j = f_{\boldsymbol{\alpha}} \qquad \boldsymbol{\alpha} \in S_{\text{odd}} \\
& \gamma_{\boldsymbol{\alpha}} \geq 0 \qquad \boldsymbol{\alpha} \in V , \\
& \mathbf{x}_j \in \mathcal{P}^*_{\boldsymbol{\lambda}(C^j)} \quad j = 1, \ldots, N .
\end{aligned} \tag{5}$$

To see this, note that the decision variable $\mathbf{x}_j \in \mathbb{R}^{C^j}$ $(j = 1, \ldots, N)$ can be interpreted as the coefficient vector of the nonnegative circuit polynomial p_j supported on C^j, $\mathbf{A}^j_{\boldsymbol{\alpha},\cdot} \mathbf{x}_j$ is the coefficient of $\mathbf{z}^{\boldsymbol{\alpha}}$ in $p_j(\mathbf{z})$, and the interpretation of the linear constraints is that $\sum_{j=1}^{n} p_j(\cdot) = f(\cdot) + \sum_{\boldsymbol{\alpha} \in V} \gamma_{\boldsymbol{\alpha}} (\cdot)^{\boldsymbol{\alpha}} - \sum_{\boldsymbol{\alpha} \in S_{\text{even}}} \delta_{\boldsymbol{\alpha}} (\cdot)^{\boldsymbol{\alpha}}$ for some nonnegative coefficients $\delta_{\boldsymbol{\alpha}}$ $(\boldsymbol{\alpha} \in S_{\text{even}})$ whose values are the slacks (differences between the left-hand side and right-hand side values) of the first two sets of inequality constraints.

In the dual problem of (5), the components of the vector of decision variables $\mathbf{y}$ may be indexed by monomials in $\operatorname{supp}(f) = V \cup (S_{\text{even}} \setminus V) \cup S_{\text{odd}}$, and the dual optimization problem can be written as follows:

$$\begin{aligned} \underset{\mathbf{y}\in\mathbb{R}^{\operatorname{supp}(f)}}{\text{maximize}} \quad & \mathbf{f}^{\mathrm{T}}\mathbf{y} \\ \text{subject to} \quad & -(\mathbf{A}^j)^{\mathrm{T}}\mathbf{y} \in \mathcal{P}_{\boldsymbol{\lambda}(C^j)} \qquad j = 1, \ldots, N \\ & 1 + y_{\boldsymbol{\alpha}} \geq 0 \qquad \boldsymbol{\alpha} \in V \\ & y_{\boldsymbol{\alpha}} \leq 0 \qquad \boldsymbol{\alpha} \in S_{\text{even}}. \end{aligned} \tag{6}$$

The constraints in (6) can be further simplified. Recalling the definition of $\mathbf{A}^j$, we have that $-(\mathbf{A}^j)^{\mathrm{T}}\mathbf{y} = (-y_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}\in C^j}$. It is also convenient to replace in notation $\mathbf{y}$ with $-\mathbf{y}$ throughout. This leads to the following representation of the dual of (3):

$$\begin{aligned} \underset{\mathbf{y}\in\mathbb{R}^{\operatorname{supp}(f)}}{\text{maximize}} \quad & -\mathbf{f}^{\mathrm{T}}\mathbf{y} \\ \text{subject to} \quad & (y_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}\in C^j} \in \mathcal{P}_{\boldsymbol{\lambda}(C^j)} \quad j = 1, \ldots, N \\ & y_{\boldsymbol{\alpha}} \geq 0 \qquad \boldsymbol{\alpha} \in S_{\text{even}}, \\ & y_{\boldsymbol{\alpha}} \leq 1 \qquad \boldsymbol{\alpha} \in V. \end{aligned} \tag{7}$$

We are now ready to show that all these problems have attained optimal values, and that strong duality holds for the optimization problems in Eq. (3) and Eq. (7).

Lemma 3. *Suppose that $V \subseteq (2\mathbb{N})^n$ and that for every $\boldsymbol{\alpha}_j \in \operatorname{supp}(f) \setminus V$ there is a circuit $C \in \mathcal{C}$ whose inner monomial is $\boldsymbol{\alpha}_j$ and whose outer monomials are all members of V. Then the optimization problem (4) has a strictly feasible solution as well as an optimal solution. Therefore, both (3) and (7) have optimal solutions, and the optimal objective function values are equal.*

Proof. We can construct a strictly feasible solution to (4) as follows. First, we fix $\delta_{\boldsymbol{\alpha}} = 1$ for each $\boldsymbol{\alpha} \in S_{\text{even}}$. Second, by assumption, for each exponent $\boldsymbol{\alpha}_j \in \operatorname{supp}(f) \setminus V$ we can find a nonnegative circuit polynomial $p_j \in NC(C^j)$ whose inner monomial has the coefficient $f_{\boldsymbol{\alpha}_j}$ (if $\boldsymbol{\alpha}_j \in S_{\text{odd}}$) or $f_{\boldsymbol{\alpha}_j} - 1$ (if $\boldsymbol{\alpha}_j \in S_{\text{even}}$), while its outer monomials have sufficiently large positive coefficients to ensure that p_j is in the interior of the $NC(C^j)$. In the resulting sum $p(\mathbf{z}) \stackrel{\text{def}}{=} \sum_j p_j(\mathbf{z}) + \sum_{\boldsymbol{\alpha}} \delta_{\boldsymbol{\alpha}} z^{\boldsymbol{\alpha}}$, the coefficient of each $\mathbf{z}^{\boldsymbol{\alpha}}$ for $\boldsymbol{\alpha} \in \operatorname{supp}(f) \setminus V$ is equal to $f_{\boldsymbol{\alpha}}$. By further increasing the outer coefficients in each p_j, we can also ensure that for each $\boldsymbol{\alpha} \in V$ the coefficient of each $\mathbf{z}^{\boldsymbol{\alpha}}$ in p is strictly greater then $f_{\boldsymbol{\alpha}}$. The resulting p is a strictly feasible solution; we can set each $\gamma_{\boldsymbol{\alpha}}$ to an appropriate positive value to equate the two sides of the first constraint of (4).

Thus, the minimization problem (4) is feasible; however it cannot be unbounded since the objective function is constrained to be nonnegative on the feasible region. Therefore, its infimum is finite. We show that this finite optimal value is attained using the Weierstrass Extreme Value Theorem. We only need to show that the feasible region can be bounded a priori. First, observe that because each $\gamma_{\boldsymbol{\alpha}}$ is nonnegative and because there is some finite objective function value Γ attained by the strictly feasible solution exhibited above, we can add to the formulation (4) the redundant constraints $\gamma_{\boldsymbol{\alpha}} \in [0, \Gamma]$ for every $\boldsymbol{\alpha} \in V$. Next, since each polynomial p_j and $\delta_{\boldsymbol{\alpha}}\mathbf{z}^{\boldsymbol{\alpha}}$ on the right-hand side of the first constraint of (4) is a nonnegative polynomial, each $\|p_j\|$ and $\|\delta_{\boldsymbol{\alpha}}\|$ can also be bounded from above by $\|f\| + \Gamma \sum_{\boldsymbol{\alpha}\in V} \|\mathbf{z}^{\boldsymbol{\alpha}}\|$. Thus, the feasible set is compact, and the infimum in (4) is attained using the Weierstrass Extreme Value Theorem.

We have shown that (4) has an optimal solution and a Slater point. This implies that (4) and its dual have optimal solutions with the same objective function values, therefore the equivalent problems (3) and its dual (7) also have optimal solutions with the same optimal objective function value. □

The number of decision variables in the explicit conic formulation (5), which can be directly fed to a conic optimization solver, is $|V| + \sum_{j=1}^{N}(r_j + 1)$. This can be prohibitively large for practical computations if the number of circuits N is large. This motivates the rest of the paper, where we narrow down the set of circuits that may be needed in a SONC decomposition and provide an algorithm to iteratively identify the useful circuits.

4. Support of SONC decompositions

Let $f(\mathbf{z}) = \sum_{\boldsymbol{\alpha}\in\mathrm{supp}(f)} f_{\boldsymbol{\alpha}}\mathbf{z}^{\boldsymbol{\alpha}}$ be a SONC polynomial. It is straightforward to argue that in every SONC decomposition of f, every circuit polynomial must be supported on a subset of $\mathrm{New}(f)$; for completeness, we include a short argument in the proof of Theorem 5 below. It is equally natural to ask whether there exists a SONC decomposition for f in which every circuit polynomial is supported on a subset of $\mathrm{supp}(f)$. That this is indeed true was first shown recently independently in (Wang, 2019) and (Murray et al., 2021) using combinatorial and algebraic techniques (and some assumptions on the structure of the support); we shall provide an independent proof using convex programming duality and removing the additional assumption. In the proof, which also motivates the algorithmic approach of the next section, we will need the following simple lemma.

