SIAM J. DISCRETE MATH.
Vol. 30, No. 3, pp. 1534–1570



# EXPLOITING CHORDAL STRUCTURE IN POLYNOMIAL IDEALS: A GRÖBNER BASES APPROACH*

DIEGO CIFUENTES† AND PABLO A. PARRILO†

**Abstract.** Chordal structure and bounded treewidth allow for efficient computation in numerical linear algebra, graphical models, constraint satisfaction, and many other areas. In this paper, we begin the study of how to exploit chordal structure in computational algebraic geometry—in particular, for solving polynomial systems. The structure of a system of polynomial equations can be described in terms of a graph. By carefully exploiting the properties of this graph (in particular, its chordal completions), more efficient algorithms can be developed. To this end, we develop a new technique, which we refer to as *chordal elimination*, that relies on elimination theory and Gröbner bases. By maintaining graph structure throughout the process, chordal elimination can outperform standard Gröbner bases algorithms in many cases. The reason is because all computations are done on "smaller" rings of size equal to the treewidth of the graph (instead of the total number of variables). In particular, for a restricted class of ideals, the computational complexity is linear in the number of variables. Chordal structure arises in many relevant applications. We demonstrate the suitability of our methods in examples from graph colorings, cryptography, sensor localization, and differential equations.

**Key words.** chordal graph, elimination theory, Gröbner bases, structured polynomials, treewidth

**AMS subject classifications.** 13P10, 68W30

**DOI.** 10.1137/151002666

## 1. Introduction.

Systems of polynomial equations can be used to model a large variety of applications. In most cases the systems have a particular sparsity structure, and exploiting such a structure can greatly improve their efficiency. When all polynomials have degree one, we have the special case of systems of linear equations, which are often represented using matrices. In such a case, it is well known that under a *chordal structure* many matrix algorithms can be performed efficiently [36, 37, 39]. Similarly, many hard combinatorial problems can be solved efficiently for chordal graphs [26]. Chordal graphs are also a keystone in constraint satisfaction, graphical models, and database theory [4,16,31]. We address the question of whether chordality might also help to solve nonlinear equations.

It is natural to expect that the complexity of "solving" a system of polynomials should depend on the underlying graph structure of the equations. In particular, a parameter of the graph called the *treewidth* determines the complexity of solving the problems described above, and it should influence polynomial equations as well. For instance, several combinatorial problems (e.g., Hamiltonian circuit, vertex colorings, vertex cover) are NP-hard in general, but are tractable if the treewidth is bounded [6]. Nevertheless, standard algebraic geometry techniques typically do not make use of this graph. This paper links Gröbner bases with this graph structure.

It should be mentioned that, unlike classical graph problems, the ubiquity of systems of polynomials makes them hard to solve in the general case, even for small treewidth. Indeed, solving zero-dimensional quadratic equations of treewidth 1 is

*Received by the editors January 5, 2015; accepted for publication (in revised form) May 4, 2016; published electronically August 18, 2016. This research was supported by Air Force Office of Scientific Research grant FA9550-11-1-0305.
http://www.siam.org/journals/sidma/30-3/100266.html

†Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 (diegcif@mit.edu, parrilo@mit.edu).

already NP-complete, as seen in the following example.

*Example* 1 (polynomials on trees are hard). Let $a_1, \ldots, a_n$ and $S$ be given integers. The subset-sum problem asks for a subset $A \subseteq \{a_1, \ldots, a_n\}$, whose sum is equal to $S$. Let $s_i$ denote the sum of $A \cap \{a_1, \ldots, a_i\}$, and note that we can recover the subset $A$ from the values of $s_1, \ldots, s_n$. We can thus formulate the problem as

$$\begin{aligned} 0 &= s_0 \\ 0 &= (s_i - s_{i-1})(s_i - s_{i-1} - a_i) \qquad \text{for } 1 \le i \le n. \\ S &= s_n \end{aligned}$$

Observe that the structure of these equations can be represented with the path graph $s_0$—$s_1$—$s_2 \cdots s_{n-1}$—$s_n$, which is a tree. However, it is well known that the subset-sum problem is NP-complete.

Despite this hardness result, it is still desirable to take advantage of this chordal structure. In this paper, we introduce a new method that exploits this structure. We refer to it as *chordal elimination*. Chordal elimination is based on ideas used in sparse linear algebra; in particular, if the equations are linear, chordal elimination defaults to sparse Gaussian elimination.

We proceed to formalize our statements. We consider the polynomial ring $R = \mathbb{K}[x_0, x_1, \ldots, x_{n-1}]$ over some algebraically closed field $\mathbb{K}$. We fix once and for all the lexicographic term order with $x_0 > x_1 > \cdots > x_{n-1}$.[1] Given a system of polynomials $F = \{f_1, f_2, \ldots, f_s\}$ in the ring $R$, we associate to it a graph $G(F)$ with vertex set $V = \{x_0, \ldots, x_{n-1}\}$. Note that the vertices of $G(F)$ inherit the order from $R$. Such a graph is given by a union of cliques: for each $f_i$, we form a clique in all of its variables. Equivalently, there is an edge between $x_i$ and $x_j$ if and only if there is some polynomial that contains both variables. We say that $G(F)$ constitutes the *sparsity structure* of $F$. In constraint satisfaction problems, $G(F)$ is usually called the primal constraint graph [16].

Throughout this paper we fix an ideal $I \subseteq R$ with a given set of generators $F$. We assume that the associated graph $G = G(F)$ is chordal. Even more, we assume that $x_0 > \cdots > x_{n-1}$ is a perfect elimination ordering (see Definition 1) of the graph. In the event that $G$ is not chordal, the same reasoning applies by considering a chordal completion. We want to compute the elimination ideals of $I$, denoted as $\mathrm{elim}_l(I)$, while preserving the sparsity structure. As we are mainly interested in the zero set of $I$ rather than finding the exact elimination ideals, we attempt to find some $I_l$ such that $\mathbf{V}(I_l) = \mathbf{V}(\mathrm{elim}_l(I))$.

QUESTION. *Consider an ideal $I \subseteq R$ with generators $F$, and fix the lex order $x_0 > x_1 > \cdots > x_{n-1}$. Assume that such an order is a perfect elimination ordering of its associated graph $G(F)$. Can we find ideals $I_l$, with some generators $F_l$, such that $\mathbf{V}(I_l) = \mathbf{V}(\mathrm{elim}_l(I))$ and the sparsity structure is preserved, i.e., $G(F_l) \subseteq G(F)$?*

We could also ask the following stronger question: Does there exist a Gröbner basis $gb$ that preserves the sparsity structure, i.e., $G(gb) \subseteq G(F)$? It turns out that it is not generally possible to find a Gröbner basis that preserves chordality, as seen in the next example.

*Example* 2 (Gröbner bases may destroy chordality). Let $I = \langle x_0 x_2 - 1, x_1 x_2 - 1 \rangle$, whose associated graph is the path $x_0$—$x_2$—$x_1$. Note that any Gröbner basis must

[1] Observe that smaller indices correspond to larger variables.

contain the polynomial $p = x_0 - x_1$, breaking the sparsity structure. Nevertheless, we can find some generators for its first elimination ideal $\text{elim}_1(I) = \langle x_1 x_2 - 1 \rangle$ that preserve such structure.

