SIAM J. APPL. ALGEBRA GEOMETRY
Vol. 5, No. 2, pp. 388–416

Computing Linear Extensions for Polynomial Posets Subject to Algebraic Constraints*

Shane Kepley†, Konstantin Mischaikow†, and Lun Zhang†

Abstract. In this paper we consider the classical problem of computing linear extensions of a given poset which is well known to be a difficult problem. However, in our setting the elements of the poset are multivariate polynomials, and only a small "admissible" subset of these linear extensions, determined implicitly by the evaluation map, are of interest. This seemingly novel problem arises in the study of global dynamics of gene regulatory networks in which case the poset is a Boolean lattice. We provide an algorithm for solving this problem using linear programming for arbitrary partial orders of linear polynomials. This algorithm exploits this additional algebraic structure inherited from the polynomials to efficiently compute the admissible linear extensions. The biologically relevant problem involves multilinear polynomials, and we provide a construction for embedding it into an instance of the linear problem.

Key words. algebraic geometry, dynamical systems, linear programming, order theory

AMS subject classifications. 14Q30, 37N25, 03C10, 06A07, 06B99

DOI. 10.1137/20M1343208

1. Introduction.

Consider a set of real polynomials $\mathcal{P}$, defined on a domain $\Xi \subset \mathbb{R}^d$, equipped with a partial order $\prec$. We are interested in identifying linear extensions (total orders compatible with $\prec$) that are satisfied by $\mathcal{P}$ under evaluation at a point in Ξ. To be more precise consider a semialgebraic set $\Xi \subset \mathbb{R}^d$, called the *evaluation domain*, and a collection of polynomials $\mathcal{P} := \{p_0, \dots, p_K\} \subset \mathbb{R}[x_1, \dots, x_d]$. Let $\prec$ denote a partial order on $\mathcal{P}$ such that if $p \prec q$, then

$$p(\xi) < q(\xi) \quad \text{for all } \xi \in \Xi. \tag{1}$$

Let S_{K+1} denote the set of permutations on $K+1$ symbols. We identify linear extensions of $\mathcal{P}$ with a subset of S_{K+1} as follows. Given $\sigma \in S_{K+1}$, let $\prec_\sigma$ denote the linear order

$$p_{\sigma(0)} \prec_\sigma p_{\sigma(1)} \prec_\sigma \cdots \prec_\sigma p_{\sigma(K)}.$$

We define the *realizable set* associated to σ by

$$\Xi_\sigma := \left\{ \xi \in \Xi : p_{\sigma(k)}(\xi) < p_{\sigma(k+1)}(\xi) \text{ for all } 0 \le k \le K-1 \right\}. \tag{2}$$

*Received by the editors June 5, 2020; accepted for publication (in revised form) March 16, 2021; published electronically June 28, 2021.
https://doi.org/10.1137/20M1343208
Funding: The authors acknowledge support from NSF DMS-1521771, DMS-1622401, DMS-1839294, the NSF HDR TRIPODS award CCF-1934924, DARPA contracts HR0011-16-2-0033 and FA8750-17-C- 0054, and NIH grant R01 GM126555-01.
†Department of Mathematics, Rutgers University, Piscataway, NJ 08854 USA (sk2011@math.rutgers.edu, mischaik@math.rutgers.edu, lz210@math.rutgers.edu).

Observe that if $\Xi_\sigma \neq \emptyset$, then $\prec_\sigma$ is a linear extension of $\prec$. The *algebraically constrained linear extension problem* (AC-LEP) defined by $(\mathcal{P}, \prec, \Xi)$ is to determine

$$\mathcal{T}(\mathcal{P}, \prec, \Xi) := \{\sigma \in S_{K+1} : \Xi_\sigma \text{ is nonempty}\}.$$

Notice that in the formulation of the AC-LEP we have identified the partial order $\prec_\sigma$ with the associated element in S_{K+1}. We will use this identification throughout the remainder of the paper. We say that each $\sigma \in \mathcal{T}(\mathcal{P}, \prec, \Xi)$ is an *admissible* linear extension.

As is discussed in section 2 our immediate motivation for studying the AC-LEP comes from modeling the dynamics of regulatory networks in biology and in particular characterizing relevant subsets of the parameter space. For the moment we attempt to put this problem into a broader mathematical context, as the problem itself, as well as our solutions for some special cases, has elements of both classical real algebraic geometry and order theory.

Quantifier elimination and real algebraic geometry. Observe that if $\Xi = \mathbb{R}^d$ and $\mathcal{P}$ is an arbitrary collection of polynomials, then $\sigma \in S_{K+1}$ is admissible if and only if there exists $\xi \in \mathbb{R}^d$ such that $p_{\sigma(k)}(\xi) - p_{\sigma(k+1)}(\xi) < 0$ for all $0 \leq k \leq K$. These inequalities define a semialgebraic set. Therefore, if $\prec$ is the trivial partial order (i.e., $\mathcal{P}$ is a single antichain), then this instance of AC-LEP is equivalent to the classical problem of decidability for real semialgebraic sets.

The previous example illustrates a major challenge in solving the AC-LEP. The first general algorithm for solving the quantifier elimination/decidability problem for polynomials in $\mathbb{R}^d$ with feasible running time was the cylindrical algebraic decomposition (CAD) given by Collins [13] in 1975. The CAD algorithm works by subdividing Ξ into subsets on which the polynomials are sign invariant. Given such a decomposition, decidability is reduced to simply evaluating each polynomial at a sample point located in each subset and checking if it satisfies the necessary inequalities. Unfortunately, the computational complexity of the algorithm grows like

$$\mathcal{O}\left((2D)^{2^d-1}(K+1)^{2^d-1}2^{2^{d-1}-1}\right) \quad \text{where} \quad D = \max\{\deg p : p \in \mathcal{P}\}. \tag{3}$$

This worst case running time is known to be sharp even for classes of "nice" polynomials, e.g., linear [10], and moreover, the worst case is also typical [4]. As a result, the question of whether or not $\sigma \in \mathcal{T}(\mathcal{P}, \prec, \Xi)$, even for a single $\sigma \in S_{K+1}$, is often intractable for problems of practical interest.

In addition, the CAD algorithm does not provide partial information. It either runs to completion, in which case it is guaranteed to provide an answer, or it provides no information. Furthermore, we note that if additional algebraic constraints are added, e.g., we assume $\Xi_0 \subset \Xi \subset \mathbb{R}^d$ is a strict semialgebraic subset, then the CAD algorithm can handle this by simply appending the polynomial constraints which define Ξ_0 to the set of polynomials. However, this dramatically increases the complexity of the CAD algorithm, despite the fact that the number of admissible linear extensions can only decrease.

Some improved algorithms have been proposed which aim to reduce the complexity of specific aspects of the problem or for special classes of polynomials (e.g., [34, 8, 9]). These improvements often provide dramatic algorithmic speedups for checking whether $\sigma \in \mathcal{T}(\mathcal{P}, \prec, \Xi)$

for a single linear extension. However, these algorithms have the same worst case running time as the general CAD algorithm, and understanding which classes of polynomials benefit is still a very active area of research. Therefore, these improved algorithms alone are not sufficient to handle instances of AC-LEP since we are interested in determining which of the $(K+1)!$ possible semialgebraic sets are nonempty. An efficient algorithm that does not produce a decomposition of Ξ into sign invariant subsets would still need to be called $(K+1)!$ times. However, we will make use of these improved algorithms as a postprocessing step which we discuss further in section 4.

Computing linear extensions of Boolean lattices. Let us momentarily ignore the algebraic structure in the AC-LEP by forgetting that $\mathcal{P}$ is a collection of polynomials. Hence, we focus only on the poset structure $(\mathcal{P}, \prec)$ and consider the problem of computing all linear extensions. The related problem of counting all linear extensions of a partial order is a well studied problem in order theory. Its importance is due in large part to its connection with the complexity of sorting elements in a list. If one considers a list of $(K+1)$ distinct values which have been partially sorted by making pairwise comparisons on a subset of its elements, then these comparisons induce a partial order. Therefore, the linearly ordered values of the fully sorted list are given by one of the possible linear extensions for the partial order. As a result, the complexity of completely sorting a list is intimately connected to counting linear extensions for posets.

Observe that computing the set of all linear extensions of $(\mathcal{P}, \prec)$ is not easier than counting them which is known to be $\#P$-complete [6]. In particular, a polynomial time algorithm for computing all possible linear extensions for arbitrary posets would imply that $P = NP$ by Toda's theorem [49]. Moreover, we are interested not only in counting linear extensions but in explicitly computing them. Therefore, we are also concerned with how fast the number of admissible linear extensions grows.

For reasons we discuss in section 4, we are specifically interested in the case that $(\mathcal{P}, \prec)$ is a Boolean lattice. Specifically, for fixed $n \in \mathbb{N}$, define $S_n := \{1, \ldots, n\}$, and let 2^{S_n} denote its power set. The *standard* n-dimensional Boolean lattice is the poset, $(2^{S_n}, \prec_B)$, where $\prec_B$ is the partial order defined by inclusion. We say a poset, $(\mathcal{P}, \prec)$, is an n-dimensional Boolean lattice if $(\mathcal{P}, \prec)$ is order isomorphic to the standard n-dimensional Boolean lattice, and we write $\prec_B$ in place of $\prec$.

Estimating the number of linear extensions for Boolean lattices was first considered in [44] which established a nontrivial upper bound. Later, Brightwell and Tetali [7] proved the following result that essentially settles the question for all practical considerations. If $Q(n)$ denotes the number of linear extensions of an n-dimensional Boolean lattice, then

$$\frac{\log Q(n)}{2^n} = \log \binom{n}{\lfloor n/2 \rfloor} - \frac{3}{2}\log e + \mathcal{O}\left(\frac{\ln n}{n}\right). \tag{4}$$

The estimate in (4) illustrates a major challenge in solving the instances of AC-LEP of interest in this paper. Consider an instance of AC-LEP given by $(\mathcal{P}, \prec_B, \Xi)$ where $(\mathcal{P}, \prec_B)$ is an n-dimensional Boolean lattice and Ξ is any evaluation domain. Suppose we had a black box for efficiently computing all linear extensions of a Boolean lattice denoted by $L \subset S_{K+1}$. Furthermore, assume we also had a "CAD"-like algorithm which could efficiently check if

$\Xi_\sigma \neq \emptyset$. Then, one would need to call this algorithm only $\#L$-many times as opposed to $(K+1)!$ as we argued above. However, the growth estimate in (4) implies that the number of calls to this algorithm would still grow exponentially.

This work. In this work we present efficient algorithms for solving two specific instances of the AC-LEP. The first is the *linearly constrained linear extension problem* (LC-LEP), in which $\mathcal{P}$ is a set of linear polynomials, Ξ is the interior of a polyhedral cone (i.e., Ξ is defined by a finite number of linear inequalities), and $\prec$ is an arbitrary partial order. We present an efficient algorithm for solving the LC-LEP in section 3. The second instance of the AC-LEP, which we call the *parameter space decomposition* (PSD) problem and describe now (see Definition 4.6 for a precise definition), is motivated by an application from systems biology described in section 2.

Definition 1.1. *For $n \in \mathbb{N}$, an* interaction function *of order n is a polynomial in n variables, $z = (z_1, \ldots, z_n)$, of the form*

$$f(z) = \prod_{j=1}^{q} f_j(z) \tag{5}$$

where each factor has the form

$$f_j(z) = \sum_{i \in I_j} z_i$$

and the indexing sets $\{I_j : 1 \leq j \leq q\}$ form a partition for $\{1, \ldots, n\}$. We define the interaction type *of f to be $\mathbf{n} := (n_1, \ldots, n_q)$ where n_j denotes the number of elements in I_j.*

Remark 1.2. We leave it to the reader to check that the order of the indexing sets I_j does not matter for any of the analysis in this paper. Therefore, for convenience in reporting results (see section 5) we will always assume that

$$n_1 \geq n_2 \geq \cdots \geq n_q.$$

To define an instance of the PSD problem, fix an interaction function f of order n, and let $\mathcal{P}$ be the collection of polynomials in the $2n$ positive real variables, $\{\ell_i, \delta_i : 1 \leq i \leq n\}$, obtained by evaluating $f(z)$ with each $z_i \in \{\ell_i, \ell_i + \delta_i\}$. Taking all possible combinations of z_i for $1 \leq i \leq n$ produces the polynomials for the PSD problem,

$$\mathcal{P} = \{p_0, \ldots, p_{2^n-1}\} \subset \mathbb{R}[\ell_1, \ldots, \ell_n, \delta_1, \ldots, \delta_n]. \tag{6}$$

In section 4, we will define an indexing map between the 2^n elements of $\mathcal{P}$ and the standard n-dimensional Boolean lattice. Let $\prec_B$ denote the Boolean lattice partial order with respect to this index map, and set $\Xi = (0, \infty)^{2n}$. The *PSD problem* is the instance of the AC-LEP defined by $(\mathcal{P}, \prec_B, \Xi)$. In section 4, we prove that $(\mathcal{P}, \prec_B, \Xi)$ satisfies (1). However, we present some examples before continuing.