Lemma 4. *Let $\mathbf{c} \in \mathbb{R}^N$ and $d \in \mathbb{R}$ be arbitrary. Furthermore, let $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N$ and $\boldsymbol{\beta}$ be given vectors in $\mathbb{R}^n$, and consider the convex polytope P consisting of all convex combinations of the $\boldsymbol{\alpha}_i$ that yield $\boldsymbol{\beta}$:*

$$P = \left\{ \boldsymbol{\lambda} \in \mathbb{R}^N_+ \;\middle|\; \sum_{i=1}^{N} \lambda_i \boldsymbol{\alpha}_i = \boldsymbol{\beta} \text{ and } \sum_{i=1}^{N} \lambda_i = 1 \right\}.$$

Then, if the inequality

$$\mathbf{c}^{\mathrm{T}}\boldsymbol{\lambda} \leq d \tag{8}$$

holds for every $\boldsymbol{\lambda} \in P$ for which the set $S_{\boldsymbol{\lambda}} \stackrel{\mathrm{def}}{=} \{\boldsymbol{\alpha}_i \mid \lambda_i > 0\}$ is affinely independent, then (8) holds for every $\boldsymbol{\lambda} \in P$.

Proof. This is a reformulation of the statement that every extreme point $\boldsymbol{\lambda}$ of the convex polytope P corresponds to an affinely independent $S_{\boldsymbol{\lambda}}$. This is immediate from the theory of linear optimization: the basic components of every basic feasible solution of the (feasibility) linear optimization problem

$$\begin{aligned} \mathrm{find}_{\boldsymbol{\lambda}} \quad & \sum_{i=1}^{N} \lambda_i \boldsymbol{\alpha}_i = \boldsymbol{\beta} \\ & \sum_{i=1}^{N} \lambda_i = 1 \\ & \lambda_i \geq 0 \qquad i = 1, \ldots, N \end{aligned}$$

correspond to linearly independent ($(n+1)$-dimensional) vectors from $\{\binom{\boldsymbol{\alpha}_1}{1}, \ldots, \binom{\boldsymbol{\alpha}_N}{1}\}$; thus, the nonzero components of every vertex of P correspond to affinely independent $S_{\boldsymbol{\lambda}}$. □

Theorem 5. *Every SONC polynomial f has a SONC decomposition in which every nonnegative circuit polynomial and monomial square are supported on a subset of* $\mathrm{supp}(f)$.

Proof. First, we argue that no monomial outside the Newton polytope $\mathrm{New}(f)$ can appear in any SONC decomposition. Suppose otherwise, then the convex hull of the union of the circuits is a convex polytope that has an extreme point $\boldsymbol{\alpha} \notin \mathrm{New}(f)$. The corresponding monomial $\mathbf{z}^{\boldsymbol{\alpha}}$ has a 0 coefficient in f. At the same time, $\mathbf{z}^{\boldsymbol{\alpha}}$ can only appear in the SONC decomposition as a monomial square or as

an outer monomial in a circuit, but never as an inner monomial. Therefore, its coefficient is 0 only if its coefficient is 0 in every circuit it appears in, which is a contradiction.

Next, consider two instances of problem (3), or equivalently (5): in the first instance, to be called (P_1), we choose the circuits $\mathcal{C} = \{C^1, \dots, C^N\}$ to be the set of all circuits that are subsets of $\mathrm{supp}(f)$, while in the second one, (P_2), we choose the circuits to be the set of all circuits that are subsets of $\mathrm{New}(f) \cap \mathbb{N}^n$. Let the duals of the corresponding problems, written in the form (7), be (D_1) and (D_2). According to the discussion around (3), it suffices to show that (P_1) and (P_2) have the same optimal objective function values, since in that case either both problems have an optimal solution attaining the value 0 (and thus SONC decompositions using both sets of circuits exist) or both problems have a strictly positive optimal value (and thus no SONC decomposition exists using either set of circuits).

Using Lemma 3, (P_1) and (D_1) have optimal solutions $(\mathbf{x}_1^*, \dots, \mathbf{x}_N^*)$ and $\mathbf{y}^*$ attaining equal objective function values. We now use these solutions to construct feasible solutions for both (P_2) and (D_2) that attain the same objective function value.

For (P_2) this is straightforward: in the formulation (5), keep the coefficients $\mathbf{x}_j$ of the circuit polynomials appearing in (P_1) the same value $\mathbf{x}_j^*$, and set $\mathbf{x}_j = \mathbf{0}$ for every new circuit that appears only in (P_2).

For (D_2), we also keep $\mathbf{y}_{\boldsymbol{\alpha}} = \mathbf{y}_{\boldsymbol{\alpha}}^*$ for every $\boldsymbol{\alpha} \in \mathrm{supp}(f)$. With this choice, regardless of the choice of the remaining components of $\mathbf{y}$, every constraint in (D_2) that already appeared in (D_1) is automatically satisfied; moreover, the objective function remains unchanged, since $f_{\boldsymbol{\alpha}} = 0$ for the new variables. Therefore, it only remains to show that $(\mathbf{y}_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha} \in (\mathrm{New}(f) \cap \mathbb{N}^n) \setminus \mathrm{supp}(f)}$ can be chosen in a way that every cone constraint $(y_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha} \in C} \in \mathcal{P}_{\lambda(C)}$ corresponding to a circuit C supported on $\mathrm{New}(f) \cap \mathbb{N}^n$ is satisfied. We show, constructively, a slightly stronger statement: that if we assign values to the new components of $\mathbf{y}$ one-by-one in any order, at each step it is possible to assign a value to the component at hand in a way that satisfies every conic inequality that only involves already processed exponents.

Suppose that some exponents have been given consistent values and let $\hat{\boldsymbol{\alpha}} \in (\mathrm{New}(f) \cap \mathbb{N}^n) \setminus \mathrm{supp}(f)$ be the exponent whose corresponding $y_{\hat{\boldsymbol{\alpha}}}$ needs to be assigned a value next. In every circuit that it appears in, the exponent $\hat{\boldsymbol{\alpha}}$ is either an inner exponent, in which case the cone constraint only provides an upper bound on $|y_{\hat{\boldsymbol{\alpha}}}|$, or an outer exponent, in which case the cone constraint only provides a lower bound on $y_{\hat{\boldsymbol{\alpha}}}$. In particular, if $\hat{\boldsymbol{\alpha}} \notin (2\mathbb{N})^n$, then it cannot be an outer exponent, and $y_{\hat{\boldsymbol{\alpha}}} = 0$ will be a consistent choice. Similarly, if $\hat{\boldsymbol{\alpha}} \in (2\mathbb{N})^n$ but $\hat{\boldsymbol{\alpha}}$ appears only as inner (respectively, outer) exponent in every circuit, then it is easy to find a consistent value for $y_{\hat{\boldsymbol{\alpha}}}$. (Zero, or a sufficiently large positive value, respectively.) The only non-trivial case is when $\hat{\boldsymbol{\alpha}} \in (2\mathbb{N})^n$ and $\hat{\boldsymbol{\alpha}}$ appears both as inner and as outer exponent in a circuit.

Let C_1 be one of the circuits in which $\hat{\boldsymbol{\alpha}}$ is an inner exponent and which gives the lowest upper bound on $y_{\hat{\boldsymbol{\alpha}}}$, and let C_2 be one of the circuits in which $\hat{\boldsymbol{\alpha}}$ is an outer exponent and which gives the greatest lower bound on $y_{\hat{\boldsymbol{\alpha}}}$. We need to show that these bounds are consistent. Let the outer exponents of the circuit C_1 be $\boldsymbol{\alpha}_1, \dots \boldsymbol{\alpha}_r$ and let $(\lambda_i)_{i=1,\dots,r}$ be the barycentric coordinates of $\hat{\boldsymbol{\alpha}}$ in this circuit:

$$\hat{\boldsymbol{\alpha}} = \sum_{i=1}^{r} \lambda_i \boldsymbol{\alpha}_i. \tag{9}$$

Similarly in circuit C_2, let $\boldsymbol{\eta}$ be the inner exponent, let $\hat{\boldsymbol{\alpha}}$ and $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_s$ be the outer exponents, and let $\boldsymbol{\xi}$ denote the barycentric coordinates of $\boldsymbol{\eta}$:

$$\boldsymbol{\eta} = \xi_0 \hat{\boldsymbol{\alpha}} + \sum_{j=1}^{s} \xi_j \boldsymbol{\omega}_j. \tag{10}$$