As evidenced in Example 2, a Gröbner basis with the same graph structure might not exist, but we still might be able to find its elimination ideals. Our main method, *chordal elimination*, attempts to find ideals $I_l$ as proposed above. It generalizes the ideas of sparse linear algebra. As opposed to Gaussian elimination, in the general case chordal elimination may not lead to the right elimination ideals. Nevertheless, we can certify when the ideals found are correct. This allows us to prove that for a large family of problems, which includes the case of linear equations, chordal elimination *succeeds* in finding the elimination ideals.

The aim of chordal elimination is to obtain a good description of the ideal (e.g., a Gröbner basis), while at the same time preserving the underlying graph structure. However, as illustrated above, there may not be a Gröbner basis that preserves the structure. For larger systems, Gröbner bases can be extremely large, and thus they may not be practical. Nonetheless, we can ask for some sparse generators of the ideal that are the closest to such Gröbner bases. We argue that one such representation can be given by finding the elimination ideals of all *maximal cliques* of the graph. We extend chordal elimination to compute these ideals in Algorithm 3. In the case when $I$ is zero-dimensional, it is straightforward to obtain the roots from such representation.

Chordal elimination shares many of the limitations of other elimination methods. In particular, if $\mathbf{V}(I)$ is finite, the complexity depends intrinsically on the size of the projection $|\pi_l(\mathbf{V}(I))|$. As such, it performs much better if such a set is small. In Theorem 6 we show complexity bounds for a certain family of ideals where this condition is met. Specifically, we show that chordal elimination is $O(n)$ if the *treewidth* is bounded.

Chordal structure arises in many different applications, and we believe that algebraic geometry algorithms should take advantage of it. The last part of this paper evaluates our methods on some of such applications, including cryptography, sensor localization, and differential equations.

We summarize our contributions as follows:

- We present a new elimination algorithm that exploits chordal structure in systems of polynomial equations. This method is presented in Algorithm 2. To the best of our knowledge, this is the first work that exploits chordal structure in computational algebraic geometry, as we will argue below in the section on related work.
- We prove that the chordal elimination algorithm computes the correct elimination ideals for a large family of problems, although (as explained in section 3) in general it may fail to do so. In particular, Lemma 4 specifies conditions under which chordal elimination succeeds. We show in Theorem 3 that these conditions are met for a large class of problems. Among others, this class includes linear equations and generic dense ideals.
- We present a recursive method (Algorithm 3) to compute the elimination ideals of all maximal cliques of the graph. These ideals provide a good sparse description from which we can easily find all solutions, as seen in section 5.2. We show in Corollary 4 that this algorithm succeeds under the same conditions of chordal elimination.
- We show in Theorem 6 and Corollary 6 that the complexity of our methods is linear in the number of variables and exponential in the treewidth for a restricted class of problems.

- Section 7 provides experimental evaluation of our methods in problems from graph colorings, cryptography, sensor localization, and differential equations. In all of these cases we show the advantages of chordal elimination over standard Gröbner basis algorithms. In some cases, we show that we can also find a lex Gröbner basis faster than with degrevlex ordering by using chordal elimination. This subverts the heuristic of preferring degrevlex.

The paper is structured as follows. In section 2 we provide a brief introduction to chordal graphs and recall some ideas from algebraic geometry. In section 3 we present our main method, chordal elimination. Section 4 presents some types of systems under which chordal elimination succeeds. In section 5 we present a method for finding the elimination ideals of all maximal cliques of the graph. In section 6 we analyze the computational complexity of the algorithms proposed for a certain class of problems. Finally, section 7 presents an experimental evaluation of our methods.

**Related work.** Even though there is a broad literature regarding chordality/treewidth and also polynomial system solving, their interaction has received almost no attention. A meeting point between these two areas is the case of linear equations, for which graph modeling methods have been very successful. There also has been much research done on connections between graph theory and computational/commutative algebra. We now proceed to review previous works in these areas, comparing them with our methods.

*Chordality and bounded treewidth.* The concepts of chordality and bounded treewidth are pervasive in many different research areas. In fact, several hard graph problems (e.g., vertex colorings, vertex covers, weighted independent set) can be solved efficiently in chordal graphs and in graphs of bounded treewidth [6, 26]. In a similar way, many problems in constraint satisfaction and graphical models become polynomial-time solvable under bounded treewidth assumptions [11, 16]. In other words, some hard problems are fixed-parameter-tractable when they are parametrized by the treewidth.

The logic community has also studied families of graph problems which are fixed-parameter-tractable with respect to the treewidth [9]. Makowsky and Meer applied these methods to algebraic problems such as evaluation, feasibility, and positivity of polynomials [33]. They show that these problems are tractable under bounded treewidth and finite domain conditions. On the contrary, our methods do not require a discrete domain, as they rely on well-studied tools from computational algebraic geometry. Moreover, their methods are not implementable due to the large underlying constants.

*Chordality in linear algebra.* The use of graph theory methods in sparse linear algebra goes back at least to the work of Parter [35]. It was soon realized that symmetric Gaussian elimination (Cholesky factorization) does not introduce additional nonzero entries if and only if the adjacency graph of the matrix is chordal. Current numerical linear algebra methods exploit this property by first finding a small chordal completion of this adjacency graph [37]. The nonsymmetric case is quite a bit more complicated, but a standard approach is to use instead a chordal completion of the adjacency graph of $A^TA$ [12, 37]. In this paper we generalize these ideas to the case of nonlinear equations. We note that chordality is also used in several sparse matrix problems from optimization, such as matrix inversion, positive semidefinite matrix completion, and Hessian evaluation [36, 43].

*Structured polynomials.* Solving structured systems of polynomial equations is a well-studied problem in computational algebraic geometry. Many past techniques

make use of different types of structure. In particular, properties such as symmetry [20, 25] and multihomogeneous structure [21] have been exploited within the Gröbner basis framework. Symmetry and multihomogeneous structure have also been used in homotopy continuation methods; see, e.g., [40].

Sparsity in the equations has also been exploited by making use of polytopal abstractions of the system [42]. This idea has led to faster algorithms based on homotopy methods [29, 32], sparse resultants [18], and, more recently, Gröbner bases [22]. All of these methods will perform efficiently provided that a certain measure of complexity of the system, known as the BKK bound, is small. Nonetheless, these types of methods do not take advantage of the chordal structure that we study in this paper. Indeed, we will see that our methods may perform efficiently even when the number of solutions, and thus the BKK bound, is very large.

A different body of methods comes from the algebraic cryptanalysis community, which considers very sparse equations over small finite fields. One popular approach is converting the problem into a Boolean satisfiability problem (SAT) problem and using SAT solvers [2]. A different idea is seen in [38], where they represent each equation with its zero set and treat it as a constraint satisfaction problem (CSP). These methods implicitly exploit the graph structure of the system as both SAT and CSP solvers can take advantage of it. Our work, on the other hand, directly relates the graph structure with the algebraic properties of the ideal. In addition, our methods apply to positive dimensional systems and arbitrary fields.