Example 1.3. The simplest nonlinear PSD problem arises from the interaction function

$$f(z) = (z_1 + z_2)z_3$$

which has interaction type, $\mathbf{n} = (2, 1)$. The PSD polynomials for this interaction function are given by

$$\begin{aligned} p_0 &= (\ell_1 + \ell_2)\ell_3, & p_4 &= (\ell_1 + \ell_2 + \delta_1)\ell_3, \\ p_1 &= (\ell_1 + \ell_2)(\ell_3 + \delta_3), & p_5 &= (\ell_1 + \ell_2 + \delta_1)(\ell_3 + \delta_3), \\ p_2 &= (\ell_1 + \ell_2 + \delta_2)\ell_3, & p_6 &= (\ell_1 + \ell_2 + \delta_1 + \delta_2)\ell_3, \\ p_3 &= (\ell_1 + \ell_2 + \delta_2)(\ell_3 + \delta_3), & p_7 &= (\ell_1 + \ell_2 + \delta_1 + \delta_2)(\ell_3 + \delta_3). \end{aligned}$$

The PSD evaluation domain is $\Xi = (0, \infty)^6$, and the partial order, $\prec_B$, is imposed on $\mathcal{P}$ by identifying p_i with the vertex of a unit cube whose coordinates are $(i)_2 \in \mathbb{F}_2^3$ where $(i)_2$ is the binary expansion of i. The solution to this PSD problem is the set of admissible linear extensions of $(\mathcal{P}, \prec_B)$ such that $\sigma \in \mathcal{T}\left(\mathcal{P}, \prec_B, (0, \infty)^6\right)$ if and only if $\Xi_\sigma \neq \emptyset$. We note that there are $8! = 40,320$ linear orders on $\mathcal{P}$. However, only 48 of these are linear extensions of $\prec_B$, and of these, only the following 20 linear extensions are admissible:

$$\begin{array}{llll} (0,1,2,3,4,5,6,7) & (0,1,2,4,6,3,5,7) & (0,1,4,5,2,3,6,7) & (0,2,1,4,6,3,5,7) \\ (0,1,2,3,4,6,5,7) & (0,1,4,2,5,3,6,7) & (0,1,4,5,2,6,3,7) & (0,2,4,1,6,3,5,7) \\ (0,1,2,4,3,5,6,7) & (0,1,4,2,5,6,3,7) & (0,2,1,3,4,6,5,7) & (0,2,4,6,1,3,5,7) \\ (0,1,2,4,3,6,5,7) & (0,1,4,2,6,5,3,7) & (0,2,1,4,3,6,5,7) & (0,4,1,2,5,6,3,7) \\ (0,4,2,6,1,5,3,7) & (0,4,1,5,2,6,3,7) & (0,4,2,1,6,5,3,7) & (0,4,1,2,6,5,3,7). \end{array}$$

The 28 "missing" linear extensions are those which do not satisfy certain algebraic constraints which are imposed by the polynomial structure. For example, observe that for fixed $\xi \in (0, \infty)^6$, if $p_3(\xi) < p_6(\xi)$, then $p_1(\xi) < p_4(\xi)$ must also hold.

Unlike the partial order which constrains all possible linear extensions, this order relation is conditional. Indeed, there exist choices of ξ such that $p_3(\xi) > p_6(\xi)$ in which case there is no requirement imposed on the order of $p_1(\xi), p_4(\xi)$, and in fact, there are admissible linear extensions which satisfy both choices, e.g., the first two orders in column four. As another example, observe that $p_5(\xi) < p_6(\xi)$ if and only if $p_1(\xi) < p_2(\xi)$. This algebraic constraint is bi-conditional. However, it too cannot be represented in the partial order since both choices occur in at least one admissible order.

To emphasize the role of the interaction function in determining the algebraic constraints, we consider a similar PSD problem that is also an instance of LC-LEP.

Example 1.4. Consider the interaction type, $\mathbf{n} = (3)$ with corresponding interaction function

$$f = z_1 + z_2 + z_3.$$

As in Example 1.3, we obtain 8 PSD polynomials given explicitly by

$$\begin{aligned} p_0 &= \ell_1 + \ell_2 + \ell_3, & p_4 &= \ell_1 + \ell_2 + \ell_3 + \delta_1, \\ p_1 &= \ell_1 + \ell_2 + \ell_3 + \delta_3, & p_5 &= \ell_1 + \ell_2 + \ell_3 + \delta_1 + \delta_3, \\ p_2 &= \ell_1 + \ell_2 + \ell_3 + \delta_2, & p_6 &= \ell_1 + \ell_2 + \ell_3 + \delta_1 + \delta_2, \\ p_3 &= \ell_1 + \ell_2 + \ell_3 + \delta_2 + \delta_3, & p_7 &= \ell_1 + \ell_2 + \ell_3 + \delta_1 + \delta_2 + \delta_3. \end{aligned}$$

The evaluation domain and partial order are identical to the PSD problem in Example 1.3. Nevertheless, only the following 12 linear extensions are admissible:

$$\begin{array}{llll}(0,1,2,3,4,5,6,7) & (0,1,4,5,2,3,6,7) & (0,2,4,1,6,3,5,7) & (0,4,1,5,2,6,3,7)\\ (0,1,2,4,3,5,6,7) & (0,2,1,3,4,6,5,7) & (0,2,4,6,1,3,5,7) & (0,4,2,1,6,5,3,7)\\ (0,1,4,2,5,3,6,7) & (0,2,1,4,3,6,5,7) & (0,4,1,2,5,6,3,7) & (0,4,2,6,1,5,3,7).\end{array}$$

Similarly, the missing 36 linear extensions in this example fail to satisfy some algebraic constraints. In both cases, the set of admissible linear extensions is a fraction of the set of all linear extensions of the Boolean lattice. In other words, the algebraic structure implies that the admissible linear extensions are a sparse subset of all linear extensions. The algorithm in this paper exploits the algebraic and order theoretic aspects of the PSD problem to overcome the computational complexity limitations which plague both problems in general. Furthermore, we prove that this algorithm finds all possible linear extensions. For both examples we obtained the (12 and 20, respectively) admissible solutions *without* first computing the linear extensions of $(\mathcal{P}, \prec_B)$ and then checking which are admissible.

Related work. As is indicated above our original motivation for this paper arises from problems in systems biology for which explicit complete solutions to the PSD problem are required. As such the majority of this introduction has focused on the question of efficacy of computation. However, there is another direction in which the work of this paper overlaps with other efforts. In particular, observe that the case where the interaction function is linear, i.e., has interaction type $\mathbf{n} = (n)$, solving the AC-LEP is equivalent to identifying all the cells of a hyperplane arrangement. This latter problem has been the subject of considerable study (see [46] for an introduction), and in particular, Maclagan [32] provides the number of solutions for the linear PSD problem for $n = 1, \ldots, 7$. Our computations (see Table 1) lead to the same numbers, as expected.

After accounting for symmetry in the number of linear PSD solutions for interaction types $(1,1,1,1)$, $(1,1,1,1,1)$, and $(1,1,1,1,1,1)$, reported in column two of Table 1, we obtain

$$\frac{336}{4!} = 14 =: a_4, \quad \frac{61920}{5!} = 516 =: a_5, \quad \frac{89414640}{6!} = 124,187 =: a_6$$

which align with sequence A009997 in the On-Line Encyclopedia of Integer Sequences [42]. From [22], we know this sequence represents the number of comparative probability orderings on all subsets of n elements that can arise by assigning a probability distribution to the individual elements. The equivalence of comparative probability orderings and solutions to the linear PSD problem follows directly from the definition of comparative probability.

Organization of paper. The remainder of this paper is organized as follows. In section 2 we briefly describe how the PSD problem arises naturally in the study of global dynamics for gene regulatory networks. In section 3, we present an efficient algorithm for solving instances of the LC-LEP. In section 4, we show that the LC-LEP is related to the PSD problem in the following way. If $\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ is an instance of the PSD problem, then we construct an associated instance of LC-LEP, denoted by $(\mathcal{P}', \prec_B, \Xi')$, which satisfies the inclusion

$$\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subseteq \mathcal{T}\left(\mathcal{P}', \prec_B, \Xi'\right). \tag{7}$$

We refer to this instance of LC-LEP as the *linearized* PSD problem associated to $(\mathcal{P}, \prec_B, (0,\infty)^{2n})$. We exploit this construction and the algorithm for solving the LC-LEP presented in section 3 to provide a means of efficiently computing a collection of candidates that contains the solution to the PSD problem. We prove that in some cases the inclusion in (7) is actually an equality. More generally, this inclusion is strict, but the candidate set is a sparse subset of the collection of all linear extensions of $(\mathcal{P}, \prec_B)$. In this case we describe algorithms for removing the nonadmissible solutions.

Finally, in section 5 we present the results for all PSD solutions with order up to 4. Additionally, we have some results for PSDs of orders 5 and 6. For the remaining cases and PSDs of higher order the computations become too large, i.e., we do not have immediate use for them in applications, and they require significant computing resources.

2. Dynamic Signatures Generated by Regulatory Networks. This section provides a brief description of how the AC-LEP arises in the context of mathematical modeling of problems from systems biology with a few comments at the end indicating its potential application in more general settings of nonlinear dynamics. Biologists often describe regulatory networks in terms of annotated directed graphs, such as that shown in Figure 1(a) where the labeling of the edges, $n \to m$ or $n \dashv m$, indicates whether node n activates or represses node m. Our goal is to describe the type of dynamics that can be expressed by the regulatory network. This requires two things: (i) a mathematical framework in which we can rigorously discuss dynamics when an analytic expression of the nonlinearity is unknown, and more specifically, (ii) imposing a mathematical interpretation on the regulatory network that is compatible with its use as a biological model.

We discuss both of these topics in the following sections but hasten to add that there are other complementary approaches to this problem. Perhaps the strongest dichotomy is between Boolean models and ordinary differential equation (ODE) models (see [1] for a brief overview and references). Recent efforts in the ODE direction seek to go from networks to

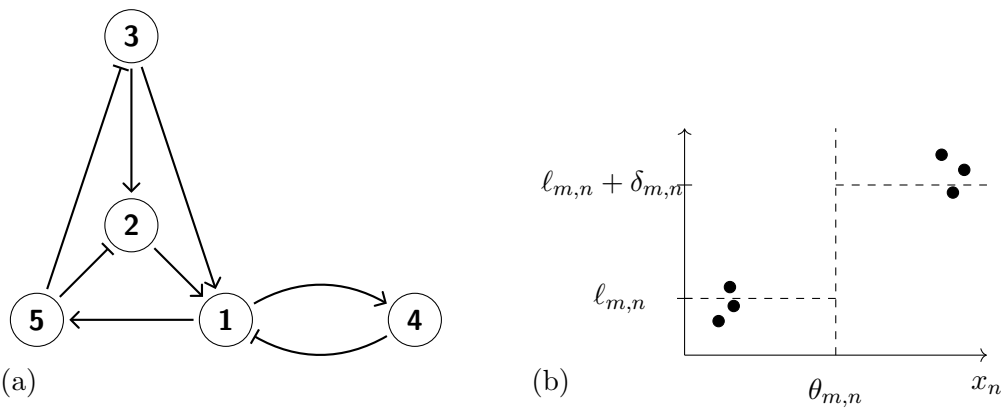

Figure 1. (a) *Example of a regulatory network.* (b) *Model for edge $n \to m$ where $\ell_{m,n}$ indicates low level of growth rate of x_m induced by x_n and $\ell_{m,n} + \delta_{m,n}$ indicates high level of growth rate of x_m induced by x_n. $\theta_{m,n}$ provides information about the value of x_n that lies between low values inducing low and high expression levels.*

explicit polynomial vector fields (see [26] and references therein). The approach we describe lies somewhere between these two. Our approach is partially combinatorial, and thus we obtain efficacy similar to that of the Boolean models, but we maintain continuous parameterization (hence the necessity of the results of this paper), and therefore our computations can be interpreted in the context of ODEs. However, unlike methods based on explicit vector fields, we derive the structure of the dynamics via order theory and algebraic topology. Understanding the relative strengths and weaknesses of these various methods is still an active area of research.

2.1. Mathematical framework. Classical nonlinear dynamics focuses on invariant sets and bifurcations, both of which are extremely sensitive to parameters, e.g., nonlinearities. Unfortunately, this is precisely the information that is unknown in multiscale systems such as those associated with biology. C. Conley proposed that in these situations one should focus on isolated invariant sets [14], as there is a sheaf structure associated to them [41], and thus they are not subject to the above-mentioned sensitivity. Furthermore, he developed an algebraic topological invariant for isolated invariant sets, called the Conley index, from which one can recover considerable information about the existence and structure of the invariant set [40]. In particular, using the homology Conley index one can identify the existence of invariant sets [14], equilibria [45, 35], periodic orbits [37], heteroclinic orbits [15], chaotic dynamics [39, 48], and semiconjugacies onto nontrivial dynamics [38, 36].

One indication of the strength of Conley's proposition is that it is possible to study a differential equation (ordinary or partial) or a continuous map (finite or infinite dimensional) numerically and with appropriate bounds on the errors, to draw rigorous conclusions about the dynamics from homological Conley index computations [39, 18, 27, 19]. The weakness of the above-mentioned results is that they are perturbative in nature, i.e., the results are computed at given parameter values and the results are guaranteed to be true for a sufficiently small neighborhood of that parameter value. In practice one can extract numerical bounds for the size of the neighborhood, but they tend to be small.