Then it suffices to show that there exists a $y_{\hat{\boldsymbol{\alpha}}} > 0$ such that

$$\log(y_{\hat{\boldsymbol{\alpha}}}) \leq \sum_{i=1}^{r} \lambda_i \log(y_{\boldsymbol{\alpha}_i}) \tag{11}$$

to satisfy the cone constraint $|y_{\hat{\boldsymbol{\alpha}}}| \leq \prod_{i=1}^{r} y_{\boldsymbol{\alpha}_i}^{\lambda_i}$ from C_1 and

$$\log(|y_{\boldsymbol{\eta}}|) \leq \xi_0 \log(y_{\hat{\boldsymbol{\alpha}}}) + \sum_{j=1}^{s} \xi_j \log(y_{\boldsymbol{\omega}_j}) \tag{12}$$

to satisfy the cone constraint $|y_{\boldsymbol{\eta}}| \leq y_{\hat{\boldsymbol{\alpha}}}^{\xi_0} \prod_{j=1}^{s} y_{\boldsymbol{\omega}_j}^{\xi_j}$ from C_2. The inequalities (11) and (12) are consistent if and only if the lower and upper bounds they give for $\log(y_{\hat{\boldsymbol{\alpha}}})$ are consistent, that is, if

$$\frac{1}{\xi_0}(\log(|y_{\boldsymbol{\eta}}| - \sum_{j=1}^{s} \xi_j \log(y_{\boldsymbol{\omega}_j})) \leq \sum_{i=1}^{r} \lambda_i \log(y_{\boldsymbol{\alpha}_i}),$$

which can be rearranged as

$$\log|y_{\boldsymbol{\eta}}| \leq \sum_{i=1}^{r} \xi_0 \lambda_i \log(y_{\boldsymbol{\alpha}_i}) + \sum_{j=1}^{s} \xi_j \log(y_{\boldsymbol{\omega}_j}). \tag{13}$$

Now, note that from (9) and (10) we also have

$$\boldsymbol{\eta} = \sum_{i=1}^{r} \xi_0 \lambda_i \boldsymbol{\alpha}_i + \sum_{j=1}^{s} \xi_j \boldsymbol{\omega}_j,$$

with coefficients $\xi_0 \lambda_i \geq 0$ and $\xi_j \geq 0$ satisfying $\sum_{i=1}^{r} \xi_0 \lambda_i + \sum_{j=1}^{s} \xi_j = 1$. Thus, (13) is almost identical to a power cone inequality corresponding to a circuit. The only difference is that the "outer exponents" $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r, \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_s\}$ are not necessarily affinely independent, thus these exponents and $\boldsymbol{\eta}$ do not form a circuit. (If they do, we are done, by the inductive assumption that all power cone constraints corresponding to circuits that consists of assigned components of $\mathbf{y}$ are satisfied.)

We can now invoke Lemma 4 with $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r, \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_s\}$ playing the role of $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N$, the exponent vector $\boldsymbol{\eta}$ playing the role of $\boldsymbol{\beta}$, and $(\log(y_{\boldsymbol{\alpha}_1}), \ldots, \log(y_{\boldsymbol{\alpha}_r}), \log(y_{\boldsymbol{\omega}_1}), \ldots, \log(y_{\boldsymbol{\omega}_s}))$ playing the role of $\mathbf{c}$, and $\log|y_{\boldsymbol{\eta}}|$ playing the role of d: if every power cone inequality corresponding to a circuit with inner exponent $\boldsymbol{\eta}$ holds, then (13) also holds. By the argument preceding (13), this implies that $y_{\hat{\boldsymbol{\alpha}}}$ can be assigned a value that is consistent with the values of all already processed component of $\mathbf{y}$. □

5. Optimal SONC bounds and circuit generation

Theorem 5 allows us to dramatically simplify the search for SONC decompositions when the polynomial to decompose is sparse, that is, when $\operatorname{supp}(f)$ is much smaller than $\operatorname{New}(f) \cap \mathbb{N}^n$. That said, even the number of circuits supported on $\operatorname{supp}(f)$ can be exponentially large in the number of variables as the following example shows.

Example 6. Let $\mathbf{e}_i$ denote the ith unit vector and $\mathbf{1} \stackrel{\text{def}}{=} \sum_{i=1}^{n} \mathbf{e}_i$, and let $\operatorname{supp}(f)$ be the set $\{\mathbf{0}, \mathbf{1}, 2n\mathbf{e}_1, \ldots, 2n\mathbf{e}_n, 4n\mathbf{e}_1, \ldots, 4n\mathbf{e}_n\}$. This support set has only $2n+2$ elements, but it supports 2^n different circuits with $\mathbf{1}$ as the inner exponent: independently for each $i = 1, \ldots, n$, we can add either $2n\mathbf{e}_i$ or $4n\mathbf{e}_i$ to the circuit as an outer exponent, in addition to $\mathbf{0}$ (as the last outer exponent) and $\mathbf{1}$ (as the inner exponent).

In this section, we present an iterative method to identify the circuits that are necessary in a SONC decomposition of a given polynomial f. We present the algorithm for the more general and widely applicable problem of finding *the highest SONC lower bound* for a polynomial, which is defined as the negative of the optimal value of the optimization problem

$$\begin{aligned} &\underset{\gamma\in\mathbb{R}}{\text{minimize}} \quad \gamma \\ &\text{subject to} \quad (\mathbf{z}\mapsto f(\mathbf{z})+\gamma)\in\mathcal{S}(\mathcal{C}) \end{aligned} \tag{14}$$

This is a well-defined quantity for every polynomial f that has a SONC decomposition, and by extension for every polynomial that has a SONC bound, as the following Lemma shows.

Lemma 7. *Suppose that f has a SONC decomposition with a given set of circuits $\mathcal{C}$. Then (14) attains a minimum.*

Proof. The proof is essentially the same as the argument used in the last step of the proof of Lemma 3. If f is SONC, then $\gamma = 0$ is a feasible solution to (14). At the same time, the problem cannot be unbounded; indeed, the infimum cannot be lower than $-f(\mathbf{0})$. So the infimum in (14) is finite. Moreover, problem (14) can be equivalently written as

$$\begin{aligned} &\text{minimize} \quad \gamma \\ &\text{subject to} \quad f(\mathbf{z})+\gamma = \sum_{j=1}^{N} p_j(\mathbf{z}) + \sum_{\boldsymbol{\alpha}\in\operatorname{supp}(f)\cap(2\mathbb{N})^n} \delta_{\boldsymbol{\alpha}}\mathbf{z}^{\boldsymbol{\alpha}} \\ &\qquad p_j\in NC(C^j) \quad j=1,\dots,N \\ &\qquad \gamma\in[-f(\mathbf{0}),0] \\ &\qquad \delta_{\boldsymbol{\alpha}}\geq 0 \qquad \boldsymbol{\alpha}\in\operatorname{supp}(f)\cap(2\mathbb{N})^n \end{aligned} \tag{15}$$

Since γ is already bounded, and each of the polynomials p_j and $\delta_{\boldsymbol{\alpha}}\mathbf{z}^{\boldsymbol{\alpha}}$ on the right-hand side of the first constraint is a nonnegative polynomial, any norm of each p_j and $\delta_{\boldsymbol{\alpha}}$ can also be bounded a priori by the same norm of $f+\gamma$, and thus the feasible region of (15) is compact. The claim now follows from the Weierstrass Extreme Value Theorem. □

We now consider the problem of identifying the circuits necessary to obtain the strongest possible SONC lower bound on a polynomial. Consider the optimal solution of (14) for a set of circuits $\mathcal{C} = \{C^1,\dots,C^N\}$ for which this problem attains a minimum. Analogously to (7), the dual of (14) can be written as

$$\begin{aligned} &\underset{\mathbf{y}\in\mathbb{R}^{\operatorname{supp}(f)}}{\text{maximize}} \quad -\mathbf{f}^{\mathrm{T}}\mathbf{y} \\ &\text{subject to} \quad (y_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}\in C^j}\in\mathcal{P}_{\boldsymbol{\lambda}(C^j)} \quad j=1,\dots,N \\ &\qquad y_{\boldsymbol{\alpha}}\geq 0 \quad \boldsymbol{\alpha}\in\operatorname{supp}(f)\cap(2\mathbb{N})^n, \\ &\qquad y_{\mathbf{0}}=1. \end{aligned} \tag{16}$$

Although Eq. (14) does not always have a Slater point, its dual (16) trivially has, therefore, the supremum in (16) equals the attained minimum in (14). Thus, an (approximately) optimal solution to (16) serves as a certificate of (approximate) optimality of the bound given by (14) for the given set of circuits. For brevity, we state this formally without a proof.

Lemma 8. *For every polynomial f and set of circuits $\mathcal{C} = \{C^1,\dots,C^N\}$, the optimization problem (16) has a Slater point. Therefore, if f has a SONC lower bound using the circuits in $\mathcal{C}$, then the optimal value of (16) equals the (attained) optimal values of (14) and (15).*

Applying this Lemma by substituting the set of all circuits supported on $\operatorname{supp}(f)$ for $\mathcal{C}$, we have that if the optimal solution $\mathbf{y}^*$ of (16) satisfies

$$(y^*_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}\in C}\in\mathcal{P}_{\boldsymbol{\lambda}(C)} \tag{17}$$

for every circuit C supported on $\operatorname{supp}(f)$, then the optimal value γ^* of (14) cannot be improved by adding more circuits supported on $\operatorname{supp}(f)$ to the problem. Conversely, if γ^* can be improved by adding *any* circuits, then (17) must be violated by some circuit C supported on $\operatorname{supp}(f)$. Adding any C that violates (17) to the set $\mathcal{C}$ may improve the bound given by (14). Finally, we can repeat the argument of Theorem 5 (with the primal-dual optimization pair (14)-(16) playing the role of (4)-(7)) to show that adding any circuits that are not supported on $\operatorname{supp}(f)$ to $\mathcal{C}$ also cannot improve the bound.