*Graphs in computer algebra.* There is a long-standing interaction between graph theory and computational algebra. Indeed, several polynomial ideals have been associated to graphs in the past years [3,13,27,44]. One of the earliest such ideals was the coloring ideal [3, 13, 28], which was recently considered for the special case of chordal graphs [14]. The edge ideal is perhaps the most widely studied example [44,45], given its tight connections with simplicial complexes, and the many graph properties that can be inferred from the ideal (e.g., connectedness, acyclicity, colorability, chordality). More recently, the related binomial edge ideals have also motivated much research [27].

These graph ideals allow us to infer combinatorial properties by means of commutative/computational algebra, and they are also crucial in understanding the complexity of computational algebra problems. However, previous work has mostly focused on structural properties of these specific families of ideals, as opposed to effective methods for general polynomials, such as those from sparse linear algebra. In contrast, our paper uses graph theoretic methods as a constructive guide to perform computations on arbitrary sparse polynomial systems.

## 2. Preliminaries.

### 2.1. Chordal graphs.

Chordal graphs, also known as triangulated graphs, have many equivalent characterizations. A good presentation is found in [5]. For our purposes, we use the following definition.

DEFINITION 1. *Let $G$ be a graph with vertices $x_0, \dots, x_{n-1}$. An ordering of its vertices $x_0 > x_1 > \cdots > x_{n-1}$ is a* perfect elimination ordering *if for each $x_l$ the set*

$$X_l := \{x_l\} \cup \{x_m : x_m \text{ is adjacent to } x_l,\ x_m < x_l\} \tag{1}$$

*is such that the restriction $G|_{X_l}$ is a clique. A graph $G$ is* chordal *if it has a perfect elimination ordering.*

*Remark.* Observe that lower indices correspond to larger vertices.

Chordal graphs have many interesting properties. Observe, for instance, that the number of maximal cliques is at most $n$. The reason is because any clique should be contained in some $X_l$. It is easy to see that trees are chordal graphs: by successively pruning a leaf from the tree, we get a perfect elimination ordering.

Given a chordal graph $G$, a perfect elimination ordering can be found in linear time. A classic and simple algorithm for doing so is *maximum cardinality search* (MCS) [5]. This algorithm successively selects a vertex with a maximal number of neighbors among previously chosen vertices, as shown in Algorithm 1. The ordering obtained is a reversed perfect elimination ordering.

**Algorithm 1** Maximum cardinality search [5].

**Input:** A chordal graph $G = (V, E)$ and an optional initial clique
**Output:** A reversed perfect elimination ordering $\sigma$

```
1: procedure MCS(G, start = ∅)
2:     σ := start
3:     while |σ| < n do
4:         choose v ∈ V − σ that maximizes |adj(v) ∩ σ|
5:         append v to σ
6:     return σ
```

DEFINITION 2. *Let $G$ be an arbitrary graph. We say that $\overline{G}$ is a* chordal completion *of $G$ if it is chordal and $G$ is a subgraph of $\overline{G}$. The* clique number *of $\overline{G}$ is the size of its largest clique. The* treewidth *of $G$ is the minimum clique number of $\overline{G}$ (minus one) among all possible chordal completions.*

Observe that given any ordering $x_0 > \cdots > x_{n-1}$ of the vertices of $G$, there is a natural chordal completion $\overline{G}$; i.e., we add edges to $G$ in such a way that each $G|_{X_l}$ is a clique. In general, we want to find a chordal completion with a small clique number. However, there are $n!$ possible orderings of the vertices, and thus finding the best chordal completion is not simple. Indeed, this problem is NP-hard [1], but there are good heuristics and approximation algorithms [6].

*Example* 3. Let $G$ be the blue/solid graph in Figure 1. This graph is not chordal, but if we add the three green/dashed edges shown in the figure, we obtain a chordal completion $\overline{G}$. In fact, the ordering $x_0 > \cdots > x_9$ is a perfect elimination ordering of the chordal completion. The clique number of $\overline{G}$ is four and the treewidth of $G$ is three.

As mentioned in the introduction, we will assume throughout this paper that the graph $G = G(F)$ is chordal and the ordering of its vertices (inherited from the polynomial ring) is a perfect elimination ordering. However, for a nonchordal graph $G$ the same results hold by considering a chordal completion.

*Remark* (the linear case). We finalize this section by explaining how for linear equations finding a chordal completion of $G(F)$ agrees with standard methods from numerical linear algebra. A linear set of equations $F$ can be written in matrix form as $Ax = b$. This is equivalent to $(A^T A)x = A^T b$, which can be solved with a Cholesky factorization. As mentioned earlier, to minimize the *fill-in* (nonzero entries) we need to find a small chordal completion of the adjacency graph $G$ of matrix $A^T A$. It can be seen that this adjacency graph $G$ coincides with the graph $G(F)$ that we associate to the equations. Alternatively, we can directly perform an LU decomposition (Gaussian elimination) on matrix $A$, and it turns out that the adjacency graph $G$ also bounds the fill-in of the LU factors [12, 37].

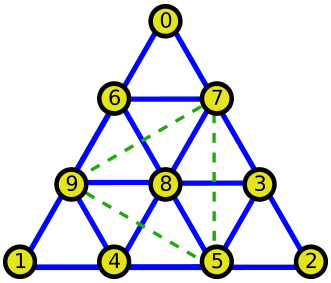


FIG. 1. 10-*vertex graph (blue, solid) and a chordal completion (green, dashed).*

**2.2. Algebraic geometry.** We use standard tools from computational algebraic geometry, following the notation from [10]. In particular, we assume the reader has familiarity with Gröbner bases, elimination ideals, and resultants.

We let $\text{elim}_l(I)$ be the $l$th elimination ideal, i.e.,

$$\text{elim}_l(I) := I \cap \mathbb{K}[x_l, \ldots, x_{n-1}].$$

We will denote by $I_l$ the "approximation" that we will compute to this elimination ideal, defined in section 3. We also denote by $\pi_l : \mathbb{K}^n \to \mathbb{K}^{n-l}$ the projection onto the last $n - l$ coordinates.

We recall the correspondence between elimination and projection given by

$$\mathbf{V}(\text{elim}_l(I)) = \overline{\pi_l(\mathbf{V}(I))},$$

where $\overline{S}$ denotes the closure of $S$ with respect to the Zariski topology.

*Remark.* In order for the above equation to hold, we require that $\mathbb{K}$ be algebraically closed, as we assume throughout this paper. However, if the coefficients of the equations $F$ are contained in a smaller field (e.g., $\mathbb{K} = \mathbb{C}$ but the coefficients are in $\mathbb{Q}$), then all computations in our algorithms will stay within such a field.

**3. Chordal elimination.** In this section, we present our main method, chordal elimination. As mentioned, we attempt to compute some generators for the elimination ideals with the same structure $G$. The approach we follow mimics the Gaussian elimination process by isolating the polynomials that do not involve the variables that we are eliminating. The output of chordal elimination is an "approximate" elimination ideal that preserves chordality. We call it approximate in the sense that, in general, it might not be the exact elimination ideal, but we hope it will be close to it. In fact, we will find inner and outer approximations to the ideal, as will be seen later. The case when both approximations are the same ensures us that the elimination ideal was computed correctly.

**3.1. Incremental elimination.** We follow an incremental approach to compute the elimination ideals, in a similar way as in Gaussian elimination. We illustrate the basic methodology through the following example.