For multiscaled systems such as those arising in systems and synthetic biology we need to be able to investigate dynamics over large ranges of parameter values. This was done in the context of low dimensional maps with a low dimensional parameter space [3, 12, 11]. The strategy is to subdivide parameter space uniformly, for each region of parameter space compute dynamics with error bounds valid over the entire region, and once again interpret the structure of the dynamics using the Conley index. While the approach is extremely effective for low dimensional problems it is difficult to extend to higher dimensions for two reasons:

R1 the cost of computing effective numerical error bounds becomes prohibitive, and

R2 the subdivision of parameter space is not chosen according to the underlying dynamics and refinement of the subdivision leads to exponential growth with dimension in the number of subdivisions.

With **R1** in mind W. Kalies, R. Vandervorst, and the third author have developed an approach to dynamics that is based on combinatorics, order theory (posets and lattices), and algebraic topology. The combinatorics arises from the choice of a decomposition of the phase space into closed regular sets, and the dynamics is characterized by a relation on this collection of closed regular sets. It is proven that Conley's fundamental characterization of dynamics

in terms of attractor-repeller pairs or equivalently Morse decompositions can be completely recovered via this combinatorial approach [28, 29, 30]. The algebra of the order theory, in particular Birkhoff's theorem relating finite posets with finite distributive lattices, provides an algorithmic approach for the identification of index pairs from which the Conley index can be computed [31]. This in turn allows one to obtain a chain complex that codifies gradient-like structure of the dynamics and the Conley indices of all the associated isolated invariant sets [25]. Observe that, at least conceptually, we have replaced the numerical computations by combinatorial computations. In practice, at least for the biological regulatory networks discussed below, this combinatorial approach is extremely efficient with respect to both time and memory [17].

Dealing with **R2** is the raison d'être for this paper. As is made clear below for the regulatory networks of interest our closed regular sets take the form of cubes determined by parameter values. The relation on these cubes that represents the dynamics is obtained by determining the directions of inequalities. All possible sets of directions are in turn determined by understanding the orders of the polynomials $\mathcal{P}$ defined by (6). Thus, solving the AC-LEP provides us with an a priori optimal potentially nonlinear decomposition of parameter space.

2.2. Mathematical interpretation of regulatory networks. We now turn to our mathematical interpretation of a regulatory network that is compatible with its use as a biological model. With this in mind, we assign to node m a state variable, $x_m > 0$, that corresponds to the quantity of a protein, mRNA, or a signaling molecule. Precise nonlinear expressions for the interactions of the variables are *not* assumed to be known, but we do assume that the sign of the rate of change of x_m is determined by the expression

$$-\gamma_m x_m + \Lambda_m(x), \tag{8}$$

where γ_m indicates the decay rate and Λ_m is a parameter dependent function that characterizes the rate of growth of x_m. Note that Λ_m is a function of x_n if and only if there exists an edge from n to m in the regulatory network.

Since the biological model provides minimal information about the effect of x_n on x_m we want to choose a mathematical expression with a minimal set of assumptions. The rates of growth of x_m due to x_n are labeled as $0 < \ell_{m,n} < \ell_{m,n} + \delta_{m,n}$. Figure 1(b) corresponds to an edge $n \to m$, and $\theta_{m,n}$ indicates that the rate of increase $\ell_{m,n}$ must occur at some lesser value of x_n and the rate of increase $\ell_{m,n} + \delta_{m,n}$ must occur at some greater value of x_n. An arrow of the form $n \dashv m$ leads to the opposite relation.

This introduces three positive parameters, $\ell_{m,n}$, $\delta_{m,n}$, and $\theta_{m,n}$, for each edge in the regulatory network. Note that this is the minimal number of parameters that allows one to quantify the assumption that x_n activates x_m (or equivalently that x_n represses x_m). We encode this information with the following functions:

$$\lambda^+_{m,n}(x_n) := \begin{cases} \ell_{m,n} & \text{if } x_n < \theta_{m,n}, \\ \ell_{m,n} + \delta_{m,n} & \text{if } x_n > \theta_{m,n}, \end{cases} \quad \text{and} \quad \lambda^-_{m,n}(x_n) := \begin{cases} \ell_{m,n} + \delta_{m,n} & \text{if } x_n < \theta_{m,n}, \\ \ell_{m,n} & \text{if } x_n > \theta_{m,n}. \end{cases} \tag{9}$$

We do *not* assume that the values of $\ell_{m,n}, \delta_{m,n}$, or $\theta_{m,n}$ are known (see Figure 1(b)). This is intentional as many of these parameters do not have an easy biological interpretation and/or

correspond to physical constants which are difficult or impossible to precisely measure. Thus, the goal is not to determine the dynamics at any particular choice of parameters but to determine the range and robustness of the qualitative dynamics exhibited by a network.

A regulatory network such as that of Figure 1(a) does not indicate how multiple inputs to a particular node should be processed. An approach that is used is to assume a simple algebraic relationship involving sums and products of the $\lambda^{\pm}$. As an example, a reasonable choice for x_1 of Figure 1(a) is

$$\Lambda_1(x_2, x_3, x_4) = \left(\lambda^+(x_2) + \lambda^+(x_3)\right) \lambda^-(x_4). \tag{10}$$

Observe that each $\lambda^{\pm}$ takes only two values, and therefore, generically (10) takes 8 distinct values which are precisely the values of the elements of $\mathcal{P}$ in the PSD of Example 1.3.

As is suggested in the caption of Figure 1(b), we do not interpret the values of $\lambda^{\pm}$ or Λ as literal expressions of the nonlinear interactions, but rather, we regard the associated parameter values as landmarks of whatever of the "true" nonlinear function is. This has several consequences.

1. We cannot expect (8) to provide precise information about the growth rate of x_m. Therefore we restrict our attention to asking whether x_m is increasing or decreasing. However, we wish to answer this question over all the possible parameter values γ, θ, ℓ, and δ.
2. The only values of x_m at which the dynamics of x_n change are of the form $\theta_{m,*}$. The associated hyperplanes $x_j = \theta_{k,j}$ decompose the phase space, $[0,\infty)^N$, where N is the number of nodes in the network, into N-dimensional rectangular regions called *domains*.
3. Since we have determined $\mathcal{T}\left(\{p_0,\ldots,p_7\}, \prec, (0,\infty)^6\right)$ for (10) we can determine all possible signs of (8) associated with (10) by cataloguing the relative values of $\gamma_1\theta_{j,1}$, $j = 4, 5$, with respect to $\{p_0,\ldots,p_7\}$.

The Dynamic Signatures Generated by Regulatory Networks (DSGRN) library contains software that, given the information of the form provided by consequence 3, is capable of efficiently building a database of the global dynamics of a regulatory network over all of parameter space [17, 33]. In [17] the authors considered networks whose nodes have at most three in-edges and at most three out-edges. This constraint was due to the difficulty in solving the PSD problem, and the results of this paper provide a means to vastly expand the capabilities of DSGRN [33]. In particular, DSGRN can now handle the algebraic combinations of 4 to 6 in-edges, as is indicated in section 5, and arbitrarily many out-edges.

We remark that DSGRN is proving to be a versatile tool in the realm of systems and synthetic biology with applications in the realm of regulatory networks relevant to oncology [17, 24, 53], developmental biology [20], yeast cell cycle [16], and synthetic biology [23].

2.3. Extensions. The focus of this paper is on the PSD problem because of the immediate need, arising from applications, to expand the capabilities of DSGRN to handle regulatory networks with nodes that have more than three in- and out-edges. However, as should be clear from section 2.1 the mathematical foundations for our approach is quite general. In particular, the dynamics associated with functions described in section 2.2 or more generally with the PSD problem (see Definition 4.6) represent a rather special subclass. Two immediate

generalizations that are being pursued in other work, but rely on application of the techniques of this work, are as follows.

The first generalization is to replace the constant γ_m in (8) by a function $\gamma_m(x)$. This allows the DSGRN strategy to be applied in the systems biology setting to regulatory networks that involve post-transcriptional regulation. The second generalization is to replace the functions $\lambda^{\pm}_{m,n}$ given in (9) by a step function involving multiple steps without necessarily assuming monotonicity. In this setting we no longer have the Boolean lattice as the minimal partial order, but the essential arguments of this paper still apply.

3. Solving the LC-LEP.

In this section, we provide an efficient algorithm to solve the LC-LEP defined in section 1. Note that if $q \in \mathbb{R}[x_1, \ldots, x_d]$ is a linear polynomial, then evaluation of q defines a linear functional on $\mathbb{R}^d$. Thus, there exists a unique vector $\mathbf{u}_q \in \mathbb{R}^d$, that we call the *representation vector* for q, satisfying

$$q(\xi) = \mathbf{u}_q \cdot \xi \qquad \text{for all} \quad \xi \in \mathbb{R}^d.$$

Recall that the evaluation domain for the LC-LEP is the interior of a polyhedral cone. Thus, there exists a collection of linear polynomials $\mathcal{Q}_\Xi$ such that

$$\Xi = \left\{ \xi \in \mathbb{R}^d : \xi \cdot \mathbf{u}_q > 0 \text{ for all } \quad q \in \mathcal{Q}_\Xi \right\}. \tag{11}$$

We assume that (11) is satisfied for the remainder of this section.

To foreshadow our approach recall that by definition $\sigma \in \mathcal{T}(\mathcal{P}, \prec, \Xi)$ if and only if $\Xi_\sigma \neq \emptyset$. Our approach to determining the latter is to recast it in the language of linear algebra on cones in $\mathbb{R}^d$. From this perspective, the problem is equivalent to rigorously solving a linear programming problem, and the efficacy of our algorithm is based on the fact that this can be done efficiently. With this goal in mind, we begin with a few remarks concerning cones and ordered vector spaces.

3.1. Cones.

Definition 3.1. *A subset $C \subset \mathbb{R}^d$ is a* cone *if $v \in C$ and $\theta \in [0, \infty)$ implies that $\theta v \in C$. The cone C is* pointed *if it is closed and convex and satisfies*

$$C \cap -C = C \cap \{-v : v \in C\} = 0. \tag{12}$$

Observe that (12) implies that a pointed cone does not contain any lines. A vector $v \in \mathbb{R}^d$ is a *conic combination* of $\{v_1, \ldots, v_k\} \subset \mathbb{R}^d$ if $v = \theta_1 v_1 + \cdots + \theta_k v_k$ where $\theta_1, \ldots, \theta_k \geq 0$. Suppose $V = \{v_1, \ldots, v_k\} \subset \mathbb{R}^d$. The *conic hull* of V is given by

$$\text{cone}(V) := \left\{ \sum_{i=1}^{k} \theta_i v_i : 0 \leq \theta_i, \ i = 1, \ldots, k \right\}.$$

The following result is left to the reader to check.

Proposition 3.2. *Given $V = \{v_1, \ldots, v_k\} \subset \mathbb{R}^d$, $\text{cone}(V)$ is the smallest closed convex cone that contains V.*

We make use of the following propositions.

Proposition 3.3. *Suppose* $V = \{v_0, \ldots, v_m\} \subset \mathbb{R}^d$ *is a collection of nonzero vectors such that* $\text{cone}(V)$ *is a pointed cone. Then, there exists some* $v' \in \mathbb{R}^d$ *such that* $v' \cdot v_i > 0$ *for all* $0 \leq i \leq m$.

Proof. Observe that $-v_m \notin \text{cone}(\{v_0, \ldots, v_{m-1}\}) \subset \text{cone}(V)$ since $\text{cone}(V)$ is pointed. Hence $\{-v_m\}$ and $\text{cone}(\{v_0, \ldots, v_{m-1}\})$ are disjoint, convex, closed subsets of $\mathbb{R}^d$.

Therefore, by the hyperplane separation theorem [5], there exists $v' \in \mathbb{R}^d$ such that $v' \cdot -v_m < 0$ and $v' \cdot v > 0$ for any $v \in \text{cone}(\{v_0, \ldots, v_{m-1}\})$. ∎

Proposition 3.4. *Suppose* $V = \{v_0, \ldots, v_m\} \subset \mathbb{R}^d$ *is a collection of nonzero vectors such that* $\text{cone}(V)$ *is a pointed cone. If* $-v \notin \text{cone}(V)$, *then* $\text{cone}(V \cup \{v\})$ *is pointed.*

Proof. Suppose that $\text{cone}(V \cup \{v\})$ is not pointed. Then, there exists $w \neq 0$ such that $w, -w \in \text{cone}(V \cup \{v\})$ or equivalently

$$w = \sum_{i=0}^{m} \alpha_i v_i + \alpha v \quad \text{and} \quad -w = \sum_{i=0}^{m} \beta_i v_i + \beta v$$

where $\alpha_i, \beta_i, \alpha, \beta$ are all nonnegative. Note that if $\alpha = \beta = 0$, then $\pm w \in \text{cone}(V)$, which contradicts the assumption that V is pointed. The sum of the two equations above is

$$-(\alpha + \beta)v = \sum_{i=0}^{m} (\alpha_i + \beta_i) v_i.$$

This implies that $-v \in \text{cone}(V)$, contradicting the assumption that $\text{cone}(V)$ is pointed. ∎

The previous propositions illustrate the importance of solving the *cone inclusion* problem: given a vector $v \in \mathbb{R}^d$ and finite set of vectors $V \subset \mathbb{R}^d$, determine whether or not $v \in \text{cone}(V)$. Algorithm 1, stated below, solves this problem. Observe that checking if $v \in \text{cone}(V)$ is equivalent to solving the following linear programming feasibility problem:

$$\begin{aligned} &\text{Does there exist} \quad \alpha \\ &\text{such that} \quad \mathbf{V}\alpha = v \\ &\text{and} \quad \alpha \geq 0? \end{aligned} \tag{13}$$

where $\mathbf{V}$ is the column matrix of V.