This motivates the iterative algorithm shown in Algorithm 1.

Algorithm 1: SONC bound with iterative circuit generation.

```
   input  : A polynomial f.
   outputs: The optimal SONC lower bound for f and a SONC decomposition certifying the bound.
 1 initialize C = {C^1, ..., C^N}
 2 repeat
 3     solve the primal-dual pair (15)-(16) for the optimal (γ*, p*, δ*) and y*
 4     find the circuit C supported on supp(f) for which (17) is most violated
 5     if no circuit violating (17) exists then
 6         return γ* and the SONC decomposition (p*, δ*) of f + γ*
 7     else
 8         add circuit C found in Step 4 (and possibly other circuits) to C
 9     end if
10 until false
```

We defer the discussion on the initialization step to the end of this subsection and focus on the main loop first, assuming that the initial set of circuits $\mathcal{C}$ has been chosen such that the optimal solutions sought in Line 3 of the first iteration exist.

The most violated constraint in Line 4 can be efficiently computed using the following observation: for a fixed exponent vector $\boldsymbol{\beta}$, finding the circuit corresponding to the most violated constraint among circuits with inner monomial $\mathbf{z}^{\boldsymbol{\beta}}$ amounts to solving the linear optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{\boldsymbol{\alpha}\in\operatorname{supp}(f)\setminus\{\boldsymbol{\beta}\}} \lambda_{\boldsymbol{\alpha}} \log(y_{\boldsymbol{\alpha}}) \\ \text{subject to} \quad & \sum_{\boldsymbol{\alpha}} \lambda_{\boldsymbol{\alpha}} \boldsymbol{\alpha} = \boldsymbol{\beta} \\ & \sum_{\boldsymbol{\alpha}} \lambda_{\boldsymbol{\alpha}} = 1 \\ & \lambda_{\boldsymbol{\alpha}} \geq 0 \qquad \forall \boldsymbol{\alpha} \in \operatorname{supp}(f) \setminus \{\boldsymbol{\beta}\} \end{aligned} \tag{18}$$

Based on Lemma 4, every basic feasible solution $\boldsymbol{\lambda}$ of (18) corresponds to a circuit whose outer monomials are $\{z^{\boldsymbol{\alpha}} \mid \lambda_{\boldsymbol{\alpha}} > 0\}$ and whose inner monomial is $\mathbf{z}^{\boldsymbol{\beta}}$. Recalling the definition of the power cone from Eq. (1), if $\boldsymbol{\lambda}^*$ is an optimal basic feasible solution of (18) and the optimal value is v^*, then the inequality (17) corresponding to $\boldsymbol{\lambda}^*$ (and the circuit C determined by $\boldsymbol{\lambda}^*$) is violated if and only if $\exp(v^*) < |y_{\boldsymbol{\beta}}|$. Solving (18) for each $\boldsymbol{\beta}$, we can either conclude that there are no circuits to add to the formulation or find up to one promising circuit for each $\boldsymbol{\beta}$ to add to the formulation in Line 8. In our implementation we add to $\mathcal{C}$ the circuit corresponding to the most violated inequality for each $\boldsymbol{\beta}$.

Initialization. Problem (3) and the proof of Lemma 3 suggest a strategy for the initialization step of Algorithm 1, which is also entirely analogous to solving linear optimization problems using a two-phase method. We can apply the same circuit generation strategy as above to an instance of problem (3), where $\mathcal{C}$ contains all possible circuits supported on $\operatorname{supp}(f)$. (Additionally, we may replace f by any $f + c$ with an arbitrary constant c.) An initial set of circuits for which this optimization problem is feasible can be easily found: for each exponent $\boldsymbol{\alpha} \in \operatorname{supp}(f)$ that is not a monomial square (that is, for which either $\boldsymbol{\alpha} \notin (2\mathbb{N})^n$ or $f_{\boldsymbol{\alpha}} < 0$ or both), find a circuit whose inner exponent is $\boldsymbol{\alpha}$ and

whose outer exponents are among the vertices V of New(f). This can be done by computing a basic feasible solution of a linear feasibility problem with $|V|$ variables. If for any $\boldsymbol{\alpha}$ such a circuit does not exist, then (3) trivially does not have a feasible solution, and $f + c$ does not have a SONC bound. On the other hand, if the initial set of circuits exists, then (3) can be solved using the same column generation strategy, and we either find that the optimal objective function value of (3) is positive, in which case $f + c$ does not have a SONC bound, or the optimal value is 0. In the latter case we also obtain a feasible solution with a current set of circuits $\mathcal{C}$. This set can be used as the initial set of circuits in Algorithm 1 to find the best SONC bound on f.

Remark 9. There are many polynomials for which Algorithm 1 for the SONC bounding problem (14) can be trivially initialized, without using (3) as a "Phase I" problem as described above. A sufficient condition is the following: suppose that for every $\boldsymbol{\alpha} \in \text{supp}(f)$ for which $\boldsymbol{\alpha} \notin (2\mathbb{N})^n$ or $f_{\boldsymbol{\alpha}} < 0$, the exponent vector $\boldsymbol{\alpha}$ is contained in the interior of a face of New(f) that also contains $\mathbf{0}$. Then for each such $\boldsymbol{\alpha}$ we can find a circuit whose inner exponent is $\boldsymbol{\alpha}$ and for which $\mathbf{0}$ is one of the outer exponents. Taking $\mathcal{C}$ as the set of these circuits, we see that (14) (or equivalently, (15)) is feasible. This is the same condition as the *nondegeneracy* condition of (Seidler and de Wolff, 2018) and (Wang, 2019).

We end this section with a toy example to illustrate the steps of the algorithm.

Example 10. Consider the polynomial f given by

$$f(z_1, z_2) = 1 + z_2^2 - z_1^2 z_2^2 + z_1^2 z_2^6 + z_1^6 z_2^2.$$

This polynomial has a SONC lower bound, since it has only one monomial that is not a monomial square, $-z_1^2 z_2^2$, and the exponent of that monomial is the inner exponent of the circuit $C_1 = \{(0,0), (2,6), (6,2), (2,2)\}$, which contains $\mathbf{0}$ as an outer exponent and has signature $\boldsymbol{\lambda}(C_1) = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$. Thus, for a sufficiently large constant γ, we have $\gamma + z_1^2 z_2^6 + z_1^6 z_2^2 - z_1^2 z_2^2 \geq 0$ for every $\mathbf{z}$, and the remaining terms in f are monomial squares.

Solving the primal-dual pair (14)-(16) with $\mathcal{C} = \{C_1\}$, we obtain the optimal value $\gamma^* = -\frac{7}{8}$, and the SONC decomposition

$$f(z_1, z_2) - \frac{7}{8} = (z_2)^2 + \left(\frac{1}{8} + z_1^2 z_2^6 + z_1^6 z_2^2 - z_1^2 z_2^2\right);$$

the first term on the right-hand side is a monomial square, the second one is a member of $NC(C_1)$. The dual optimal solution (indexing the components in degree lexicographic order) is $\mathbf{y}^* = (1, 0, \frac{1}{4}, \frac{1}{16}, \frac{1}{16})$.

The constraint generation algorithm consists of solving two linear optimization problems: one to find the most promising circuit with z_2^2 as the inner monomial and one to find the most promising circuit with $z_1^2 z_2^2$ as the inner monomial. The remaining three monomials are vertices of the Newton polytope, and need not be considered. The first search is unsuccessful: $y^*_{(0,2)} = 0$, therefore no circuit with $(0,2)$ as an inner exponent can violate its corresponding power cone inequality (17). The second linear optimization problem identifies the circuit $C_2 = \{(0,2), (6,2), (2,2)\}$, with signature $\boldsymbol{\lambda}(C_2) = \left(\frac{2}{3}, \frac{1}{3}\right)$. The corresponding power cone constraint (17) is violated, since $y^*_{(2,2)} = \frac{1}{4} > 0 = y^*_{(0,2)}$.

Solving the primal-dual pair (14)-(16) with $\mathcal{C} = \{C_1, C_2\}$, the optimal value improves to $\gamma^* = -1$, and we obtain the SONC decomposition

$$f(z_1, z_2) - 1 = z_1^2 z_2^6 + \left(z_2^2 + z_1^6 z_2^2 - z_1^2 z_2^2\right);$$

the first term on the right-hand side is a monomial square, the second one is a member of $NC(C_2)$. The circuit C_1 is superfluous. The new optimal dual solution is $\mathbf{y}^* = (1, 0, 0, 0, 0)$. Since every component of $\mathbf{y}^*$ that corresponds to a non-vertex exponent is zero, there cannot be any circuits whose corresponding power cone inequality is violated, proving that we have found the optimal SONC bound.

In this example, we also have $f(z_1, 0) = 1$, proving that 1 is the best possible global lower bound on f, that is, the optimal SONC bound is the global minimum.