*Example* 4. Consider the ideal

$$I = \langle x_0^4 - 1, x_0^2 + x_2, x_1^2 + x_2, x_2^2 + x_3 \rangle.$$

The associated graph is the tree in Figure 2. We incrementally eliminate each of the

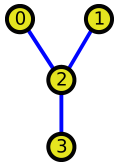


FIG. 2. *Simple* 3*-vertex tree.*

variables, considering at each step only the equations involving it. First, we consider only the polynomials involving $x_0$, of which there are two: $x_0^4 - 1, x_0^2 + x_2$. If we eliminate $x_0$ from these equations, we obtain $x_2^2 - 1$. This leads to the elimination ideal

$$I_1 = \langle x_1^2 + x_2, x_2^2 - 1, x_2^2 + x_3 \rangle.$$

We now consider the polynomials involving $x_1$, of which there is only one: $x_1^2 + x_2$. Thus, we cannot eliminate $x_1$, so our second elimination ideal is

$$I_2 = \langle x_2^2 - 1, x_2^2 + x_3 \rangle.$$

Finally, we eliminate $x_2$ from the remaining equations, obtaining

$$I_3 = \langle x_3 + 1 \rangle.$$

For this example all elimination ideals found are correct, as can be seen from the lex Gröbner basis $gb = \{x_0^2 + x_2, x_1^2 + x_2, x_2^2 - 1, x_3 + 1\}$.

Example 4 shows the basic idea we follow. Namely, to eliminate a variable $x_i$ we only consider a subset of the equations. In the above example, these equations only involved two variables at each step. In general, to eliminate $x_i$ we only take into account its neighboring variables in the graph. Therefore, if the neighborhood of each $x_i$ is small, we should require less computation. The chordality property will imply that these neighborhoods (cliques) are never expanded in the process.

This successive elimination process is simple, but it is not clear whether it always leads to the correct elimination ideals. The following example illustrates that this is not always the case.

*Example* 5 (incremental elimination may fail). Consider the ideal

$$I = \langle x_0 x_1 + 1, x_1 + x_2, x_1 x_2 \rangle.$$

The associated graph is the path $x_0$—$x_1$—$x_2$. We proceed in an incremental way as before. First, we consider only the polynomials involving $x_0$, of which there is only one: $x_0 x_1 + 1$. Thus, we cannot eliminate $x_0$, and we are left with the ideal

$$I_1 = \langle x_1 + x_2, x_1 x_2 \rangle.$$

Eliminating $x_1$ from the two equations above, we obtain

$$I_2 = \langle x_2^2 \rangle.$$

Observe that the original ideal $I$ is infeasible, i.e., $I = \langle 1 \rangle$, but the ideals $I_1, I_2$ found are feasible. Thus, the elimination ideals found are not correct.

Examples 4 and 5 show this incremental approach to obtain elimination ideals. In the first case the elimination process was correct, but in the second case it was not correct. The problem in the second example can be seen in the following equation:

$$\mathrm{elim}_1(\langle x_0x_1+1, x_1+x_2, x_1x_2\rangle) \neq \mathrm{elim}_1(\langle x_0x_1+1\rangle) + \langle x_1+x_2, x_1x_2\rangle.$$

The goal now is to understand why these ideals are different, and to determine when we can ensure that we successfully found the elimination ideals.

**3.2. Bounding the first elimination ideal.** We just introduced an incremental approach to compute elimination ideals, and we observed that it might not be correct. As will be shown next, the result of this process is always an inner approximation to the actual elimination ideal. Even more, we will see that we can also find an outer approximation to it. By comparing these approximations (or bounds) we can certify the cases where the elimination is correct. We now analyze the case of the first elimination ideal, and we will later proceed to further elimination ideals.

We formalize the elimination procedure presented in section 3.1. Let $I_1$ be our estimation of the first elimination ideal as described before. Recall that to compute the ideal $I_1$ we want to use only a subset of the equations, that is, in the examples above, those containing variable $x_0$. Let us denote by $J$ the ideal of these equations and by $K$ the ideal of the remaining equations. Then $I = J + K$, and our estimation of the first elimination ideal is given by $I_1 = \mathrm{elim}_1(J) + K$. Note that the equations of $I$ involving $x_0$ must certainly be part of $J$.

In this way, to compute $I_1$ we need only perform operations on the generators of $J$; we never deal with $K$. As a result, the computation of $I_1$ can be done on a smaller ring, whose variables correspond to a neighborhood, or clique, of the chordal graph. Chordality will ensure that graphical structure of $I$ is preserved; i.e., the graph associated to (the generators of) $I_1$ is a subgraph of $G$. We elaborate more on this later.

We want to show the relationship between our estimate $I_1$ and the actual elimination ideal $\mathrm{elim}_1(I)$. To do so, the key will be the closure theorem [10, Chapter 3].

DEFINITION 3. *Let $1 \leq l < n$, and let $I = \langle f_1, \ldots, f_s\rangle \subseteq \mathbb{K}[x_{l-1}, \ldots, x_{n-1}]$ be an ideal with a fixed set of generators. For each $1 \leq t \leq s$ assume that $f_t$ is of the form*

$$f_t = u_t(x_l, \ldots, x_{n-1})x_{l-1}^{d_t} + (\textit{terms with smaller degree in } x_{l-1})$$

*for some $d_t \geq 0$ and $u_t \in \mathbb{K}[x_l, \ldots, x_{n-1}]$. We define the* coefficient ideal *of $I$ to be*

$$\mathrm{coeff}_l(I) := \langle u_t : 1 \leq t \leq s\rangle \subseteq \mathbb{K}[x_l, \ldots, x_{n-1}].$$

THEOREM 1 (closure theorem). *Let $I = \langle f_1, \ldots, f_s\rangle \subseteq \mathbb{K}[x_0, \ldots, x_{n-1}]$. Let $W := \mathrm{coeff}_1(I)$ be the coefficient ideal, let $\mathrm{elim}_1(I)$ be the first elimination ideal, and let $\pi : \mathbb{K}^n \to \mathbb{K}^{n-1}$ be the projection onto the last factor. Then,*

$$\mathbf{V}(\mathrm{elim}_1(I)) = \overline{\pi(\mathbf{V}(I))},$$
$$\mathbf{V}(\mathrm{elim}_1(I)) - \mathbf{V}(W) \subseteq \pi(\mathbf{V}(I)).$$

The next lemma tells us that $I_1$ is an inner approximation of the actual elimination ideal $\mathrm{elim}_1(I)$. It also describes an outer approximation to it, which depends on $I_1$ and some ideal $W$. If the two bounds are equal, this implies that we successfully found the elimination ideal.