Linear programming is a powerful tool that is widely used in convex optimization, and as a result, there are many available solvers/algorithms for solving the linear programming feasibility problem [50]. The results can be made rigorous by performing computations using interval arithmetic [47] or rational linear programming [2] in the case that $\mathbf{V}$ is rational. Observe that the PSD problem defined in section 1 satisfies this constraint. As is made clear in section 5, we use different solvers depending on the machine employed to do the computations. In any case, we take for granted the existence of a rigorous solver for the feasibility problem in (13) as a "black box" which we call `LPSolver` which is employed in the following algorithm.

Algorithm 1: Cone inclusion

Input: $v, V = \{v_1, \ldots, v_m\}$, `LPSolver`
Output: **True**, **False**
Result: Return **True** if $v \in \text{cone}(V)$ otherwise **False**

```
1 Function InCone(v, V, LPSolver):
2 |  Return LPSolver(v, V)
3 End Function
```

The next algorithm uses Proposition 3.4 (see line 4) and Algorithm 1 to determine if a set of vectors defines a pointed cone.

Algorithm 2: Cone pointedness

Input: $V = \{v_1, \ldots, v_m\}$
Output: **True**, **False**
Result: Return **True** if $\text{cone}(V)$ is pointed otherwise **False**

```
1  Function CheckCone(V):
2  |  V' = {v_1}
3  |  for i = 2 ... m do
4  |  |  if InCone(−v_i, V') then
5  |  |  |  Return False
6  |  |  else
7  |  |  |  V' = V' ∪ {v_i}
8  |  |  end
9  |  end
10 |  Return True
11 End Function
```

We now show that the LC-LEP can be reformulated as a problem of identifying whether some specific subsets of vectors generate pointed cones.

Definition 3.5. *Given an instance of the LC-LEP,* $(\mathcal{P}, \prec, \Xi)$*, we define the* base cone *as* $\text{cone}(V_0) := \text{cone}(V_\Xi \cup V_\prec)$ *where* V_Ξ *and* $V_\prec$ *are defined as follows. Set*

$$V_\Xi := \{\mathbf{u}_q : q \in \mathcal{Q}_\Xi\},$$

where $\mathcal{Q}_\Xi$ *are the representation vectors as defined in* (11)*. Applying Algorithm 2 to* V_Ξ *(and the fact that we assume* $\Xi \neq \emptyset$*) shows that* $\text{cone}(V_\Xi)$ *is pointed. Define*

$$V_\prec := \{\mathbf{u} : \mathbf{u} \text{ is the representing vector of } p - q \text{ where } q \prec p \text{ and } p, q \in \mathcal{P}\}.$$

Observe that if $(\mathcal{P}, \prec)$ *satisfies* (1)*, then the representation vector for* $p - q$ *is an element of* $V_\prec$ *by definition. Therefore, by Proposition 3.3,* $V_\prec$ *is pointed.*

The motivation behind our definition of V_0 is that it characterizes the algebraic constraints in the LC-LEP in terms of linear algebra that can be efficiently checked. The next proposition shows that the same idea works for linear extensions.

Given $\sigma \in S_{K+1}$, we define

$$V_\sigma := V_0 \cup \left\{\mathbf{u}_{p_{\sigma(i+1)}} - \mathbf{u}_{p_{\sigma(i)}} : p_{\sigma(i)} \in \mathcal{P},\ i = 0, \ldots, K-1\right\}. \tag{14}$$

Proposition 3.6. *For any* $\sigma \in S_{K+1}$, $\Xi_\sigma \neq \emptyset$ *if and only if* $\mathrm{cone}(V_\sigma)$ *is a pointed cone.*

This proposition is a trivial application of a well known result in convex optimization. If $\bar{\Xi}$ denotes the closure of Ξ, then $\bar{\Xi}$ is a convex cone. Its *dual cone* is defined to be the set

$$\left\{ y \in \mathbb{R}^d : y \cdot \xi \geq 0 \text{ for all } \xi \in \bar{\Xi} \right\},$$

and we observe that this set is nothing more than $\mathrm{cone}(V_\Xi)$. Similarly, for each $\sigma \in S_{K+1}$, $\overline{\Xi_\sigma}$ is again a convex cone, as it is simply a restriction of $\bar{\Xi}$ obtained by imposing finitely many additional linear constraints, and $\mathrm{cone}(V_\sigma)$ is its associated dual cone. Consequently, Proposition 3.6 follows from the fact that a convex cone is pointed if and only if its dual cone is nonempty. A proof of this result can be found in [21]. We emphasize that the importance of Proposition 3.6 is the implied equivalence

$$\mathcal{T}(\mathcal{P}, \prec, \Xi) = \{\sigma : \Xi_\sigma \neq \emptyset\} = \{\sigma : \mathrm{cone}(V_\sigma) \text{ is pointed}\}.$$

3.2. An algorithm for identifying $\mathcal{T}(\mathcal{P}, \prec, \Xi)$. In this section we present an algorithm for solving an arbitrary instance of the LC-LEP.

Algorithm 3: LC-LEP solver

Input: $\sigma_{\text{part}} = [\,], \mathcal{P}, V = V_0, \texttt{Ret} = \{\}$
Output: $\mathcal{T}(\mathcal{P}, \prec, \Xi)$
Result: `Ret`: collection of all linearly realizable total order under restriction of V

```
Function OrderingGenerator(σ_part, P, V, Ret):
    if σ_part == [ ] and CheckCone(V) is not True then
        Return
    end
    l + 1 = length of σ_part
    if l == K then
        add σ_part to Ret
        Return
    end
    for i = 0 .. K do
        if i ∉ σ_part then
            u' = u_{p_i} − u_{p_{σ_part(l)}}
            if not InCone(−u', V) then
                OrderingGenerator(σ_part + [i], P, V ∪ {v'}, Ret)
            end
        end
    end
End Function
```

In the Algorithm 3, for convenience, we take $\mathbf{u}_{p_{\sigma_{\text{part}}(-1)}} = 0$. To prove the correctness of the algorithm it is useful to denote the return of Algorithm 3 given input $(\mathcal{P}, \prec, \Xi)$ as $\mathcal{T}_{alg}(\mathcal{P}, \prec, \Xi)$.

Definition 3.7. *For fixed* $(\mathcal{P}, \prec, \Xi)$, $\sigma \in S_{K+1}$ *and for* $k = 1, \ldots, K$, *define*

$$V_{\sigma,k} = \{\mathbf{u}_{p_{\sigma(i)}} - \mathbf{u}_{p_{\sigma(i-1)}}\}_{i=1,\ldots,k} \cup V_0,$$

where V_0 *is the base cone for* $(\mathcal{P}, \prec, \Xi)$ *as in Definition* 3.5, *and* $\mathbf{u}_{p_{\sigma(j)}}, j = 0, \ldots, K$, *is the representation vector of* $p_j \in \mathcal{P}$. *For convenience, we define* $V_{\sigma,0} = V_0$, *and we observe that* $V_{\sigma,K} = V_\sigma$ *from* (14).

Theorem 3.8. *Algorithm* 3 *solves the LC-LEP.*

Proof. Given $(\mathcal{P}, \prec, \Xi)$, we need to show that $\mathcal{T}(\mathcal{P}, \prec, \Xi) = \mathcal{T}_{alg}(\mathcal{P}, \prec, \Xi)$. We may assume that $\text{cone}(V_0)$ is pointed since if not, then both $\mathcal{T}_{alg}(\mathcal{P}, \prec, \Xi)$ and $\mathcal{T}(\mathcal{P}, \prec, \Xi)$ are empty.

We first show that $\mathcal{T}_{alg}(\mathcal{P}, \prec, \Xi) \subset \mathcal{T}(\mathcal{P}, \prec, \Xi)$, i.e., for any $\sigma \in \mathcal{T}_{alg}(\mathcal{P}, \prec, \Xi)$ we show that the set $\Xi_\sigma \neq \emptyset$. As indicated above we assume $\text{cone}(V_0) = \text{cone}(V_{\sigma,0})$ is pointed. For Algorithm 3, lines 2–4 returns the empty set if $\text{cone}(V_0)$ is not pointed. Otherwise, it passes to lines 5–9 which check if σ_{part} is a total order over $\{0, \ldots, K\}$. If so, it is added to the return variable, `Ret`. If σ_{part} is not a total order, then lines 10–17 extend it to a total order by recursively constructing $V_{\sigma,i}$ from $V_{\sigma,i-1}$ for $1 \leq i \leq K$.

Therefore, it suffices to show that $V_{\sigma,k}$ are all pointed for $k = 1, \ldots, K$.

Fix $k \in \{1, \ldots, K\}$. In lines 11–12, we find a candidate $i \in \{0, \ldots, K\}$ which is not in the image of σ_{part}, and we define $V_{\sigma,k+1} = V_{\sigma,k} \cup \{\mathbf{u}'\}$ where $\mathbf{u}' = \mathbf{u}_{p_i} - \mathbf{u}_{p_{\sigma(k)}}$. In line 13, we verify that $-\mathbf{u}' \notin V = V_{\sigma,k-1}$, and it follows from Proposition 3.4 that $\text{cone}(V_{\sigma,k})$ is pointed. For each σ appended to `Ret`, we have that $\text{cone}(V_{\sigma,k})$ is pointed for $k = 0, \ldots, K$. In particular, $\text{cone}(V_{\sigma,K}) = \text{cone}(V_\sigma)$ is pointed, and from Proposition 3.6, we have $\Xi_\sigma \neq \emptyset$.

We now prove that $\mathcal{T}(\mathcal{P}, \prec, \Xi) \subset \mathcal{T}_{alg}(\mathcal{P}, \prec, \Xi)$. Assume that $\sigma \in \mathcal{T}(\mathcal{P}, \prec, \Xi)$. By definition this means that $\Xi_\sigma \neq \emptyset$, and from Proposition 3.6, V_σ is pointed. For each $k = 1, \ldots, K$, we have $V_{\sigma,k} \subset V_\sigma$, and thus $V_{\sigma,k}$ is pointed. As $V_{\sigma,k}$ is pointed, we know $-(\mathbf{u}_{p_{\sigma(k)}} - \mathbf{u}_{p_{\sigma(k-1)}}) \notin V_{\sigma,k-1}$ for $k = 1, \ldots, K$. Therefore, line 13 in Algorithm 3 will not fail to extend σ at each step in the recursion, and after K recursive extensions, σ will be appended to `Ret`, and thus, $\sigma \in \mathcal{T}_{alg}(\mathcal{P}, \prec, \Xi)$. ∎

3.3. Complexity analysis. As discussed in section 1, computing linear extensions of a poset is at least as hard as counting them, which has been established to be a difficult problem. While the algebraic constraints induced by the evaluation domain reduce the number of admissible linear extensions, it is not clear whether or not the constrained problem is easier (i.e., whether or not the LC-LEP is also $\#P$-complete). The answer may depend on the partial order, the evaluation domain, or both. In any case, this appears to be a difficult and open problem which is outside the scope of this work. Consequently, we do not attempt to provide any rigorous asymptotic estimates of Algorithm 3.

However, for the instances of LC-LEP solved in this work, our algorithm is dramatically more efficient than a brute force algorithm as evidenced by the results in section 5. By brute force we mean an algorithm which checks every linear order on $\mathcal{P}$ to determine whether or not it is admissible. Thus we conjecture that Algorithm 3 is more efficient in the typical case which explains our results.

A heuristic, but not rigorous, justification is as follows. We start by assuming a fixed implementation for the `InCone` function in Algorithm 1. This is the core algorithm in the

sense that it dominates the computational cost of Algorithm 3. Let $(\mathcal{P}, \prec, \Xi)$ be a given instance of the LC-LEP where $\mathcal{P} = \{p_0, \ldots, p_K\} \subset \mathbb{R}[x_1, \ldots, x_d]$, and let $\mathcal{T}(\mathcal{P}, \prec, \Xi) \subseteq S_{K+1}$ denote the solution of the LC-LEP.

We consider the number of `InCone` function calls required to solve this instance (i.e., we suppose that each call to `InCone` has unit computational cost). It is trivial to count the number of calls to `InCone` necessary for a brute force search. By Proposition 3.6, Ξ_σ is nonempty for any $\sigma \in S_{K+1}$ if and only cone(V_σ) is pointed which is determined by a single `InCone` call. Therefore, a brute force search recovers $\mathcal{T}(\mathcal{P}, \prec, \Xi)$ in exactly $(K+1)!$ calls to `InCone`.