6. Implementation

In our implementation we use the open-source Matlab code `alfonso` (Papp and Yıldız, 2021, 2019b), a nonsymmetric cone optimization code that can directly solve the primal-dual pair (14)-(16) using a predictor-corrector approach without any model transformation. (In particular, there is no need to represent the SONC cone or its dual as an affine slice of a Cartesian product of exponential, relative entropy, or second-order cones.) `alfonso` requires only an interior point in the primal cone and a logarithmically homogeneous self-concordant barrier function for the primal cone as input. Since the primal cone is a Cartesian product of nonnegative half-lines and dual cones of generalized power cones, both an easily computable initial point and a suitable barrier function are readily available; see, for example, (Chares, 2009).

Alternatively, we can use (16) as the "primal" problem for `alfonso`. This cone is an intersection of generalized power cones and a nonnegative orthant, so the barrier function is once again readily available, this time as the sum of well-known barrier functions. Furthermore, the Slater point for this problem (recall the discussion around Lemma 8) can be used as an easily computable initial point after scaling to satisfy the only non-homogeneous constraint $y_{\mathbf{0}} = 1$. In our implementation we used the latter variant. When started with a feasible initial solution, `alfonso` maintains feasibility throughout. Therefore, using the dual variant and the dual Slater point as an initial feasible solution, we are guaranteed that are our near-optimal solution to (14)-(16) is dual feasible, and thus the dual optimal value is a lower bound on the minimum even if the other optimality conditions are not satisfied to a high tolerance.

In our first set of experiments (smaller instances with general Newton polytopes) we used the two-phase version of the circuit generation algorithm. In our second set of experiments (larger problems with simplex Newton polytopes) it was easy to find an initial set of circuits, and started with Phase II. In the circuit generation steps, we added every promising circuit identified (up to one circuit for each monomial that is not a vertex of the Newton polytope).

The linear optimization problems used in circuit generation were solved using Matlab's built-in `linprog` function with options that ensure that an optimal basic feasible solution is returned (and not the analytic center of the optimal face).

7. Numerical experiments

The algorithm was tested on two sets of benchmark problems. These can be found in the online repository https://github.com/dpapp-github/crup.

7.1. The Seidler–de Wolff benchmark problems

The first set of instances the algorithm was tested on were problems from the database of unconstrained minimization benchmark problems accompanying the paper (Seidler and de Wolff, 2018). Each instance is a polynomial generated randomly in a way that the polynomial is guaranteed to have a lower bound and a prescribed number of unknowns, degree, and cardinality of support (number of monomials with nonzero coefficients). Since this database is enormous (it has over 30 000 instances), we opted to use only the largest and most difficult instances: the ones with general (not simplex) Newton polytopes and 500 monomials in their support. There are 438 such instances; the number of unknowns n in these instances ranges from 4 to 40, the degree d between 6 and 60. These are indeed very sparse polynomials, the dimensions $\binom{n+d}{d}$ of their corresponding spaces of "dense" polynomials ranges from 8008 to $6 \cdot 10^{25}$.

All experiments were run using Matlab 2017b on a Dell Optiplex 7050 desktop with a 3.6 GHz Intel Core i7 CPU and 32 GB RAM.

Fig. 1 shows the histogram of the total number of circuit generation iterations in Phase I and Phase II combined. The smallest possible value is therefore 2 (in the case when the initial set of

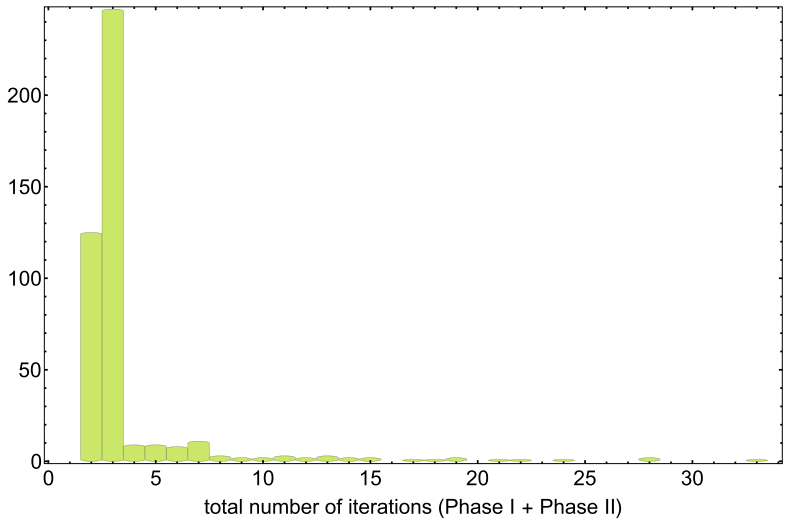

Fig. 1. Histogram of the total number of circuit generation iterations for the largest instances of the Seidler–de Wolff instances. (438 instances; each with 500 monomials, with a varying number of unknowns and degree.) The smallest possible value is 2 (one Phase I iteration and one Phase II iteration). Most instances were solved in two or three iterations.

circuits is optimal). The histogram shows that in the vast majority of these instances no more than 1 additional iteration was needed, that is, all necessary circuits were either among the initial ones, or were identified in the first circuit generation step of Phase II. Correspondingly, the scatterplot in Fig. 2 shows that most of the instances could be solved under a minute, and that the total number of circuits needed to certify the optimal bound was under 1000. (Recall that the initial set of circuits is below $|\mathrm{supp}(f)| = 500$.) It is perhaps interesting to note that even in the "hardest" instance, the algorithm generated fewer than 4500 circuits before the optimal bound was found. This was the only instance where the total running time exceeded one hour; most instances were solved under one minute, and nearly all of them under 5 minutes. There was no discernible pattern indicating what made the difficult instances difficult. In particular, the number of unknowns and the degree alone are not good predictors of the number of circuits or the number of circuit generation iterations.

The optimal solutions or the best known lower bounds are not available in the database. However, upper bounds on the minima of the polynomials can be computed using multi-start local optimization. For simplicity and reproducibility, we used the `NMinimize` function in Mathematica (version 11.3) with default settings to compute approximate minimizers for each of the 438 instances. As the histogram of optimality gaps in Fig. 3 shows, the computed SONC bounds were near-optimal for each instance. This is somewhat surprising, and merits further investigation, as it is in general not guaranteed that a polynomial that is bounded from below has a SONC bound at all; one certainly cannot expect that this bound will always be close to (or equal to) the infimum of the polynomial. Similarly, it cannot be hoped that the local minimum returned by Mathematica is a global minimum. Nevertheless, in each of these instances, the SONC bound was within 1.2% of the global minimum of the polynomial, and with the exception of 46 instances (=10.5%), the relative optimality gap was within 10^{-6}.

7.2. Larger instances

The second set of instances were generated in a somewhat similar fashion as those in the previous set, but the parameters were increased to test the limits of our approach (in particularly, increasing the size of the support above 500). The instances for this experiment were polynomials of degree $d = 8$ with $n = 25$ unknowns. The random supports and coefficients were generated in the following

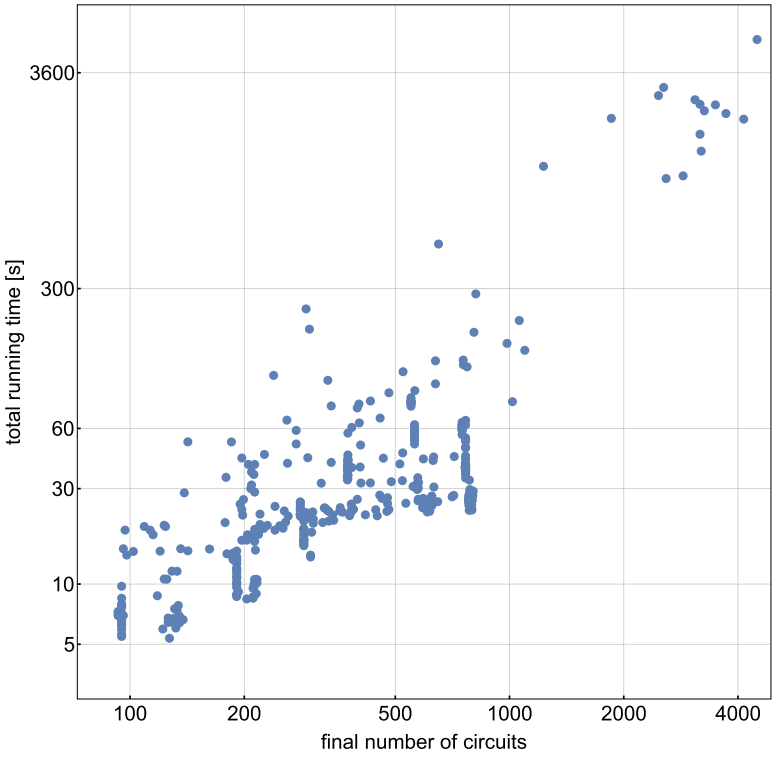

Fig. 2. Scatter plot of the number of circuits in the final iteration of the algorithm and the total running time of the algorithm for the Seidler–de Wolff instances, shown on a logarithmic scale for better visibility. Each dot represents an instance. Since the number of iterations was uniformly small for most instances, the running times and the final number of circuits correlate well. Most instances were solved under a minute, and nearly all of them under 5 minutes. One instance took over an hour to solve.

manner: the constant monomial and the monomials x_i^d were given random integer coefficients between 1 and 5, then a random subset of monomials with componentwise even exponents with total degree less than d were selected (without replacement) and given a random non-zero integer coefficient between -5 and 5. The size of the support was varied in 5% increments up to the maximum of 3301 (the number of componentwise even 25-variate monomials with total degree less than $d = 8$).