LEMMA 1. *Let* $J = \langle f_1, \ldots, f_s \rangle \subseteq \mathbb{K}[x_0, \ldots, x_{n-1}]$, $K = \langle g_1, \ldots, g_r \rangle \subseteq \mathbb{K}[x_1, \ldots, x_{n-1}]$, *and*

$$I := J + K = \langle f_1, \ldots, f_s, g_1, \ldots, g_r \rangle.$$

*Let the ideals* $I_1, W \subseteq \mathbb{K}[x_1, \ldots, x_{n-1}]$ *be*

$$\begin{aligned} I_1 &:= \operatorname{elim}_1(J) + K, \\ W &:= \operatorname{coeff}_1(J) + K. \end{aligned}$$

*Then the following equations hold:*

$$\mathbf{V}(\operatorname{elim}_1(I)) = \overline{\pi(\mathbf{V}(I))} \subseteq \mathbf{V}(I_1), \tag{2}$$

$$\mathbf{V}(I_1) - \mathbf{V}(W) \subseteq \pi(\mathbf{V}(I)). \tag{3}$$

*Proof.* We first show (2). The closure theorem says that $\mathbf{V}(\operatorname{elim}_1(I)) = \overline{\pi(\mathbf{V}(I))}$. We will show that $\pi(\mathbf{V}(I)) \subseteq \mathbf{V}(I_1)$, from which (2) follows because $\mathbf{V}(I_1)$ is closed.

In the following equations, sometimes we will consider the varieties in $\mathbb{K}^n$ and sometimes in $\mathbb{K}^{n-1}$. To specify, we will denote them as $\mathbf{V}^n$ and $\mathbf{V}^{n-1}$, respectively. Notice that

$$\pi(\mathbf{V}^n(I)) = \pi(\mathbf{V}^n(J + K)) = \pi(\mathbf{V}^n(J) \cap \mathbf{V}^n(K)).$$

Now observe that

$$\pi(\mathbf{V}^n(J) \cap \mathbf{V}^n(K)) = \pi(\mathbf{V}^n(J)) \cap \mathbf{V}^{n-1}(K).$$

The reason is the fact that if $S \subseteq \mathbb{K}^n$, $T \subseteq \mathbb{K}^{n-1}$ are arbitrary sets, then

$$\pi(S \cap (\mathbb{K} \times T)) = \pi(S) \cap T.$$

Finally, note that $\overline{\pi(\mathbf{V}^n(J))} = \mathbf{V}(\operatorname{elim}_1(J))$. Combining everything, we conclude that

$$\begin{aligned} \pi(\mathbf{V}(I)) &= \pi(\mathbf{V}^n(J)) \cap \mathbf{V}^{n-1}(K) \\ &\subseteq \overline{\pi(\mathbf{V}^n(J))} \cap \mathbf{V}^{n-1}(K) \\ &= \mathbf{V}^{n-1}(\operatorname{elim}_1(J)) \cap \mathbf{V}^{n-1}(K) \\ &= \mathbf{V}^{n-1}(\operatorname{elim}_1(J) + K) \\ &= \mathbf{V}(I_1). \end{aligned}$$

We now show (3). The closure theorem states that

$$\mathbf{V}(\operatorname{elim}_1(J)) - \mathbf{V}(\operatorname{coeff}_1(J)) \subseteq \pi(\mathbf{V}(J)).$$

Then,

$$\begin{aligned} \mathbf{V}(I_1) - \mathbf{V}(W) &= [\mathbf{V}^{n-1}(\operatorname{elim}_1(J)) \cap \mathbf{V}^{n-1}(K)] - [\mathbf{V}^{n-1}(\operatorname{coeff}_1(J)) \cap \mathbf{V}^{n-1}(K)] \\ &= [\mathbf{V}^{n-1}(\operatorname{elim}_1(J)) - \mathbf{V}^{n-1}(\operatorname{coeff}_1(J))] \cap \mathbf{V}^{n-1}(K) \\ &\subseteq \pi(\mathbf{V}^n(J)) \cap \mathbf{V}^{n-1}(K) \\ &= \pi(\mathbf{V}(I)). \end{aligned}$$

This concludes the proof. ∎

Note that the lemma above implies the equations

$$\mathbf{V}(I_1) - \mathbf{V}(W) \subseteq \mathbf{V}(\mathrm{elim}_1(I)) \subseteq \mathbf{V}(I_1),$$
$$\sqrt{I_1} : \sqrt{W} \supseteq \sqrt{\mathrm{elim}_1(I)} \supseteq \sqrt{I_1},$$

where we used the fact that set difference of varieties corresponds to ideal quotient. Thus, the ideal $W$ bounds the approximation error of our estimation $I_1$ to the ideal $\mathrm{elim}_1(I)$. In particular, if $\mathbf{V}(W)$ is empty, then $I_1$ and $\mathrm{elim}_1(I)$ determine the same variety.

**3.3. Bounding all elimination ideals.** Lemma 1 gave us the relationship between our estimation $I_1$ and the actual elimination ideal $\mathrm{elim}_1(I)$. We generalize this now to further elimination ideals.

We denote by $I_l$ our estimation of the $l$th elimination ideal. As before, to estimate $\mathrm{elim}_{l+1}(I_l)$ we only use a subset of the equations of $I_l$, which we denote as $J_l$. The remaining equations are denoted as $K_{l+1}$. Then $I_{l+1} = \mathrm{elim}_{l+1}(I_l) + K_{l+1}$. The following theorem establishes the relationship between $I_{l+1}$ and $\mathrm{elim}_{l+1}(I)$.

THEOREM 2. *Let $I \subseteq \mathbb{K}[x_0, \ldots, x_{n-1}]$ be an ideal. Consider ideals $I_l \subseteq \mathbb{K}[x_l, \ldots, x_{n-1}]$ for $0 \leq l < n$, with $I_0 := I$, which are constructed recursively as follows:*

(i) *Given $I_l$, let $J_l \subseteq \mathbb{K}[x_l, \ldots, x_{n-1}]$, $K_{l+1} \subseteq \mathbb{K}[x_{l+1}, \ldots, x_{n-1}]$ be*[2] *such that $I_l = J_l + K_{l+1}$.*

(ii) *Let $I_{l+1} := \mathrm{elim}_{l+1}(J_l) + K_{l+1}$.*

(iii) *Also denote $W_{l+1} := \mathrm{coeff}_{l+1}(J_l) + K_{l+1}$.*

*Then for each $l$ the following equations hold:*

$$\mathbf{V}(\mathrm{elim}_l(I)) = \overline{\pi_l(\mathbf{V}(I))} \subseteq \mathbf{V}(I_l), \tag{4}$$

$$\mathbf{V}(I_l) - [\pi_l(\mathbf{V}(W_1)) \cup \cdots \cup \pi_l(\mathbf{V}(W_l))] \subseteq \pi_l(\mathbf{V}(I)). \tag{5}$$

*Proof.* The proof follows from Lemma 1 by induction. See section A.1 in the appendix. ∎

The lemma above implies the following equations:

$$\mathbf{V}(I_L) - \mathbf{V}(W) \subseteq \mathbf{V}(\mathrm{elim}_L(I)) \subseteq \mathbf{V}(I_L), \tag{6a}$$

$$\sqrt{I_L} : \sqrt{W} \supseteq \sqrt{\mathrm{elim}_L(I)} \supseteq \sqrt{I_L}, \tag{6b}$$

where the ideal $W$ is

$$W := \mathrm{elim}_L(W_1) \cap \cdots \cap \mathrm{elim}_L(W_L). \tag{7}$$

Note also that by construction we always have that if $x_m < x_l$, then $I_m \subseteq I_l$.

**3.4. Chordal elimination algorithm.** The recursive construction given in Theorem 2 is not fully specified (see item (i)). In particular, it is not clear which decomposition of type $I_l = J_l + K_{l+1}$ to use. We now describe the specific decomposition we use to obtain the chordal elimination algorithm.