To contrast this with Algorithm 3, consider a fixed $\sigma \in S_{K+1}$ which is *not* admissible. Hence, there exists a least index, $i \in \{0, \ldots, K\}$, such that

$$\sigma'\big|_{\{0,\ldots,i-1\}} = \sigma\big|_{\{0,\ldots,i-1\}} \qquad \text{for at least one } \sigma' \in \mathcal{T}(\mathcal{P}, \prec, \Xi)$$

and

$$\sigma'\big|_{\{0,\ldots,i\}} \neq \sigma\big|_{\{0,\ldots,i\}} \qquad \text{for any } \sigma' \in \mathcal{T}(\mathcal{P}, \prec, \Xi).$$

Consequently, in the ith recursive call, line 13 of Algorithm 3 returns `False`, and σ is "pruned" (i.e., removed from consideration as a candidate solution). In other words, σ has already been ruled incompatible with the algebraic constraints imposed by $\prec$ and Ξ using only its partial image obtained by restriction to the subset $\{0, \ldots, i\}$. The key observation is that this applies to every such incompatible linear extension. In other words, every $\sigma' \in S_{K+1}$ satisfying

$$\sigma'\big|_{\{0,\ldots,i\}} = \sigma\big|_{\{0,\ldots,i\}}$$

is also pruned in this step. Evidently, there are $(K-i)!$ such extensions which are simultaneously removed which results in saving $(K-i)!$ calls to `InCone` compared to the brute force search.

Pruning incompatible partial images early leads to an exponential reduction in the number of `InCone` calls which explains the results shown in Table 1. However, a more rigorous analysis for the complexity of Algorithm 3 amounts to estimating how early a given incompatible partial image is pruned. It is likely that this is heavily dependent on either $\prec$ or Ξ, or both, and may also depend on the order in which the partial images are extended.

4. Solving the general PSD problem.

In this section we present a solution for the PSD problem described in section 1. The solution is based on the observation that the PSD problem naturally has a Boolean lattice structure. Therefore, the linear PSD problem is an instance of the LC-LEP and reduces to a simple application of the algorithms presented in section 3. For nonlinear PSD problems, we construct a map that "embeds" it into an instance of LC-LEP (of higher dimension) in the sense that the inclusion in (7) holds for the Boolean partial order. We prove a sufficient condition for which this inclusion is equality and describe a method for disqualifying spurious solutions when it is strict.

4.1. The PSD as an instance of AC-LEP.

Throughout this section $\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ denotes a PSD problem for a fixed interaction function f of order type $\mathbf{n} \in \mathbb{N}^q$ as defined in (6) where $\prec_B$ is the Boolean lattice partial order. Our first goal is to show that $\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ satisfies (1) and, in particular, that every $\sigma \in \mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ is a linear extension of a Boolean lattice. We start by defining appropriate indices for the elements of $\mathcal{P}$.

Definition 4.1. *Suppose* $\mathbf{n} \in \mathbb{N}^q$ *is the interaction type for* $f \in \mathbb{R}[z_1, \ldots, z_n]$. *As in Definition 1.1 let* $\{I_1, \ldots, I_q\}$ *denote the indexing sets for each summand of* f *which form an integer partition of* $I := \{1, \ldots, n\}$. *For each* $1 \leq j \leq q$, *we consider* I_j *as an ordered set, and for* $1 \leq k \leq n_j$, *we let* $I_j(k)$ *denote the* kth *largest element of* I_j. *Let* $E := \{\alpha : \{1, \ldots, n\} \to \{0, 1\}\}$ *be the set of all Boolean functions defined on* I. *The* Boolean indexing map, *denoted by* $B : E \to \{0, \ldots, 2^{n-1}\}$, *is defined by the formula*

$$B(\alpha) := \sum_{j=1}^{q} \sum_{k=1}^{n_j} \alpha(I_j(k)) 2^{n - I_j(k)}.$$

In other words, E *and* B *are defined so that for each* $d \in I$, $\alpha(d)$ *is the* dth *binary digit of* $B(\alpha)$ *in little-endian order.*

We will also consider $\alpha \in E$ as a vector of Boolean functions defined as follows. Let E_j denote the set of Boolean functions defined on I_j. Then, elements of E can be represented as vectors of the form

$$\alpha = (\alpha_1, \ldots, \alpha_q) \quad \text{where } \alpha_j := \alpha\Big|_{I_j} \in E_j \quad \text{for } 1 \leq j \leq q.$$

Note that under this identification, E has the equivalent representation as $E = E_1 \times \cdots \times E_q$.

Definition 4.2. *Suppose* $\mathbf{n} \in \mathbb{N}^q$ *is the interaction type for an interaction function,* $f \in \mathbb{R}[z_1, \ldots, z_n]$ *as in Definition 1.1, and* E *denotes the corresponding Boolean indices. For* $\alpha \in E$, *define* $p_\alpha \in \mathcal{P} \subset \mathbb{R}[\ell_1, \ldots, \ell_n, \delta_1, \ldots, \delta_n]$ *by the formula*

$$p_\alpha := \prod_{j=1}^{q} \left(\sum_{k \in I_j} \ell_k + \alpha(k) \delta_k \right). \tag{15}$$

When convenient, we use a linear indexing scheme for elements of $\mathcal{P}$ which we define via the Boolean indexing map by identifying $p_i := p_\alpha$ where $\alpha = B^{-1}(i)$. To avoid confusion, we exclusively use Greek subscripts when referring to elements of $\mathcal{P}$ by their Boolean indices and Latin subscripts when referring to elements of $\mathcal{P}$ by their linear indices. We leave it to the reader to check that the linearly indexed polynomials in Examples 1.3 and 1.4 are in agreement with that of Definition 4.2 via this identification.

Definition 4.3. *Let* $\alpha, \beta \in E$ *be a pair of Boolean indices corresponding to* $\mathbf{n} \in \mathbb{N}^q$. *An ordered pair* (α, β) *satisfies the* one bit condition *if* $\alpha(I_j(k)) \leq \beta(I_j(k))$, *for all* $1 \leq j \leq q$ *and* $0 \leq k \leq n_j - 1$, *with equality for all but* exactly *one* (j, k) *pair.*

Remark 4.4. Observe that if (α, β) satisfy the one bit condition and (j_0, k_0) is the unique pair for which α and β take different values, then $\alpha(I_{j_0}(k)) = 0$ and $\beta(I_{j_0}(k)) = 1$.

Remark 4.5. The one bit condition induces a poset structure on E by setting $\alpha \prec \beta$ for each (α, β) satisfying the one bit condition and extending the relation transitively. The one bit condition is a covering relation for the partial order.

Definition 4.6. *Suppose* $\mathbf{n} \in \mathbb{N}^q$ *is the interaction type for an interaction function,* $f \in \mathbb{R}[z_1, \ldots, z_n]$ *as in Definition 1.1, and* E *denotes the corresponding Boolean indices. Let* $\mathcal{P}$ *be*

the set of polynomials indexed as in Definition 4.2. The PSD problem *is defined by the triple,* $(\mathcal{P}, \prec, (0,\infty)^{2n})$*, where* $\prec$ *is given by Remark 4.5.*

The next proposition proves that $(\mathcal{P}, \prec, (0,\infty)^{2n})$ satisfies (1) and furthermore that $(\mathcal{P}, \prec)$ is a Boolean partial order which justifies expressing the PSD problem as $\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$.

Proposition 4.7. *Consider a PSD problem* $(\mathcal{P}, \prec, (0,\infty)^{2n})$*. Then,*

1. $(\mathcal{P}, \prec)$ *is a Boolean lattice;*
2. *for any* $\alpha, \beta \in E$*, if* $\alpha \prec \beta$*, then*

$$p_\alpha(\xi) < p_\beta(\xi) \quad \textit{for all} \quad \xi \in (0,\infty)^{2n}.$$

Proof. To prove the first claim, let $S_n = \{1, \ldots, n\}$, and let $(2^{S_n}, \prec_B)$ denote the standard Boolean lattice. Define a map, $\varphi : E \to 2^{S_n}$, by the formula

$$\varphi(\alpha) = \{j \in S_n : \alpha(j) = 1\},$$

and we note that φ is a bijection since E is defined to be the collection of all Boolean maps defined on S_n. Furthermore, for any $\alpha, \beta \in E$, we have by Definition 4.3 that $\alpha \prec \beta$ if and only if

$$\{j \in S_n : \alpha(j) = 1\} \subset \{j \in S_n : \beta(j) = 1\}$$

implying that φ is an order isomorphism.

To establish the second claim, we must show that if $\alpha \prec \beta$, then $p_\alpha(\xi) < p_\beta(\xi)$ holds for all $\xi \in (0,\infty)^{2n}$. Note that by transitivity, it suffices to prove this holds for (α, β) satisfying the one bit condition. In this case we have

$$\beta(I_j(k)) - \alpha(I_j(k)) = \begin{cases} 1 & \text{if } j = j_0 \text{ and } k = k_0, \\ 0 & \text{otherwise} \end{cases}$$

for some $j_0 \in \{1, \ldots, q\}$, $k_0 \in \{0, \ldots, n_{j_0 - 1}\}$. If $\xi = (\ell_1, \ldots, \ell_n, \delta_1, \ldots, \delta_n) \in \Xi$, then from (15) we have

$$\begin{aligned} p_\beta(\xi) &= \left(\left(\sum_{k \in I_{j_0}} \ell_k + \alpha(I_j(k))\delta_k \right) + \delta_{k_0} \right) \prod_{j \neq j_0} \sum_{k \in I_j} \ell_k + \alpha(I_j(k))\delta_k \\ &= p_\alpha(\xi) + \delta_{k_0} \prod_{j \neq j_0} \sum_{k \in I_j} \ell_k + \alpha(I_j(k))\delta_k \\ &> p_\alpha(\xi) \end{aligned}$$

as required. ∎

With Proposition 4.7 in mind, we return to writing $\prec_B$ in place of $\prec$ for the PSD problem where $\prec_B$ is the partial order of a Boolean lattice inherited by $\mathcal{P}$ from the one bit condition.

4.2. The linear PSD problem. We consider two cases of the PSD problem: the interaction type $\mathbf{n} \in \mathbb{N}^q$ for the function $f \in \mathbb{R}[z_1, \ldots, z_n]$ has the form $\mathbf{n} = (n)$ or $\mathbf{n} = (1, 1, \ldots, 1)$. In the first case, f is linear (see Example 1.4) and the PSD problem is an instance of LC-LEP. In the second case, $\log f$ is linear, so after a simple change of variables, we obtain an instance of LC-LEP with equivalent solutions since log is monotone and hence order preserving. We focus on the first case, leaving it to the reader to check that the second case is same modulo the evaluation domain ($\mathbb{R}^{2n}$ versus $(0, \infty)^{2n}$). Following the algorithm described in section 3, we encode the partial order defined by $\prec_B$ as a set of linear constraints defined by a base cone which we must show is pointed. We begin by denoting the set of representation vectors for $\mathcal{P}$ as

$$\mathcal{V} := \left\{ \mathbf{u}_{p_\alpha} \in \{0,1\}^{2n} : \mathbf{u}_{p_\alpha} \text{ is the representation vector of } p_\alpha,\ \alpha \in E \right\}.$$

We define the set

$$V_{\prec_B} := \left\{ \mathbf{u}_{p_\beta} - \mathbf{u}_{p_\alpha} : \mathbf{u}_{p_\alpha}, \mathbf{u}_{p_\beta} \in \mathcal{V},\ (\alpha, \beta) \text{ satisfies the one bit condition} \right\}, \tag{16}$$

which encodes the $\prec_B$ partial order into the representation vectors. These vectors will be the generators of the base cone for the algorithm in section 3. Thus, we must show that $\operatorname{cone}(V_{\prec_B})$ generates a pointed cone.

Lemma 4.8. *Let $C_0 := \operatorname{cone}(V_{\prec_B})$ denote the cone generated by $V_{\prec_B}$; then C_0 is pointed.*

Proof. By Proposition 3.2, C_0 is closed and convex, so it suffices to prove that if $v \in C_0$ and $-v \in C_0$, then $v = 0$. Fix $\xi \in (0, \infty)^{2n}$, and suppose (α, β) satisfies the one bit condition. By the formula in (15) it follows that

$$p_\beta - p_\alpha = \delta_i$$

for some $i \in \{1, \ldots, n\}$. Since $\delta_i = \xi_{n+i} > 0$ for all $\xi \in \Xi$, it follows that

$$p_\beta(\xi) - p_\alpha(\xi) > 0$$

for every (α, β) satisfying the one bit condition. Passing to the representation vectors, it follows that for every $v \in V_{\prec_B}$, we have $v \cdot \xi > 0$. Taking the conic hull, we have that if $v \in C_0 \setminus \{0\}$, then $v \cdot \xi > 0$. It follows that if $v, -v \in C_0$ simultaneously, then $v \cdot \xi \geq 0$ and $-v \cdot \xi \geq 0$ implying $v = 0$. ∎

4.3. The general PSD problem. Given a general PSD problem $\left(\mathcal{P}, \prec_B, (0, \infty)^{2n}\right)$ we present the construction of an LC-LEP denoted by $(\mathcal{P}', \prec_B, \mathbb{R}^m)$ with the property that $\mathcal{T}\left(\mathcal{P}, \prec_B, (0, \infty)^{2n}\right) \subseteq \mathcal{T}(\mathcal{P}', \prec_B, \mathbb{R}^m)$. The importance of this is that $(\mathcal{P}', \prec_B, \mathbb{R}^m)$ can be solved using Algorithm 3, and hence we obtain a rigorous upper bound on $\mathcal{T}\left(\mathcal{P}, \prec_B, (0, \infty)^{2n}\right)$.