Generating the instances in this fashion achieves the following: (1) it is clear a priori that the polynomials can be bounded from below; (2) the Newton polytope $\mathrm{New}(f)$ is known in advance (an $(n+1)$-simplex whose vertices correspond to the monomials 0 and $x_1^d, \ldots, x_n^d$); (3) Phase I can be skipped, and Phase II can be started with an easily computable set of circuits: every exponent in $\mathrm{supp}(f) \setminus V$ is the inner exponent of exactly one initial circuit whose outer exponents are appropriate vertices of the simplex Newton polytope.

Componentwise even monomials were chosen to maximize the number of circuits that can be formed by points in the support and thus make the problems more challenging. (Every exponent of the support other than the vertices of the Newton polytope can be an inner or outer monomial of a number of circuits.) One can also think of the lower bounding of these polynomials over $\mathbb{R}^n$ as problems of bounding polynomials f of degree 4 over the nonnegative orthant by first applying the change of variables $z_i \leftarrow w_i^2$ and then bounding the polynomial $\mathbf{w} \to f(\mathbf{w}^2)$ over $\mathbb{R}^n$.

Each experiment was replicated 10 times (that is, 10 randomly generated instances were solved for each problem size) using the same software and hardware as in the first set of experiments. Fig. 4 shows the distribution of running times for each problem size. The running time increases fairly moderately (approximately cubically) as the number of monomials increases; it remained under

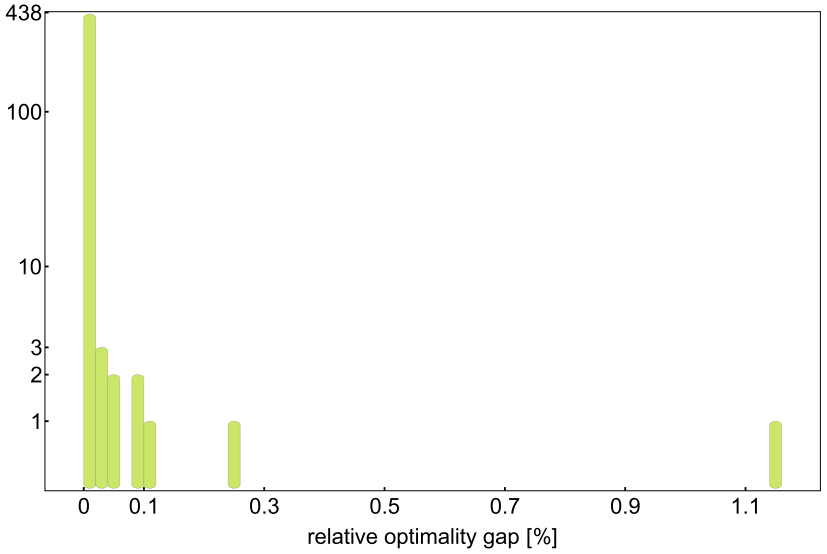

Fig. 3. Histogram of the relative optimality gaps obtained for the Seider–de Wolff instances. Surprisingly, the computed SONC lower bounds were close to the optimal value for each instance. (Note the logarithmic scale on the vertical axis.) The majority of the instances had an optimality gap of 10^{-6} or smaller; too small for the resolution of this picture.

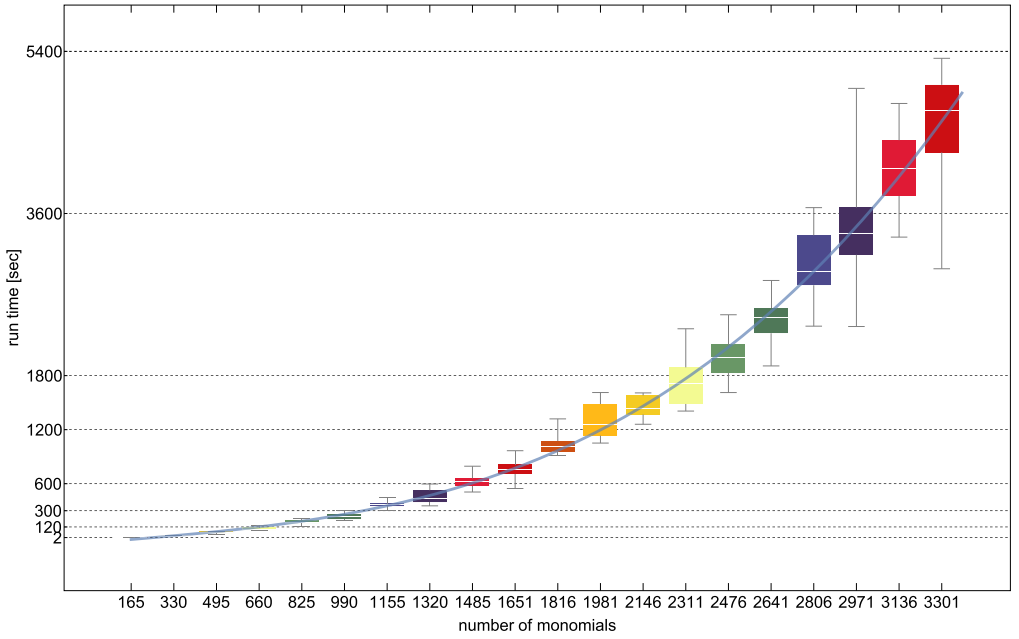

Fig. 4. Box-whisker plot of total running times as a function of problem size from the second experiment. Problem size (horizontal axis) is measured by the number of monomials. Each box represents results from 10 experiments with random polynomials of the same size. A cubic function fitted to the mean values is also shown.

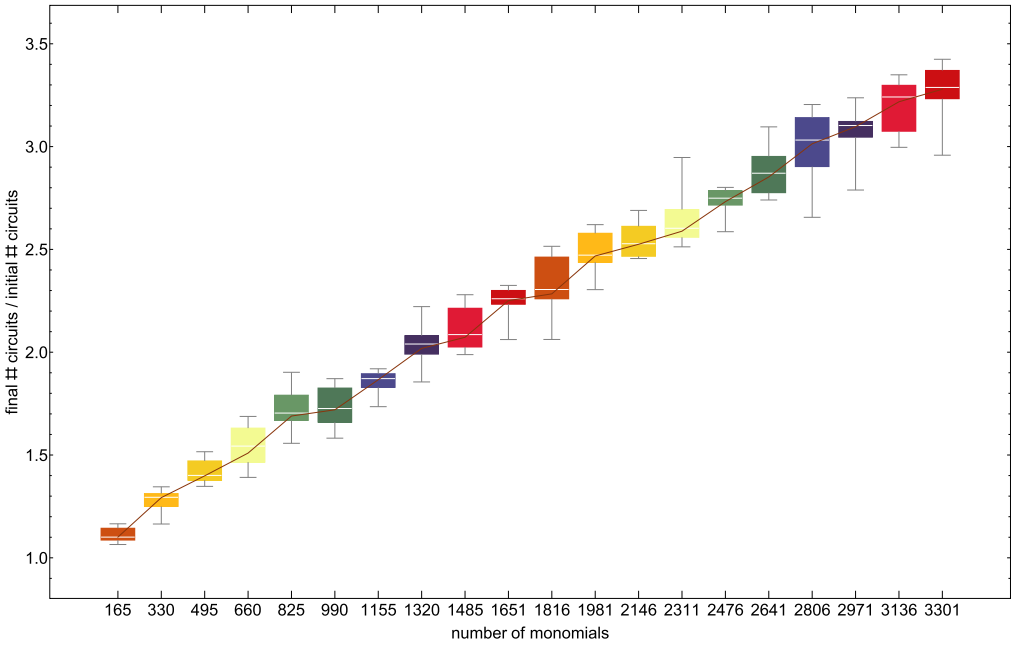

Fig. 5. Box-whisker plot of the ratio between the final number of circuits and the initial number of circuits as a function of problem size from the second experiment. Problem size (horizontal axis) is measured by the number of monomials. Each box represents results from 10 experiments with random polynomials of the same size. The ratio appears to increase only linearly.

1.5 hours for every instance. To see where the increase in running time comes from, in Fig. 5 we plot the ratio between the number of circuits at the end of the circuit generation algorithm and the number of initial circuits, and in Fig. 6 we plot the number of circuit generation iterations. The ratio appears to increase only linearly with the initial number of monomials, showing that the circuit generation algorithm is very effective in choosing the right circuits to add to the formulation out of the exponentially many circuits. (We have no theoretical explanation for this.) Although the number of circuit generation iterations increases with increasing problem sizes (as expected), this increase is very slow (clearly sublinear); most instances were solved in fewer than 8 iterations. Fig. 7 shows the evolution of the number of circuits for each instance as the algorithm progresses. In conclusion, most of the increase in the running time with the increasing problem size is attributable to the increased computational cost per iteration; that is, to the increased amount of time that it takes to solve each instance of the primal-dual pair of optimization problems (14)-(16) in each iteration.