We recall the definition of the cliques $X_l$ from (1). Equivalently, $X_l$ is the largest clique containing $x_l$ in $G|_{\{x_l, \ldots, x_{n-1}\}}$. Let $f_j$ be a generator of $I_l$. If all of the variables in $f_j$ are contained in $X_l$, we put $f_j$ in $J_l$. Otherwise, if some variable of $f_j$ is not in $X_l$, we put $f_j$ in $K_{l+1}$. We refer to this procedure as *clique decomposition*.

[2] Note that this decomposition is not unique, since we are not fully specifying the ideals $J_l, K_{l+1}$.

*Example* 6. Let $I = \langle f, g, h\rangle$, where $f = x_0^2 + x_1x_2$, $g = x_1^3 + x_2$, and $h = x_1 + x_3$. Note that the associated graph consists of a triangle $x_0, x_1, x_2$ and the edge $x_1, x_3$. Thus, we have $X_0 = \{x_0, x_1, x_2\}$. The clique decomposition is $J_0 = \langle f, g\rangle$, $K_1 = \langle h\rangle$.

We should mention that this decomposition method is reminiscent of the bucket elimination algorithm from constraint satisfaction [15]. However, we do not place an equation $f$ in its largest variable, but rather in the largest variable $x_l$ such that $f \in \mathbb{K}[X_l]$. The reason for doing this is to further shrink the variety $\mathbf{V}(J_l)$. This leads to a tighter approximation of the elimination ideals and simplifies the Gröbner basis computation.

It is easy to see that the procedure in Theorem 2, using this clique decomposition, preserves chordality. We state that now.

PROPOSITION 1. *Let $I$ be an ideal with chordal graph $G$. If we follow the procedure in Theorem* 2 *using the clique decomposition, then the graph associated to $I_l$ is a subgraph of $G$.*

*Proof.* Observe that we do not modify the generators of $K_{l+1}$, and thus the only part where we may alter the sparsity pattern is when we compute $\text{elim}_{l+1}(J_l)$ and $\text{coeff}_{l+1}(J_l)$. However, the variables involved in $J_l$ are contained in the clique $X_l$ and thus, independent of which operations we apply to its generators, we will not alter the structure. ∎

In the above discussion we resolved the ambiguity problem of step (i) in Theorem 2. However, there is still an issue regarding the "error ideal" $W$ of (7). We recall that $W_{l+1}$ depends on the coefficient ideal of $J_l$. Thus, $W_{l+1}$ does not only depend on the ideal $J_l$, but it also depends on the specific set of generators that we are using. In particular, some set of generators might lead to a larger/worse variety $\mathbf{V}(W_{l+1})$ than others. This problem is inherent to the closure theorem, and it is discussed in [10, Chapter 3]. It turns out that a lex Gröbner basis of $J_l$ is an optimal set of generators, as shown in [10]. Therefore, it is convenient to find this Gröbner basis before computing the coefficient ideal.

Algorithm 2 presents the chordal elimination algorithm. The output of the algorithm is the inner approximation $I_L$ to the $L$th elimination ideal and the ideals $W_1, \ldots, W_L$ that satisfy (6). In the event that $\mathbf{V}(W_l) = \emptyset$ for all $l$, the elimination was correct. This is the case that we focus on in the rest of the paper.

*Remark.* Observe that in line 14 of Algorithm 2 we append a Gröbner basis to $J_l$, so that we do not remove the old generators. There are two reasons to compute this lex Gröbner basis: it allows us to find the $\text{elim}_{l+1}(J_l)$ easily, and we obtain a tighter $W_{l+1}$ as discussed above. However, we do not replace the old set of generators but instead append to them this Gröbner basis. We will explain the reason for doing this in section 3.5.

**3.5. Elimination tree.** We now introduce the concept of elimination tree and show its connection with chordal elimination. This concept will help us to analyze our methods.

DEFINITION 4. *Let $G$ be an ordered graph with vertex set $x_0 > \cdots > x_{n-1}$. We associate to $G$ the following* directed spanning tree *$T$ that we refer to as the* elimination tree*: For each $x_l > x_{n-1}$ there is an arc from $x_l$ towards the largest $x_p$ that is adjacent to $x_l$ and $x_p < x_l$. We will say that $x_p$ is* the parent *of $x_l$ and that $x_l$ is* a descendant *of $x_p$. Note that $T$ is rooted at $x_{n-1}$.*

Figure 3 shows an example of the elimination tree of a given graph. It is easy

**Algorithm 2** Chordal elimination to find the $L$th elimination ideal.

**Input:** An ideal $I$, given by generators with chordal graph $G$, and an integer $L$
**Output:** Ideals $I_L$ and $W_1, \dots, W_L$ approximating $\text{elim}_L(I)$ as in (6)

**procedure** ChordElim$(I, G, L)$
  $I_0 = I$
  **for** $l = 0 : L-1$ **do**
    get clique $X_l$ of $G$
    $J_l, K_{l+1} =$ SplitGens$(I_l, X_l)$
    FindElim&Coeff$(J_l)$
    $I_{l+1} = \text{elim}_{l+1}(J_l) + K_{l+1}$
    $W_{l+1} = \text{coeff}_{l+1}(J_l) + K_{l+1}$
  **return** $I_L, W_1, \dots, W_L$

**procedure** SplitGens$(I_l, X_l)$ ▷ Partition generators of $I_l$
  $J_l = \langle f : f \text{ generator of } I_l \text{ and } f \in \mathbb{K}[X_l] \rangle$
  $K_{l+1} = \langle f : f \text{ generator of } I_l \text{ and } f \notin \mathbb{K}[X_l] \rangle$

**procedure** FindElim&Coeff$(J_l)$ ▷ Eliminate $x_l$ in the ring $\mathbb{K}[X_l]$
  append to $J_l$ its lex Gröbner basis
  $\text{elim}_{l+1}(J_l) = \langle f : f \text{ generator of } J_l \text{ with no } x_l \rangle$
  $\text{coeff}_{l+1}(J_l) = \langle \text{leading coefficient of } f : f \text{ generator of } J_l \rangle$

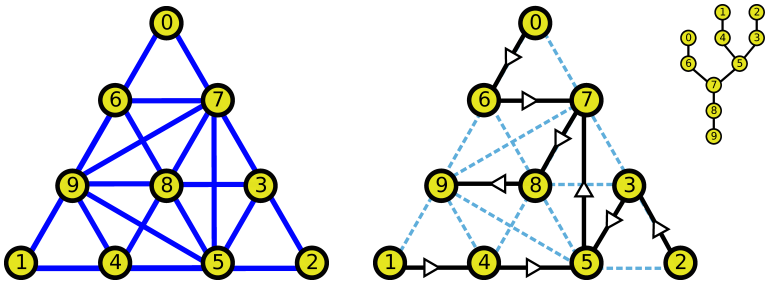


Fig. 3. *Chordal graph $G$ and its elimination tree $T$.*

to see that eliminating a variable $x_l$ corresponds to pruning one of the leaves of the elimination tree. We now present a simple property of such a tree.