Definition 4.9. *Given an interaction type $\mathbf{n} \in \mathbb{N}^q$, let $E = E_1 \times \cdots \times E_q$ denote the corresponding Boolean indices. Set $m := \sum_{j=1}^{q} 2^{n_j}$, and define the* linearized evaluation domain *to be*

$$\mathbb{R}^{2^{n_1}} \times \cdots \times \mathbb{R}^{2^{n_q}} \cong \mathbb{R}^m. \tag{17}$$

Define a polynomial ring in m indeterminates with Boolean indexing by

$$\mathcal{R} := \mathbb{R}\left[\left\{x_{j,\alpha_j} : \alpha_j \in E_j, 1 \le j \le q\right\}\right], \tag{18}$$

and define a collection of linear polynomials by

$$\mathcal{P}' := \left\{p'_\alpha : \alpha \in E\right\} \subset \mathcal{R} \qquad \text{where} \quad p'_\alpha := \sum_{j=1}^{q} x_{j,\alpha_j}.$$

The linearized PSD problem *determined by* $\mathbf{n}$ *is to compute* $\mathcal{T}(\mathcal{P}', \prec_B, \mathbb{R}^m)$.

Theorem 4.10. *Fix an interaction type,* $\mathbf{n} \in \mathbb{N}^q$, *and let* $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ *and* $\mathcal{T}(\mathcal{P}', \prec_B, \mathbb{R}^m)$ *denote the corresponding PSD and linearized PSD problems, respectively. The following are true.*

1. *Let* $\alpha, \beta \in E$ *and* $\xi \in (0,\infty)^{2n}$. *If* $p_\alpha(\xi) < p_\beta(\xi)$ *and*

$$\xi'_{j,\alpha_j} = \log\left(\sum_{k\in I_j} \xi_k + \alpha_j(k)\xi_{n+k}\right) \in \mathbb{R}^m,$$

   then $p'_\alpha(\xi') < p'_\beta(\xi')$.
2. $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subseteq \mathcal{T}(\mathcal{P}', \prec_B, \mathbb{R}^m)$.

Proof. To prove the first claim, we define a map, $T : (0,\infty)^{2n} \to \mathbb{R}^m$, by $\xi \mapsto \xi' := T(\xi)$ where the coordinates of ξ' are given by the formula

$$\xi'_{j,\alpha_j} = \log\left(\sum_{k\in I_j} \xi_k + \alpha_j(k)\xi_{n+k}\right). \tag{19}$$

Observe that T is defined to satisfy the functional equation

$$\log \circ p_\alpha(\xi) = p'_\alpha \circ T(\xi) \qquad \text{for all} \quad \alpha \in E,\ \xi \in (0,\infty)^{2n}. \tag{20}$$

Therefore, if $\alpha, \beta \in E$ and $\xi \in (0,\infty)^{2n}$ satisfies $p_\alpha(\xi) < p_\beta(\xi)$, then $\log(p_\alpha(\xi)) < \log(p_\beta(\xi))$, and it follows from (20) that $p'_\alpha(\xi') < p'_\beta(\xi')$ where $\xi' = T(\xi)$ as required.

To prove the second claim, consider $\mathcal{P}$ and $\mathcal{P}'$ equipped with the linear indices as in Definition 4.2, and suppose $\sigma \in \mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$. Then, by definition there exists $\xi \in (0,\infty)^{2n}$ satisfying

$$p_{\sigma(0)}(\xi) < p_{\sigma(1)}(\xi) < \cdots < p_{\sigma(2^n-1)}(\xi).$$

Let $\xi' = T(\xi)$, and apply the first result to successive pairs in the ordering which implies that for all $0 \le k \le 2^n - 2$, we have

$$p'_{\sigma(k)}(\xi') = \log\left(p_{\sigma(k)}(\xi)\right) < \log\left(p_{\sigma(k+1)}(\xi)\right) = p'_{\sigma(k+1)}(\xi').$$

Thus, we $\xi' \in \mathbb{R}^m$ satisfies

$$p'_{\sigma(0)}(\xi') < p'_{\sigma(1)}(\xi') < \cdots < p'_{\sigma(2^n-1)}(\xi'),$$

and it follows that $\sigma \in \mathcal{T}(\mathcal{P}', \prec_B, \mathbb{R}^m)$ which completes the proof. ∎

Example 4.11. Recall the PSD in Example 1.3 with interaction function $f(z) = (z_1+z_2)z_3$ corresponding to interaction type $\mathbf{n} = (2,1)$. The corresponding polynomials for the linearized PSD problem are the following polynomials in $m = 6$ variables:

$$\begin{aligned} p'_0 &= x_{1,(0,0)} + x_{2,0}, & p'_4 &= x_{1,(0,1)} + x_{2,0}, \\ p'_1 &= x_{1,(0,0)} + x_{2,1}, & p'_5 &= x_{1,(0,1)} + x_{2,1}, \\ p'_2 &= x_{1,(1,0)} + x_{2,0}, & p'_6 &= x_{1,(1,1)} + x_{2,0}, \\ p'_3 &= x_{1,(1,0)} + x_{2,1}, & p'_7 &= x_{1,(1,1)} + x_{2,1}, \end{aligned}$$

where we have used linear indexing for elements of $\mathcal{P}$ to match the polynomials in Example 1.3.

As defined in (18), each variable of the form $x_{1,(x,y)}$ denotes an indeterminate in $\mathcal{R}$ which is identified with the element $\alpha_1 \in E_1$, which satisfies $\alpha_1(1) = x$ and $\alpha_1(2) = y$. In other words, the binary vector subscript (x,y) denotes the two values which $\alpha_1 \in E_1$ takes for the inputs in $I_1 = \{1,2\}$. Similarly, $x_{2,x}$ is identified with $\alpha_2 \in E_2$ satisfying $\alpha_2(3) = x$.

4.4. Solving the PSD problem for interaction type $\mathbf{n} = (2,1,\dots,1)$. In this section we prove the following theorem.

Theorem 4.12. *Let f be an interaction function with interaction type, $\mathbf{n} = (2,1,\dots,1)$. Let $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ denote the corresponding PSD problem and $(\mathcal{P}', \prec_B, \mathbb{R}^m)$ the associated linearized PSD problem. Then $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) = \mathcal{T}(\mathcal{P}', \prec_B, \Xi')$ where $\Xi' = \mathbb{R}^m \cap \{-\xi'_{1,0} + \xi'_{1,1} + \xi'_{1,2} - \xi'_{1,3} > 0\}$.*

The proof of the theorem is based on the following lemma.

Lemma 4.13. *Fix parameters, $x_0, x_1, x_2, x_3 \in \mathbb{R}$, and define the function $g : \mathbb{R} \to \mathbb{R}$ by the formula*

$$g(t) = \exp(tx_0) - \exp(tx_1) - \exp(tx_2) + \exp(tx_3).$$

If $x_0 < x_1 \le x_2 < x_3$, then g has a positive root if and only if $g'(0) < 0$.

Proof. Suppose first that t_0 is a root of g. Expanding $\exp(t_0x_1)$ and $\exp(t_0x_2)$ to first order about x_0 and x_3, respectively, and applying the mean value theorem yield the formula

$$g(t_0) = -t_0 \exp(t_0c_1)(x_1 - x_0) - t_0 \exp(t_0c_2)(x_2 - x_3) = 0 \tag{21}$$

for some $c_1 \in (x_0, x_1)$ and $c_2 \in (x_2, x_3)$. We define $k = c_2 - c_1$ and multiply (21) by $t_0e^{-kt_0}$ to obtain

$$e^{kt_0}(x_3 - x_2) - (x_1 - x_0) = 0.$$

Noting that $c_1 < x_2 < c_2$, it follows that $k > 0$. Therefore if $t_0 > 0$, then $x_3 - x_2 < x_1 - x_0$ or equivalently, $g'(0) = x_0 - x_1 - x_2 + x_3 < 0$.

Conversely, if $g'(0) < 0$, then g has at least one positive root since clearly $g(0) = 0$ and $\lim_{t\to\infty} g(t) = \infty$. ∎

Proof of Theorem 4.12. Suppose $\sigma \in \mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$, $(0,\infty)^{2n}_\sigma$ is the associated realizable set as defined in (2), and $\xi \in (0,\infty)^{2n}_\sigma$; then by Theorem 4.10 we have $\xi' = T(\xi) \in \mathbb{R}^m_\sigma$. Note that by definition the first four coordinates of ξ' are given by the formulas

$$\begin{aligned}\xi'_{1,0} &= \log(\xi_1 + \xi_2),\\ \xi'_{1,1} &= \log(\xi_1 + \xi_2 + \xi_{n+2}),\\ \xi'_{1,2} &= \log(\xi_1 + \xi_2 + \xi_{n+1}),\\ \xi'_{1,3} &= \log(\xi_1 + \xi_2 + \xi_{n+1} + \xi_{n+2}).\end{aligned}$$

Since $\xi_i > 0$ for $i \in \{1, 2, n+1, n+2\}$, it follows that

$$-\xi'_{1,0} + \xi'_{1,1} + \xi'_{1,2} - \xi'_{1,3} > 0,$$

so we have $\sigma \in \mathcal{T}(\mathcal{P}', \prec_B, \Xi')$.

Conversely, suppose $\sigma \in \mathcal{T}(\mathcal{P}', \prec_B, \Xi')$ and $\xi' \in \Xi'_\sigma$. From the Boolean lattice $\prec_B$ we have $\xi'_{1,0} < \xi'_{1,1} \le \xi'_{1,2} < \xi'_{1,3}$ or $\xi'_{1,0} < \xi'_{1,2} \le \xi'_{1,1} < \xi'_{1,3}$. Moreover, ξ' also satisfies $-\xi'_{1,0} + \xi'_{1,1} + \xi'_{1,2} - \xi'_{1,3} > 0$. Hence, Lemma 4.13 implies that there exists $t' > 0$ such that $\hat{\xi}' := t'\xi'$ satisfies

$$\exp(\hat{\xi}'_{1,0}) - \exp(\hat{\xi}'_{1,1}) - \exp(\hat{\xi}'_{1,2}) + \exp(\hat{\xi}'_{1,3}) = 0.$$

Next, we define $\hat{\xi} \in (0,\infty)^{2n}$ by

$$\hat{\xi}_j = \begin{cases} \exp(\hat{\xi}'_{j,0}), & 2 < j \le n,\\ \exp(\hat{\xi}'_{j,1}) - \exp(\hat{\xi}'_{j,0}), & n+2 < j < 2n,\\ \frac{1}{2}\exp(\hat{\xi}'_{1,0}), & j = 1, 2,\\ \exp(\hat{\xi}'_{1,2}) - \exp(\hat{\xi}'_{1,0}), & j = n+1,\\ \exp(\hat{\xi}'_{1,1}) - \exp(\hat{\xi}'_{1,0}), & j = n+2.\end{cases}$$

One easily verifies that $\hat{\xi}_j > 0$ for all $1 \le j \le 2n$, and that $T(\hat{\xi}) = \hat{\xi}'$. From Theorem 4.10, we have $\hat{\xi} \in (0,\infty)^{2n}_\sigma = \{\xi \in (0,\infty)^{2n} : p_{\sigma(0)}(\xi) < p_{\sigma(1)}(\xi) < \cdots < p_{\sigma(2^n-1)}(\xi)\}$ which implies that $\sigma \in \mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$. ∎

4.5. Solving the general PSD problem. In the general case, we have $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subsetneq \mathcal{T}(\mathcal{P}', \prec_B, \mathbb{R}^m)$, and thus, computing $\mathcal{T}(\mathcal{P}', \prec_B, \mathbb{R}^m)$ provides only a set of candidates for $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$. This candidate set contains spurious linear extensions, so we consider the problem of removing linear extensions which are nonadmissible. We have two strategies for doing this efficiently.

The first is to restrict the evaluation domain to a strict subset, $\Xi' \subsetneq \mathbb{R}^m$, such that we still have the inclusion

$$\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subseteq \mathcal{T}(\mathcal{P}', \prec_B, \Xi'). \tag{22}$$

Restricting to a smaller evaluation domain amounts to imposing more of the algebraic constraints a priori which results in improved efficiency. In order for the candidate set on the right-hand side to be efficiently computable using the algorithm in section 3, it must be an instance of the LC-LEP, i.e., Ξ' should be the interior of a polyhedral cone. For example, for the PSD with interaction type $\mathbf{n} = (2, 1, \ldots, 1)$, analyzed in section 4.4, we computed on the restricted domain

$$\Xi' = \mathbb{R}^m \cap \left\{\xi' \in \mathbb{R}^m : -\xi'_{1,0} + \xi'_{1,1} + \xi'_{1,2} - \xi'_{1,3} > 0\right\}.$$

In terms of the algorithm in section 3, this domain restriction amounts to taking our base cone in Algorithm 3 to be cone(V_0), where

$$V_0 = V_{\prec_B} \cup \{\mathbf{u}\}$$

and $\mathbf{u}$ is the representation vector for the linear functional defined by the formula

$$x \mapsto -x_{1,0} + x_{1,1} + x_{1,2} - x_{1,3}.$$

The requirement that this linear functional must be strictly positive is a special case of the following lemma whose proof is a trivial computation.