8. Discussion

The computational results confirm that the proposed approach is well-suited for bounding sparse polynomials even when the number of unknowns and the degree are fairly large. Theoretically, the primary driver of the running time is the size of the support, which determines the number of circuits required for an optimal SONC decomposition. The number of circuit generation iterations also appears to depend on the support size, but this dependence was surprisingly mild in all the experiments. (This does not have an apparent theoretical support, but is in line with our experience with column generation approaches in other settings.) Additionally, the dimension of the power cones (and dual power cones) may depend on the number of unknowns, since each circuit may have up $n+1$ outer exponents for polynomials with n unknowns. However, assuming that the support size and the number of unknowns are fixed, the degree of the polynomials does not have an additional impact on the time complexity of the algorithm.

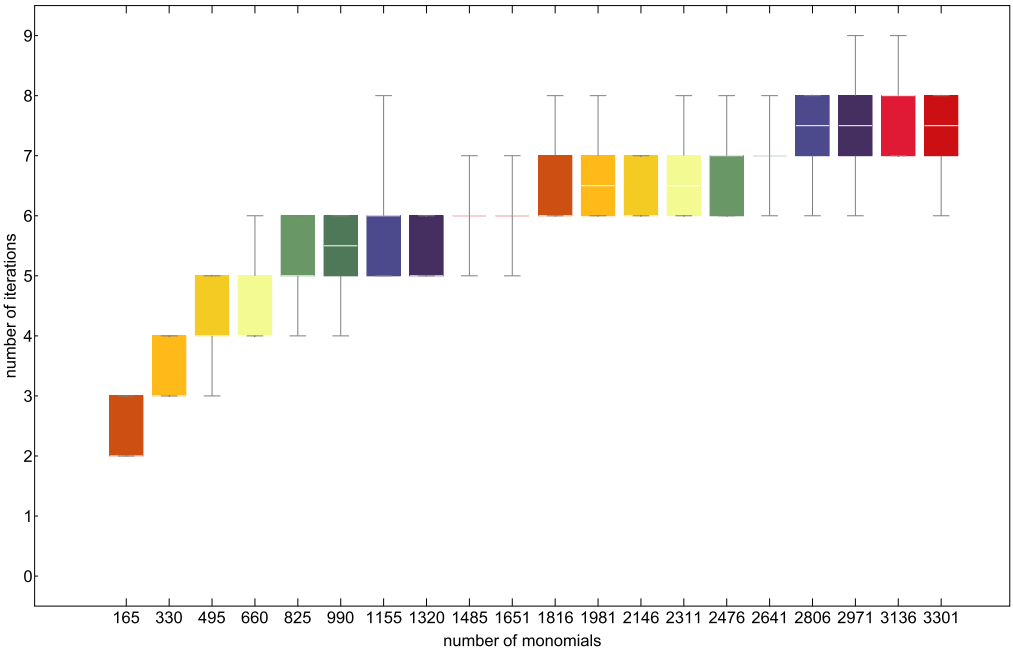

Fig. 6. Box-whisker plot of the number of circuit generation iterations as a function of problem size from the second experiment. Problem size (horizontal axis) is measured by the number of monomials. Each box represents results from 10 experiments with random polynomials of the same size. The ratio appears to increase very slowly (sublinearly).

The second phase of the circuit generation approach finds the optimal SONC bound (and the corresponding circuits and SONC decomposition) once a SONC bound is known to exist from Phase I. The first phase, however, does something slightly weaker than certifying the existence or non-existence of a SONC bound: it finds circuits to prove a target lower bound if possible; in other words, for a given polynomial f and constant c, it can decide whether $f + c$ is SONC or not. If it is, it finds a SONC decomposition of $f + c$, if it is not, it finds a (numerical) certificate of $f + c$ being outside of SONC. It is not clear how one would close this theoretical gap with a purely numerical method: we cannot certify the non-existence of SONC bounds in general, since the set of polynomials with a finite SONC lower bound is not closed. For instance, $f_\varepsilon(\mathbf{z}) \stackrel{\text{def}}{=} (1+\varepsilon)z_1^2 - 2z_1z_2 + z_2^2 - 2z_1$ has a SONC lower bound for every $\varepsilon > 0$ (because $f_\varepsilon + 1/\varepsilon$ is SONC) but f_0 does not have a SONC lower bound (because it is not bounded from below). Practically, this means that we can run the first phase with a "large" value of c and either conclude that a "useful" SONC bound does not exist (because $f + c$ is not SONC) or that f has a SONC lower bound (greater than $-c$); in the latter case Phase II can compute the optimal SONC lower bound.

There are many possible extensions of the algorithm proposed in this paper. The theoretically most straightforward one is to apply the same principle to general optimization problems in which the nonnegativity of an unknown polynomial appears as a constraint. Replacing the nonnegativity constraint with a SONC constraint, this leads to optimization problems similar to the ones we considered, except that every coefficient of the polynomials in question becomes a decision variable (rather than only the constant term being an optimization variable), and the problem may have additional optimization variables. A circuit generation procedure can be derived entirely analogously for problems of this type as long as the additional optimization variables are related to the coefficients of the SONC polynomials through linear constraints.

One may also use this approach to generate circuits for an optimal decomposition of a polynomial into the sum of a SONC polynomial and a sum-of-squares (SOS) polynomial. Theoretically, neither the SOS nor the SONC bound is always better than the other (bivariate counterexamples are easy to

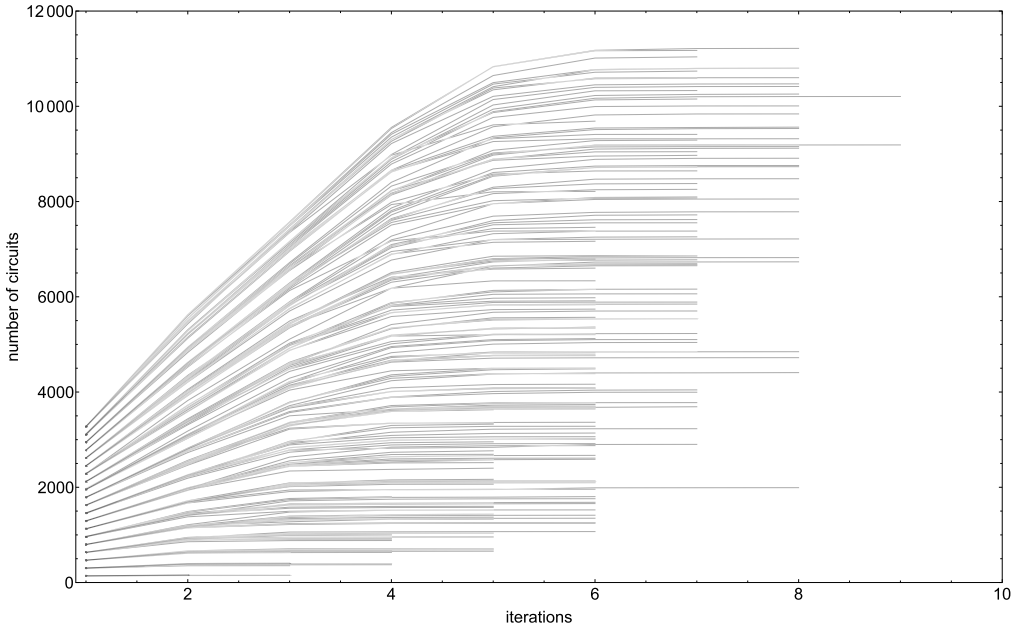

Fig. 7. Diagram showing the number of circuits in each iteration for each instance of the second experiment. Most circuits are added in the first few iterations of the algorithm, in which a new circuit is added for nearly each monomial; later iterations add circuits more selectively. For most instances, several iterations add only a very small number of circuits. The objective function values (not shown) also reveal that in these iterations the bound often does not improve, but the promising circuits need to be added in order to certify the optimality of the bound.

find); a combined SOS+SONC bound would of course be at least as good as either of them. This is not a straightforward computational problem, however, because SOS bounds are typically computed using semidefinite programming algorithms, using software that cannot handle the power cone constraints used in our algorithm. However, the primal-dual algorithm and software used in this paper (`alfonso`) had also been used earlier to efficiently compute SOS bounds for polynomials (Papp and Yıldız, 2019a), implying that the same code could also be used to compute SOS+SONC bounds. The most recent version (version 9) of the commercial conic optimization software Mosek (MOSEK ApS, 2019) also supports the simultaneous use of semidefinite and power cone constraints.