Lemma 2. *Let $G$ be a chordal graph, let $x_l$ be some vertex, and let $x_p$ be its parent in the elimination tree $T$. Then,*

$$X_l \setminus \{x_l\} \subseteq X_p,$$

*where $X_i$ is as in* (1).

*Proof.* Let $C = X_l \setminus \{x_l\}$. Note that $C$ is a clique that contains $x_p$. Even more, $x_p$ is its largest variable because of the definition of $T$. As $X_p$ is the unique largest clique satisfying such a property, we must have $C \subseteq X_p$. ∎

A consequence of the lemma above is the following relation:

$$\text{elim}_{l+1}(I \cap \mathbb{K}[X_l]) \subseteq I \cap \mathbb{K}[X_p], \tag{8}$$

where $I \cap \mathbb{K}[X_l]$ is the set of all polynomials in $I$ that involve only variables in $X_l$.

The reason for the relation above is

$$\text{elim}_{l+1}(I \cap \mathbb{K}[X_l]) = (I \cap \mathbb{K}[X_l]) \cap \mathbb{K}[x_{l+1}, \ldots, x_{n-1}] = I \cap \mathbb{K}[X_l \setminus \{x_l\}].$$

There is a simple geometric interpretation of (8). The variety $\mathbf{V}(I \cap \mathbb{K}[X_l])$ can be interpreted as the set of partial solutions restricted to the set $X_l$. Thus, (8) is telling us that any partial solution on $X_p$ extends to a partial solution on $X_l$ (the inclusion is reversed). Even though this equation is very simple, this is a property that we would like to keep in chordal elimination.

Clearly, we do not have a representation of the clique elimination ideal $I \cap \mathbb{K}[X_l]$. However, the natural relaxation to consider is the ideal $J_l \subseteq \mathbb{K}[X_l]$ that we compute in Algorithm 2. To preserve the property above (i.e., every partial solution of $X_p$ extends to $X_l$), we would like to have the following relation:

$$\text{elim}_{l+1}(J_l) \subseteq J_p. \tag{9}$$

It turns out that there is a very simple way to ensure this property: we preserve the old generators of the ideal during the elimination process. This is precisely the reason why in line 14 of Algorithm 2 we *append* a Gröbner basis to $J_l$.

We now prove that (9) holds. We need the following lemma.

LEMMA 3. *In Algorithm* 2*, let $f \in I_l$ be one of its generators. If $x_m$ is such that $x_m \leq x_l$ and $f \in \mathbb{K}[X_m]$, then $f$ is a generator of $J_m$. In particular, this holds if $x_m$ is the largest variable in $f$.*

*Proof.* For a fixed $x_m$, we will show this by induction on $x_l$.

The base case is $l = m$. In such a case, by construction of $J_m$ we have that $f \in J_m$.

Assume now that the assertion holds for any $x_l'$ with $x_m \leq x_{l'} < x_l$, and let $f$ be a generator of $I_l$. There are two cases: either $f \in J_l$ or $f \in K_{l+1}$. In the second case, $f$ is a generator of $I_{l+1}$, and using the induction hypothesis we get $f \in J_m$. In the first case, as $f \in \mathbb{K}[X_m]$, all variables of $f$ are less than or equal to $x_m$ and thus strictly smaller than $x_l$. Following Algorithm 2, we see that $f$ is a generator of $\text{elim}_{l+1}(J_l)$. Thus, $f$ is again a generator of $I_{l+1}$, and we conclude by induction.

We now prove the second part; i.e., it holds if $x_m$ is the largest variable. We just need to show that $f \in \mathbb{K}[X_m]$. Let $X_{lm} := X_l \cap \{x_m, \ldots, x_{n-1}\}$; then $f \in \mathbb{K}[X_{lm}]$, as $x_m$ is the largest variable. Note that as $f \in \mathbb{K}[X_l]$ and $f$ involves $x_m$, we have $x_m \in X_l$. Thus, $X_{lm}$ is a clique of $G|_{\{x_m,\ldots,x_{n-1}\}}$ and contains $x_m$. However, $X_m$ is the unique largest clique that satisfies this property. Then $X_{lm} \subseteq X_m$ so that $f \in \mathbb{K}[X_{lm}] \subseteq \mathbb{K}[X_m]$. ∎

COROLLARY 1. *Let $x_l$ be arbitrary, and let $x_p$ be its parent in the elimination tree $T$. Then* (9) *holds for Algorithm* 2*.*

*Proof.* Let $f \in \text{elim}_{l+1}(J_l)$ be one of its generators. Then $f$ is also one of the generators of $I_{l+1}$ by construction. It is clear that the variables of $f$ are contained in $X_l \setminus \{x_l\} \subseteq X_p$, where we used Lemma 2. From Lemma 3 we get that $f \in J_p$, concluding the proof. ∎

The reader may believe that preserving the old set of generators is not necessary. The following example shows that it is necessary in order to have the relation in (9).

*Example* 7. Consider the ideal

$$I = \langle x_0 - x_2, x_0 - x_3, x_1 - x_3, x_1 - x_4, x_2 - x_3, x_3 - x_4, x_2^2 \rangle,$$

whose associated graph consists of two triangles $\{x_0, x_2, x_3\}$ and $\{x_1, x_3, x_4\}$. Note that the parent of $x_0$ is $x_2$. If we preserve the old generators (as in Algorithm 2), we get $I_2 = \langle x_2 - x_3, x_3 - x_4, x_2^2, x_3^2, x_4^2\rangle$. If we do not preserve them, we get instead $\hat{I}_2 = \langle x_2 - x_3, x_3 - x_4, x_4^2\rangle$. In the last case we have $\hat{J}_2 = \langle x_2 - x_3\rangle$ so that $x_2^2 \notin \hat{J}_2$, even though $x_2^2 \in J_0$. Moreover, the ideal $J_0$ is zero-dimensional, but $\hat{J}_2$ has positive dimension. Thus, (9) does not hold.

## 4. Successful elimination.

NOTATION. *We will write* $I \overset{\text{rad}}{=} J$ *whenever we have* $\mathbf{V}(I) = \mathbf{V}(J)$.

In section 3 we showed an algorithm that gives us an approximate elimination ideal. In this section we are interested in finding conditions under which such an algorithm returns the actual elimination ideal. We will say that chordal elimination, i.e., Algorithm 2, *succeeds* if we have $\mathbf{V}(I_l) = \mathbf{V}(\text{elim}_l(I))$. Following the convention above, we write $I_l \overset{\text{rad}}{=} \text{elim}_l(I)$.

**4.1. The domination condition.** Theorem 2 gives us lower and upper bounds on the actual elimination ideals. We use these bounds to obtain a condition that guarantees that chordal elimination succeeds.

DEFINITION 5. *We say that a polynomial* $f$ *is* $x_i$-dominated *if its leading monomial has the form* $x_i^d$ *for some* $d$. *We say that an ideal* $J$ *is* $x_i$*-dominated if there is some* $f \in J$ *that is* $x_i$*-dominated. Equivalently,* $J$ *is* $x_i$*-dominated if its initial ideal* $\text{in}(J)$ *contains a pure power of* $x_i$.

DEFINITION 6 (domination condition). *Let* $I$ *be an ideal and use Algorithm* 2. *We say that the* domination condition *holds if* $J_l$ *is* $x_l$*-dominated for each* $l$.