Lemma 4.14. *Suppose $\alpha, \alpha', \beta, \beta'$ are Boolean indices such that for any $\xi \in (0,\infty)^{2n}$, the following equations are satisfied:*

$$p_\alpha(\xi) < p_\beta(\xi) < p_{\beta'}(\xi) < p_{\alpha'}(\xi),$$
$$p_\alpha(\xi) + p_{\alpha'}(\xi) = p_\beta(\xi) + p_{\beta'}(\xi).$$

Then,

$$\log(p_\alpha(\xi)) + \log(p_{\alpha'}(\xi)) - \log(p_\beta(\xi)) - \log(p_{\beta'}(\xi)) > 0.$$

Lemma 4.14 provides a means to restrict the evaluation domain for the general linearized PSD problem as follows. Fix $j \in \{1,\dots,q\}$, and suppose $\{\alpha,\alpha',\beta,\beta'\} \subset E$ differ only in the jth coordinate with $\alpha \prec_B \beta \prec_B \beta' \prec_B \alpha'$, and also assume that $B(\alpha) + B(\alpha') = B(\beta) + B(\beta)'$ where B is the Boolean indexing map. Then, it follows that for any $\xi \in \Xi$, the values $\{p_\alpha(\xi), p_{\alpha'}(\xi), p_\beta(\xi), p_{\beta'}(\xi)\}$ satisfy both equations in Lemma 4.14. Therefore, if $\mathbf{u}(\{\alpha,\alpha',\beta,\beta'\})$ is the representation vector for the linear functional defined by

$$x \mapsto x_{j,B(\beta)} + x_{j,B(\beta')} - x_{j,B(\alpha)} - x_{j,B(\alpha')},$$

then $v(\{\alpha,\alpha',\beta,\beta'\})$ lies in V_σ for any $\sigma \in \mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$. Equivalently, we may impose the required linear constraint $x_{j,B(\beta)} + x_{j,B(\beta')} - x_{j,B(\alpha)} - x_{j,B(\alpha')} > 0$ on the evaluation domain of the linearized problem. Hence, for each $1 \le j \le q$, we define

$$V_j := \left\{\mathbf{u}(\{\alpha,\alpha',\beta,\beta'\}) : B(\alpha) + B(\alpha') = B(\beta) + B(\beta)',\ \alpha \prec_B \beta \prec_B \beta' \prec_B \alpha'\right\},$$

and for an arbitrary PSD problem, we may take our base cone to be

$$V_0 = V_{\prec_B} \cup V_\Xi \qquad \text{where} \quad V_\Xi = \bigcup_{j=1}^{q} V_j.$$

Applying Algorithm 3 with the base cone generated by V_0 is equivalent to solving the instance of LC-LEP defined by $(\mathcal{P}', \prec_B, \Xi')$ where Ξ' is the restriction of $\mathbb{R}^m$ to the subset for which the linear functionals defined by each $v \in V_j$ are strictly positive for each $1 \le j \le q$.

In addition to restricting the computation to the polyhedral cones discussed above, we can reuse solutions of smaller PSD problems in some larger computations. As an example, suppose $\mathcal{P}' = \{p'_0,\dots,p'_7\}$ is the set of interaction polynomials for the PSD with interaction

type $\mathbf{n}' = (2,1)$ and $\mathcal{P} := \{p_0, \ldots, p_{15}\}$ the polynomials for the PSD problem with interaction type $\mathbf{n} = (2,1,1)$. Observe that each admissible linear order on $\mathcal{P}'$ induces an imposed linear order on the even indexed polynomials, $\mathcal{P}_{\text{even}} := \{p_0, p_2, \ldots, p_{14}\} \subset \mathcal{P}$. A similar linear order is induced on the odd indexed polynomials, $\mathcal{P}_{\text{odd}} := \{p_1, p_3, \ldots, p_{15}\} \subset \mathcal{P}$. Hence, a necessary condition to have an admissible linear extension for $\mathcal{P}$ is that the orders of $\mathcal{P}_{\text{even}}$ and $\mathcal{P}_{\text{odd}}$ must both be consistent with one of the PSD solutions in $\mathcal{T}\left(\mathcal{P}', \prec_B, (0,\infty)^6\right)$. This implies the inclusion

$$\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^8\right) \subseteq \bigcup_{\sigma' \in \mathcal{T}(\mathcal{P}', \prec_B, (0,\infty)^6)} \mathcal{T}(\mathcal{P}, \prec_B \cup \prec_{\sigma'}, (0,\infty)^8) \tag{23}$$

where $\prec_B \cup \prec_{\sigma'}$ represents the refinement of the Boolean lattice partial order and the partial order induced by σ' on the even/odd subsets.

To exploit this in general, we say that the PSD problem of type $\mathbf{n}'$ is a subproblem of the PSD problem of type $\mathbf{n}$ whenever the polynomials for $\mathbf{n}$ must obey an implied partial order determined by the solutions of $\mathbf{n}'$. Notice that the preceding discussion as well as (23) applies also to an arbitrary polyhedral cone. Therefore, if there is a total of k admissible linear extensions for all subproblems of the PSD problem of type $\mathbf{n}$ which we have previously computed, then we bootstrap those results when computing $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ via the inclusion

$$\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subseteq \bigcup_{i=1}^{k} \mathcal{T}(\mathcal{P}, \prec_B \cup \prec_{\sigma_i'}, \Xi') \subseteq \mathcal{T}(\mathcal{P}, \prec_B, \Xi')$$

where $\prec_B \cup \prec_{\sigma_i'}$ represents the refinement of the Boolean lattice partial order and the partial order induced by σ_i' on the corresponding subsets obtained from any subproblem. This technique has been used in the computation for all the cases of order ≥ 4. Observe that the computation of $\mathcal{T}(\mathcal{P}, \prec_B \cup \prec_{\sigma_i'}, \Xi')$ can be done distributively for $i = 1, \ldots, k$ on different computational nodes, which, as is indicated in section 5, we employed for the PSD problems of orders 5 and 6.

In the special case of section 4.4, we proved that inclusion in (22) is actually equality when Ξ' is constructed as we have described. However, in the typical case, these additional algebraic constraints are not sufficient to remove all spurious linear extensions except in the case $\mathbf{n} = (2,1,\ldots,1)$. It remains an open problem to determine a smaller set Ξ' such that $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) = \mathcal{T}\left(\mathcal{P}', \prec_B, \Xi'\right)$ for other interaction types. However, in the remainder of this section we consider the problem of extracting $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ from $\mathcal{T}\left(\mathcal{P}', \prec_B, \Xi'\right)$ when they are not equal.

Observe that we may obtain large subsets of $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ simply by sampling. The particular strategy that we adopted is as follows. We uniformly sampled between 10^8 and 10^9 points

$$\xi = (l_1, \ldots, l_n, \delta_1, \ldots, \delta_n) \in \mathbb{Z}_+^{2n} \cap B_\infty^{2n}(r),$$

where $B_\infty^{2n}(r) = \{\|\xi\|_\infty \leq r\}$. We chose $r = 1000$. Mathematically the particular choice of r is not important since the PSD polynomials are homogeneous, though in practice it does have an effect on sampling precision and speed. For each such ξ we evaluated $\{p_\alpha(\xi) : p \in \mathcal{P}\}$. If

$\sigma \in S_{2^n}$ denotes the linear order of these values, then ξ serves as a "witness" for the claim that $\Xi_\sigma \neq \emptyset$. This produces

$$\mathcal{S}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) := \left\{\sigma \in \mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) : \sigma \text{ is witnessed by at least one sample}\right\}.$$

Obviously,

$$\mathcal{S}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subseteq \mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subseteq \mathcal{T}(\mathcal{P}, \prec_B, \Xi').$$

In general, sampling is relatively efficient, and in cases where $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ is not too large (see Table 1 for details), we recover the entire solution.

Once we have constructed the set $\mathcal{S}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right), \mathcal{T}(\mathcal{P}, \prec_B, \Xi')$ from sampling and algorithms in section 3, respectively, we apply a CAD algorithm to check whether or not the semialgebraic set

$$\Xi_\sigma = \{\xi \in \Xi : p_{\sigma(0)}(\xi) < p_{\sigma(1)}(\xi) < \cdots < p_{\sigma(2^n-1)}(\xi)\}$$

is empty for each $\sigma \in \mathcal{T}(\mathcal{P}, \prec_B, \Xi') \setminus \mathcal{S}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$, and then $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$ is recovered. The CAD algorithm implementation we are using is CylindricalAlgebraicDecomposition in Mathematica 11 [52].

4.6. Computational complexity. Analyzing the efficiency of our solver for the PSD problem can be broken into three cases. The first case is the linear PSD problem. This is an instance of the LC-LEP in $2n$ variables, and consequently, the complexity analysis in section 3.3 applies.

The second case is the PSD problem with interaction type $\mathbf{n} = (2, 1, \ldots, 1)$. By Theorem 4.12, the solution set is identical to the solutions of the associated linearized PSD problem. The latter is an instance of LC-LEP in $m = 4 + 2(q-1) = 2(q+1)$ variables, and since $q = n - 1$, we have $m = 2n$. In other words, the computational cost of solving this PSD problem is again equivalent to solving an instance of the LC-LEP in the same number of variables, and the analysis in section 3.3 applies.

The final case is the general PSD problem for interaction type $\mathbf{n} \in \mathbb{N}^q$, which we assume does not satisfy either of the two previous cases. Here we cannot guarantee that the complexity is any better than that of a CAD-based approach since the final step in our algorithm is to determine the admissibility of every linear extension

$$\sigma \in \mathcal{S}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \setminus \mathcal{T}(\mathcal{P}, \prec_B, \Xi').$$

That is, any total order which is admissible for the linearized PSD problem, but is not witnessed when sampling the nonlinear PSD problem, must be rigorously checked for admissibility. This is done using a CAD-based algorithm to certify whether or not the semialgebraic set, Ξ_σ, is empty for each such σ. As discussed in section 1, solving this problem requires, again in the worst case, computing a full CAD for $\mathcal{P}$ subject to the additional algebraic constraints $\ell_i > 0, \delta_i > 0$ for $1 \leq i \leq n$. Consequently, our algorithm for solving the PSD inherits the double exponential complexity of the CAD algorithm in (3). Based on the actual computations we have carried out, we conjecture that this worst case bound is not typical and may not even be sharp. There are several heuristics which make this conjecture plausible.

The first is the fact that sampling alone was sufficient to recover the full solution of the PSD problem in every case examined in this work (compare the first and last columns of Table 1). Consequently, every candidate ordering which was not ruled out by sampling was in fact nonadmissible, and the CAD-based computation only needed to certify this fact. This is advantageous due to the fact that many speedups for the general CAD algorithm apply specifically for proving that a semialgebraic set is in fact empty. Once again, each of these improvements maintains the same worst case running time, but they often produce much faster typical running times.

The second heuristic is due to the structure of the PSD problem itself. Not only are the polynomials in $\mathcal{P}$ linear with respect to each variable, but $\mathcal{P}$ itself contains a "complete" set of polynomials arising from the interaction function which forms a sort of template. This imposes stricter algebraic constraints on the linearized PSD problem as a consequence of Lemma 4.14. Additionally, this structure causes the PSD problems to nest into one another neatly in the sense of (23). These properties combine to produce sets of candidate solutions of the linearized PSD problem which are remarkably smaller than the candidates produced by solving the linearized PSD problem naively (compare the second and third columns in Table 1).

These heuristics combined with the results for the cases we have computed make a compelling case that this algorithm is already reasonably efficient, but this is by no means the last word on the subject. Further improvements to this approach via imposing additional algebraic constraints, improving sampling techniques, or applying the linearization technique to other classes of polynomials are the subject of ongoing research.

5. Results for some PSD problems.

In this section we provide (see Table 1) the results of our computations for interaction functions of orders 4, 5, and 6. A slightly different approach was taken to compute orders 5 and 6, from that used for 4. This had to do with the machines being used but highlights the flexibility of our method.

For interaction functions of order 4, we applied Algorithm 1 using a rational linear programming algorithm. In particular, we used the implementation MixedIntegerLinearProgram from SageMath 8 [43]. This implies that the output of Algorithm 3 is correct. Observe that interaction type (4) is linear and type $(1,1,1,1)$ is log linear, and therefore Algorithm 3 produces $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$. The fact that our output agrees with that of [32] suggests that our code is functioning as desired. To compute the interaction type $(2,1,1)$ we apply Algorithm 3 to obtain $\mathcal{T}\left(\mathcal{P}', \prec_B, \mathbb{R}^m\right)$. By Theorem 4.12 this determines $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$.

To solve the PSD problem from interaction types $(2,2)$ and $(3,1)$ requires that we make use of the strategy discussed in section 4.5. Again, we use Algorithm 3 to obtain $\mathcal{T}\left(\mathcal{P}', \prec_B, \mathbb{R}^m\right)$. By Theorem 4.10, $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right) \subset \mathcal{T}\left(\mathcal{P}', \prec_B, \mathbb{R}^m\right)$. As indicated in column 7 of Table 1, we chose 10^8 samples from $(0,\infty)^8$ and identified 5344 and 3084 linear orders, respectively. We ran CylindricalAlgebraicDecomposition in Mathematica 11 [52] on each element of $\mathcal{T}\left(\mathcal{P}', \prec_B, \mathbb{R}^m\right) \setminus \mathcal{S}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$. As can be seen by comparing columns 6 and 3, none of these elements were admissible.