Should the number of circuits generated by the algorithm become prohibitively large, one may consider an improved version of Algorithm 1 which does not only add new promising circuits but also attempts to remove the unnecessary ones in each iteration. This problem did not arise in our experiments (the number of circuits never increased above 10 times the number of circuits used in the optimal SONC decomposition), hence we did not pursue this direction in the paper. We note however that dropping all circuits not used in the last iteration may lead to cycling (the same circuits being added again in the next iteration and than dropped again). An example of a constraint generation algorithm for convex optimization that drops unnecessary cone constraints but safeguards against cycling and could be adapted to our problem is (Mehrotra and Papp, 2014).

Lastly, we leave it for future work to implement an extension of the proposed method to a hybrid symbolic-numerical method that generates rigorous global lower bounds and certificates that can be verified in exact arithmetic from the numerical SONC decompositions computed by our algorithm. Since the numerical method used in our implementation is a primal-dual interior-point approach that computes a *strictly interior* feasible solution $\mathbf{y}^*$ to the problem (16), it is a trivial matter to compute a nearby rational feasible solution $\mathbf{y}_{\text{rat}}$ to the same problem by componentwise rounding the numerical vector $\mathbf{y}^*$ to a close enough rational vector without violating any of the cone constraints. Finally, the problem's only equality constraint can be satisfied exactly by scaling $\mathbf{y}_{\text{rat}}$ (although this does leave

a square root in the final symbolic solution). The resulting dual objective function value $-\mathbf{f}^{\mathrm{T}}\mathbf{y}_{\mathrm{rat}}$ is a rigorous global lower bound on f, close to the numerically obtained bound, whose correctness can be verified in exact arithmetic by verifying the strict feasibility of $\mathbf{y}_{\mathrm{rat}}$. The reconstruction of a primal certificate, that is, a verifiable exact SONC decomposition by computing a rational feasible solution $(p_1, \ldots, p_N)$ of the primal problem (15) from the near-optimal, and only near-feasible, numerical solution is a more complicated matter.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Blekherman, G., Parrilo, P.A., Thomas, R.R. (Eds.), 2013. Semidefinite Optimization and Convex Algebraic Geometry. MOS-SIAM Series on Optimization, vol. 13. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Chandrasekaran, V., Shah, P., 2016. Relative entropy relaxations for signomial optimization. SIAM J. Optim. 26, 1147–1173. https://doi.org/10.1137/140988978.

Chares, R., 2009. Cones and interior-point algorithms for structured convex optimization involving powers and exponentials. Ph.D. thesis. Université Catholique de Louvain.

Davis, M.M., Papp, D., 2021. Dual certificates and efficient rational sum-of-squares decompositions for polynomial optimization over compact sets. arXiv:2105.11369.

Dickinson, P.J.C., Gijben, L., 2014. On the computational complexity of membership problems for the completely positive cone and its dual. Comput. Optim. Appl. 57, 403–415. https://doi.org/10.1007/s10589-013-9594-z.

Dressler, M., Iliman, S., de Wolff, T., 2017. A Positivstellensatz for sums of nonnegative circuit polynomials. SIAM J. Appl. Algebra Geom. 1, 536–555. https://doi.org/10.1137/16M1086303.

Dressler, M., Heuer, J., Naumann, H., de Wolff, T., 2020. Global optimization via the dual sonc cone and linear programming. In: Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation. Association for Computing Machinery, New York, NY, USA, pp. 138–145.

Dressler, M., Naumann, H., Theobald, T., 2021. The dual cone of sums of non-negative circuit polynomials. Adv. Geom. 21, 227–236. https://doi.org/10.1515/advgeom-2020-0019.

Ghasemi, M., Marshall, M., 2012. Lower bounds for polynomials using geometric programming. SIAM J. Optim. 22, 460–473. https://doi.org/10.1137/110836869.

Harrison, J., 2007. Verifying nonlinear real formulas via sums of squares. In: Schneider, K., Brandt, J. (Eds.), Theorem Proving in Higher Order Logics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 102–118.

Henrion, D., Lasserre, J.B., 2003. GloptiPoly: global optimization over polynomials with Matlab and SeDuMi. ACM Trans. Math. Softw. 29, 165–194. https://doi.org/10.1145/779359.779363.

Iliman, S., de Wolff, T., 2016. Amoebas, nonnegative polynomials and sums of squares supported on circuits. Res. Math. Sci. 3, 9. https://doi.org/10.1186/s40687-016-0052-2.

Kaltofen, E., Li, B., Yang, Z., Zhi, L., 2008. Exact certification of global optimality of approximate factorizations via rationalizing sums-of-squares with floating point scalars. In: Proceedings of the Twenty-First International Symposium on Symbolic and Algebraic Computation. ACM, New York, NY, pp. 155–164.

Katthän, L., Naumann, H., Theobald, T., 2020. A unified framework of SAGE and SONC polynomials and its duality theory. Math. Comput. 90, 1297–1322. https://doi.org/10.1090/mcom/3607.

Kojima, M., Kim, S., Waki, H., 2004. Sparsity in sums of squares of polynomials. Math. Program. 103, 45–62. https://doi.org/10.1007/s10107-004-0554-3.

Lasserre, J.B., 2001. Global optimization with polynomials and the problem of moments. SIAM J. Optim. 11, 796–817. https://doi.org/10.1137/S1052623400366802.

Lasserre, J.B., 2006. Convergent SDP-relaxations in polynomial optimization with sparsity. SIAM J. Optim. 17, 822–843. https://doi.org/10.1137/05064504x.

Löfberg, J., 2004. YALMIP: a toolbox for modeling and optimization in MATLAB. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation. Taipei, Taiwan, pp. 284–289. https://yalmip.github.io/.

Magron, V., Safey El Din, M., 2018. On exact Polya and Putinar's representations. In: ISSAC'18: Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation. ACM, New York, NY, USA, pp. 279–286. http://arxiv.org/abs/1802.10339.

Magron, V., Wang, J., 2021. SONC Optimization and Exact Nonnegativity Certificates via Second-Order Cone Programming. Technical report. https://arxiv.org/abs/2012.07903.

Magron, V., Allamigeon, X., Gaubert, S., Werner, B., 2015. Formal proofs for nonlinear optimization. J. Formaliz. Reason. 8, 1–24. https://doi.org/10.6092/ISSN.1972-5787/4319. http://jfr.unibo.it/article/view/4319.

Magron, V., Seidler, H., de Wolff, T., 2019. Exact optimization via sums of nonnegative circuits and arithmetic-geometric-mean-exponentials. In: Proceedings of the ISSAC '19, pp. 291–298.

Mehrotra, S., Papp, D., 2014. A cutting surface algorithm for semi-infinite convex programming with an application to moment robust optimization. SIAM J. Optim. 24, 1670–1697. https://doi.org/10.1137/130925013.

MOSEK ApS, 2019. MOSEK optimization suite release 9.1.5. https://docs.mosek.com/9.1/intro.pdf.
Murray, R., Chandrasekaran, V., Wierman, A., 2021. Newton polytopes and relative entropy optimization. Found. Comput. Math. 21, 1703–1737. https://doi.org/10.1007/s10208-021-09497-w.
Nesterov, Y., 2000. Squared functional systems and optimization problems. In: Frenk, H., Roos, K., Terlaky, T., Zhang, S. (Eds.), High Performance Optimization. In: Applied Optimization, vol. 33. Kluwer Academic Publishers, Dordrecht, pp. 405–440.
Papp, D., Yıldız, S., 2019a. Sum-of-squares optimization without semidefinite programming. SIAM J. Optim. 29, 822–851. https://doi.org/10.1137/17M1160124.
Papp, D., Yıldız, S., 2019b. alfonso: ALgorithm FOr non-symmetric optimization. https://github.com/dpapp-github/alfonso.
Papp, D., Yıldız, S., 2021. alfonso: Matlab package for nonsymmetric conic optimization. INFORMS J. Comput. 34 (1), 11–19. URL: https://arxiv.org/abs/2101.04274.
Parrilo, P.A., 2000. Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization. Ph.D. thesis. California Institute of Technology.
Prajna, S., Papachristodoulou, A., Seiler, P., Parrilo, P.A., 2004. SOSTOOLS: sum of squares optimization toolbox for MATLAB. http://www.cds.caltech.edu/sostools.
Riener, C., Theobald, T., Andrén, L.J., Lasserre, J.B., 2013. Exploiting symmetries in SDP-relaxations for polynomial optimization. Math. Oper. Res. 38, 122–141. https://doi.org/10.1287/moor.1120.0558.
Rockafellar, R.T., 1970. Convex Analysis. Princeton University Press, Princeton, NJ.
Seidler, H., de Wolff, T., 2018. An experimental comparison of SONC and SOS certificates for unconstrained optimization. arXiv preprint arXiv:1808.08431.
Shor, N.Z., 1987. An approach to obtaining global extremums in polynomial mathematical programming problems. Cybernetics 23, 695–700.
Wang, J., 2019. Nonnegative polynomials and circuit polynomials. arXiv preprint arXiv:1804.09455.
Wang, J., Magron, V., Lasserre, J.B., 2021. TSSOS: a moment-SOS hierarchy that exploits term sparsity. SIAM J. Optim. 31, 30–58. https://doi.org/10.1137/19M1307871.