We now turn to the computations of interaction functions of orders 5 and 6. As these problems are too big to be done on a laptop we turned to a server for which SageMath was not installed. Thus, we made use of a numerical linear programing algorithm, linprog from Python 3.5 package scipy [51], with the default numerical error 10^{-13}, in Algorithm 1. The

Table 1

*Computational results for several PSD problems. Column 1 indicates the interaction type. Column 2 provides the number of elements in the AC-LEP of interest. Column 3 provides the number of elements in an associated LC-LEP. This is not relevant where the AC-LEP problem of interest is a LC-LEP problem and is indicated by -. The * indicates that the computation was too large to complete. Column 4 provides the number of elements in the linearized PSD problem without additional constraints. Again the irrelevance for linear problems is indicated by -, and * indicates that the computation is large to be performed. The last column indicates the number of cells identified via sampling. We used 10^8 samples for all $n = 4$ cases and 10^9 samples for the $n = 5, 6$ cases. The symbol † indicates that our sampling was not sufficient.*

n	$\#\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$	$\#\mathcal{T}(\mathcal{P}', \prec_B, \Xi')$	$\#\mathcal{T}\left(\mathcal{P}', \prec_B, \mathbb{R}^m\right)$	$\#\mathcal{S}(\mathcal{P}', \prec_B, (0,\infty)^{2n})$
(1,1,1,1)	336	-	-	-
(4)	336	-	-	-
(2,1,1)	1,344	1,344	2,352	-
(2,2)	5,344	7,920	26,640	5,344
(3,1)	3,084	5,112	68,641	3,084
(1,1,1,1,1)	61,920	-	-	61,920
(5)	61,920	-	-	61,920
(2,1,1,1)	790,200	790,200	*	790,200
(2,2,1)	-	11,035,808	*	6,570,952
(3,2)	-	*	*	71,959,088†
(4,1)	-	*	*	11,213,616†
(1,1,1,1,1,1)	89,414,640	-	-	89,414,640
(6)	89,414,640	-	-	89,414,640

interaction types (5) and (6) are linear and $(1,1,1,1,1)$ and $(1,1,1,1,1,1)$ are log linear, and therefore via Algorithm 3 we obtain $\mathcal{T}_{alg}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$. We use the sampling technique (see columns 7 and 8 of Table 1) to verify each of the elements of $\mathcal{T}_{alg}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$, thereby obtaining $\mathcal{T}\left(\mathcal{P}, \prec_B, (0,\infty)^{2n}\right)$.

The computation for each order 4 case was done on a Mac Pro laptop (2.7 GHz Intel i5 and memory 8GB) with computation time under 4 hours. The computation of the remaining cases were done using a computing server with CentOs, intel 17.1, memory 32 GB, and less than 30 nodes. The computation time for both $(1,1,1,1,1)$ and (5) was less than 4 hours, while the computation time for $(2,1,1,1)$ was on the order of 7 days. The codes which produced all of the computations in Table 1 are available on GitHub.

Acknowledgments. The authors would like to thank Shaun Harker, Sandra Di Rocco, Tomas Gedeon, and Mike Saks for helpful conversations.

REFERENCES

[1] R. Albert, J. J. Collins, and L. Glass, *Introduction to Focus Issue: Quantitative approaches to genetic networks*, Chaos, 23 (2013), 025001.

[2] D. L. Applegate, W. Cook, S. Dash, and D. G. Espinoza, *Exact solutions to linear programming problems*, Oper. Res. Lett., 35 (2007), pp. 693–699.

[3] Z. Arai, W. Kalies, H. Kokubu, K. Mischaikow, H. Oka, and P. Pilarczyk, *A database schema for the analysis of global dynamics of multiparameter systems*, SIAM J. Appl. Dyn. Syst., 8 (2009), pp. 757–789, https://doi.org/10.1137/080734935.

[4] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in Real Algebraic Geometry*, Algorithms Comput. Math. 10, Springer-Verlag, Berlin, 2003, https://doi.org/10.1007/978-3-662-05355-3.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.
[6] G. Brightwell and P. Winkler, *Counting linear extensions is #P-complete*, in Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing, STOC '91, New York, NY, 1991, ACM, pp. 175–181, https://doi.org/10.1145/103418.103441.
[7] G. R. Brightwell and P. Tetali, *The number of linear extensions of the Boolean lattice*, Order, 20 (2003), pp. 333–345, https://doi.org/10.1023/B:ORDE.0000034596.50352.f7.
[8] C. W. Brown, *Improved projection for cylindrical algebraic decomposition*, J. Symbolic Comput., 32 (2001), pp. 447–465, https://doi.org/10.1006/jsco.2001.0463.
[9] C. W. Brown, *Constructing a single open cell in a cylindrical algebraic decomposition*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC, 2013, pp. 133–140, https://doi.org/10.1145/2465506.2465952.
[10] C. W. Brown and J. H. Davenport, *The complexity of quantifier elimination and cylindrical algebraic decomposition*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC, 2007, pp. 54–60, https://doi.org/10.1145/1277548.1277557.
[11] J. Bush, W. Cowan, S. Harker, and K. Mischaikow, *Conley–Morse databases for the angular dynamics of Newton's method on the plane*, SIAM J. Appl. Dyn. Syst., 15 (2016), pp. 736–766, https://doi.org/10.1137/15M1017971.
[12] J. Bush, M. Gameiro, S. Harker, H. Kokubu, K. Mischaikow, I. Obayashi, and P. Pilarczyk, *Combinatorial-topological framework for the analysis of global dynamics*, Chaos, 22 (2012), 047508, https://doi.org/10.1063/1.4767672.
[13] G. E. Collins, *Quantifier elimination for real closed fields by cylindrical algebraic decompostion*, in Automata Theory and Formal Languages, H. Brakhage, ed., Springer, Berlin, 1975, pp. 134–183.
[14] C. Conley, *Isolated Invariant Sets and the Morse Index*, CBMS Reg. Conf. Ser. Math. 38, American Mathematical Society, Providence, RI, 1978.
[15] C. C. Conley and J. A. Smoller, *The Existence of Heteroclinic Orbits, and Applications*, Lecture Notes in Phys. 38., Springer, Cham, 1975, pp. 511–524.
[16] B. Cummins, T. Gedeon, S. Harker, and K. Mischaikow, *Model rejection and parameter reduction via time series*, SIAM J. Appl. Dyn. Syst., 17 (2018), pp. 1589–1616, https://doi.org/10.1137/17M1134548.
[17] B. Cummins, T. Gedeon, S. Harker, K. Mischaikow, and K. Mok, *Combinatorial representation of parameter space for switching networks*, SIAM J. Appl. Dyn. Syst., 15 (2016), pp. 2176–2212, https://doi.org/10.1137/15m1052743.
[18] S. Day, Y. Hiraoka, K. Mischaikow, and T. Ogawa, *Rigorous numerics for global dynamics: A study of the Swift–Hohenberg equation*, SIAM J. Appl. Dyn. Syst., 4 (2005), pp. 1–31.
[19] S. Day, O. Junge, and K. Mischaikow, *A rigorous numerical method for the global analysis of infinite-dimensional discrete dynamical systems*, SIAM J. Appl. Dyn. Syst., 3 (2004), pp. 117–160.
[20] R. Diegmiller, L. Zhang, M. Gameiro, J. Barr, J. I. Alsous, P. Schedl, S. Y. Shvartsman, and K. Mischaikow, *Mapping parameter spaces of biological switches*, PLoS Comput. Biol., 17 (2021), e1008711.
[21] G. Ewald, *Combinatorial Convexity and Algebraic Geometry*, Grad. Texts in Math. 168, Springer, Cham, 1996.
[22] T. Fine and J. Gill, *The enumeration of comparative probability relations*, Ann. Probab., 4 (1976), pp. 667–673.
[23] M. Gameiro, T. Gedeon, S. Kepley, and K. Mischaikow, *Rational Design of Complex Phenotype via Network Models*, preprint, arXiv:2010.03803 [math.DS], 2020, https://arxiv.org/abs/2010.03803.
[24] T. Gedeon, B. Cummins, S. Harker, and K. Mischaikow, *Identifying robust hysteresis in networks*, PLoS Comput. Biol., 14 (2018), e1006121, https://doi.org/10.1371/journal.pcbi.1006121.
[25] S. Harker, K. Mischaikow, and K. Spendlove, *A Computational Framework for the Connection Matrix Theory*, preprint, arXiv:1810.04552 [math.AT], 2018, https://arxiv.org/abs/1810.04552.
[26] A. S. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman, *Reverse-engineering of polynomial dynamical systems*, Adv. Appl. Math., 39 (2007), pp. 477–489, https://doi.org/10.1016/j.aam.2006.08.004.

[27] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*, Appl. Math. Sci. 157, Springer-Verlag, New York, 2004, https://doi.org/10.1007/b97315.
[28] W. D. Kalies, K. Mischaikow, and R. C. A. M. van der Vorst, *An algorithmic approach to chain recurrence*, Found. Comput. Math., 5 (2005), pp. 409–449, https://doi.org/10.1007/s10208-004-0163-9.
[29] W. D. Kalies, K. Mischaikow, and R. C. A. M. van der Vorst, *Lattice structures for attractors* I, J. Comput. Dyn., 1 (2014), pp. 307–338, https://doi.org/10.3934/jcd.2014.1.307.
[30] W. D. Kalies, K. Mischaikow, and R. C. A. M. van der Vorst, *Lattice structures for attractors* II, Found. Comput. Math., 16 (2016), pp. 1151–1191, https://doi.org/10.1007/s10208-015-9272-x.
[31] W. D. Kalies, K. Mischaikow, and R. C. A. M. van der Vorst, *Lattice Structures for Attractors* III, preprint, arXiv:1911.09382 [math.DS], 2019, https://arxiv.org/abs/1911.09382.
[32] D. Maclagan, *Boolean term orders and the root system B_n*, Order, 15 (1999), pp. 279–295.
[33] B. C. Marcio Gameiro, Shaun Harker, *DSGRN: Dynamic Signatures Generated by Regulatory Networks*, 2020, https://github.com/marciogameiro/DSGRN.
[34] S. McCallum, *An improved projection operation for cylindrical algebraic decomposition*, in European Conference on Computer Algebra, Lecture Notes in Comput. Sci. 204, Springer, Cham, 1985, pp. 277–278, https://doi.org/10.1007/3-540-15984-3_277.
[35] C. McCord, *Mappings and homological properties in the Conley index theory*, Ergodic Theory Dynam. Systems, 8 (1988), pp. 175–198, https://doi.org/10.1017/S014338570000941X.
[36] C. McCord and K. Mischaikow, *On the global dynamics of attractors for scalar delay equations*, J. Amer. Math. Soc., 9 (1996), pp. 1095–1133, https://doi.org/10.1090/S0894-0347-96-00207-X.
[37] C. McCord, K. Mischaikow, and M. Mrozek, *Zeta functions, periodic trajectories, and the Conley index*, J. Differential Equations, 121 (1995), pp. 258–292, https://doi.org/10.1006/jdeq.1995.1129.
[38] K. Mischaikow, *Global asymptotic dynamics of gradient-like bistable equations*, SIAM J. Math. Anal., 26 (1995), pp. 1199–1224, https://doi.org/10.1137/S0036141093250827.
[39] K. Mischaikow and M. Mrozek, *Chaos in the Lorenz equations: A computer-assisted proof*, Bull. Amer. Math. Soc. (N.S.), 32 (1995), pp. 66–72.
[40] K. Mischaikow and M. Mrozek, *Conley index*, in Handbook of Dynamical Systems, Vol. 2, North-Holland, Amsterdam, 2002, pp. 393–460, https://doi.org/10.1016/S1874-575X(02)80030-3.
[41] J. Montgomery, *Cohomology of isolated invariant sets under perturbation*, J. Differential Equations, 13 (1973), pp. 257–299, https://doi.org/10.1016/0022-0396(73)90018-1.
[42] OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences*, 2020, https://www.sagemath.org.
[43] Sage Developers, *SageMath, the Sage Mathematics Software System (Version 8)*, https://www.sagemath.org.
[44] J. Sha and D. J. Kleitman, *The number of linear extensions of subset ordering*, Discrete Math. 63 (1987), pp. 271–278, https://doi.org/10.1016/0012-365X(87)90016-1.
[45] R. Srzednicki, *On rest points of dynamical systems*, Fund. Math., 126 (1985), pp. 69–81, https://doi.org/10.4064/fm-126-1-69-81.
[46] R. P. Stanley, *An introduction to hyperplane arrangements*, in Geometric Combinatorics, IAS/Park City Math. Ser. 13, Amer. Math. Soc., Providence, RI, 2007, pp. 389–496.
[47] H. Suprajitno and I. B. Mohd, *Linear programming with interval arithmetic*, Int. J. Contemp. Math. Sciences, 5 (2010), pp. 323–332.
[48] A. Szymczak, *The Conley index and symbolic dynamics*, Topology, 35 (1996), pp. 287–299, https://doi.org/10.1016/0040-9383(95)00029-1.
[49] S. Toda, *PP is as hard as the polynomial-time hierarchy*, SIAM J. Comput., 20 (1991), pp. 865–877, https://doi.org/10.1137/0220053.
[50] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, Springer, Cham, 2020.
[51] P. Virtanen, R. Gommers, T. E. Oliphant, et al.; SciPy 1.0 Contributors, *SciPy 1.0: Fundamental algorithms for scientific computing in Python*, Nature Methods, 17 (2020), pp. 261–272, https://doi.org/10.1038/s41592-019-0686-2.
[52] Wolfram, *Mathematica, Version 11*, https://www.wolfram.com/mathematica.
[53] Y. Xin, B. Cummins, and T. Gedeon, *Multistability in the epithelial-mesenchymal transition network*, BMC Bioinform., 21 (2020), 71, https://doi.org/10.1186/s12859-020-3413-1.