The domination condition implies that chordal elimination succeeds, as shown in the next lemma.

LEMMA 4 (domination implies success). *If the domination condition holds, then* $I_l \overset{\text{rad}}{=} \text{elim}_l(I)$ *and the corresponding variety is* $\pi_l(\mathbf{V}(I))$ *for all* $l$.

*Proof.* As $J_l$ is $x_l$-dominated, its initial ideal $\text{in}(J_l)$ contains a pure power of $x_l$. Thus, there must be a $g$ that is part of the Gröbner basis of $J_l$ and is $x_l$-dominated. The coefficient $u_t$ that corresponds to such a $g$ is $u_t = 1$, and therefore $1 \in W_l$ and $\mathbf{V}(W_l) = \emptyset$. Thus, the two bounds in Theorem 2 are the same, and the result follows. ∎

We will now show some classes of ideals in which the domination condition holds. Using the previous lemma, this guarantees that chordal elimination succeeds.

COROLLARY 2. *Let* $I$ *be an ideal, and assume that for each* $l$ *such that* $X_l$ *is a maximal clique of* $G$, *the ideal* $J_l \subseteq \mathbb{K}[X_l]$ *is zero-dimensional. Then the domination condition holds and chordal elimination succeeds.*

*Proof.* Let $x_m$ be arbitrary, and let $x_l \geq x_m$ be such that $X_m \subseteq X_l$ and $X_l$ is a maximal clique. As $J_l \subseteq \mathbb{K}[X_l]$ is zero-dimensional, it is $x_j$-dominated for all $x_j \in X_l$. Thus, there is a $g$ that is part of the Gröbner basis of $J_l$ and is $x_m$-dominated. From Lemma 3 we obtain that $g \in J_m$, and thus the domination condition holds. ∎

COROLLARY 3. *Let* $I$ *be an ideal, and assume that for each* $l$ *there is a generator* $f_l$ *of* $I$ *that is* $x_l$*-dominated. Then the domination condition holds and chordal elimination succeeds.*

*Proof.* It follows from Lemma 3 that $f_l \in J_l$, so that $J_l$ is $x_l$-dominated and the domination condition holds. ∎

The previous corollary presents a first class of ideals for which we are guaranteed to have successful elimination. Note that when we solve equations over a finite field $\mathbb{F}_q$, usually we include equations of the form $x_l^q - x_l$, so the corollary holds. In particular, it holds for 0/1 problems.

**4.2. Simplicial equations.** The assumptions of Corollary 3 are too strong for many cases. In particular, if $l = n-2$, the only way that such an assumption holds is if there is a polynomial that only involves $x_{n-2}, x_{n-1}$. We will show now a larger class of ideals for which the domination condition also holds and thus chordal elimination succeeds. The following concept is the basis for this class.

DEFINITION 7. *Let $f \in \mathbb{K}[x_0, \ldots, x_{n-1}]$ be such that for each variable $x_l$ of positive degree, the monomial $m_l$ of $f$ with largest degree in $x_l$ is unique and has the form $m_l = x_l^{d_l}$ for some $d_l > 0$. We say that $f$ is* simplicial.

*Example* 8. Consider the polynomials of Example 5:

$$f_1 = x_0x_1 + 1, \qquad f_2 = x_1 + x_2, \qquad f_3 = x_1x_2.$$

Then $f_2$ is simplicial, as for both $x_1, x_2$ the monomials of largest degree in these variables are pure powers. In general, linear equations are always simplicial. On the other hand, $f_1, f_3$ are not simplicial. This makes sense, as we will see that if all polynomials are simplicial, then chordal elimination succeeds, which was not the case of Example 5. On the contrary, all the polynomials of Example 4 are simplicial.

Note that the definition of simplicial is independent of the monomial ordering used, as opposed to $x_i$-domination. The reason for the term *simplicial* is that the (scaled) standard simplex

$$\Delta = \left\{ x : x \geq 0, \sum_{x_l \in X_f} x_l/d_l = |X_f| \right\},$$

where $X_f$ are the variables of $f$, is a face of the Newton polytope of $f$, and it is the whole polytope if $f$ is homogeneous.

We will make an additional genericity assumption on the simplicial polynomials. Concretely, we assume that the coefficients of $m_l = x_l^{d_l}$ are generic, in a sense that will be clear in the next lemma.

LEMMA 5. *Let $q_1, q_2$ be generic simplicial polynomials. Let $X_1, X_2$ denote their sets of variables, and let $x \in X_1 \cap X_2$. Then $h = \mathrm{Res}_x(q_1, q_2)$ is generic simplicial, and its set of variables is $X_1 \cup X_2 \setminus x$.*

*Proof.* Let $q_1, q_2$ be of degree $m_1, m_2$ when viewed as univariate polynomials in $x$. As $q_2$ is simplicial, for each $x_i \in X_2 \setminus x$ the monomial with the largest degree in $x_i$ has the form $x_i^{d_2}$. It is easy to see that the largest monomial of $h$, as a function of $x_i$, that comes from $q_2$ will be $x_i^{d_2 m_1}$. Such a monomial arises from the product of the main diagonal of the Sylvester matrix. In the same way, the largest monomial that comes from $q_1$ has the form $x_i^{d_1 m_2}$. If $d_2 m_1 = d_1 m_2$, the genericity guarantees that such monomials do not cancel each other out. Thus, the leading monomial of $h$ in $x_i$ has the form $x_i^{\max\{d_2 m_1, d_1 m_2\}}$, and then $h$ is simplicial. The coefficients of the extreme monomials are polynomials in the coefficients of $q_1, q_2$, so if they were not generic (they satisfy certain polynomial equation), then $q_1, q_2$ would not be generic either. ∎

Observe that in the lemma above we required the coefficients to be generic in order to avoid cancellations in the resultant. This is the only part where we need this assumption.

We recall that elimination can be viewed as pruning the elimination tree $T$ of $G$ (Definition 4). We attach each of the generators of $I$ to some node of $T$. More precisely, we attach a generator $f$ to the largest variable it contains, which we denote as $x(f)$. The following lemma tells us that if there are many simplicial polynomials attached to the subtree of $T$ rooted in $x_l$, then $J_l$ is $x_l$-dominated.

LEMMA 6. *Let $I = \langle f_1, \ldots, f_s \rangle$ and let $1 \leq l < n$. Let $T_l$ be a subtree of $T$ with $t$ vertices and minimal vertex $x_l$. Assume that there are $f_{i_1}, \ldots, f_{i_t}$ generic simplicial with largest variable $x(f_{i_j}) \in T_l$ for $1 \leq j \leq t$. Then $J_l$ is $x_l$-dominated.*

*Proof.* We will show that we can find a simplicial polynomial $f_l \in J_l$ that contains $x_l$, which implies the desired result. Let us ignore all $f_t$ such that its largest variable is not in $T_l$. By doing this, we get smaller ideals $J_l$, so it does not help to prove the statement. Let us also ignore all vertices which do not involve one of the remaining equations. Let $S$ be the set of variables which are not in $T_l$. As in any of the remaining equations the largest variable should be in $T_l$, then for any $x_i \in S$ there is some $x_j \in T_l$ with $x_j > x_i$. We will show that for any $x_i \in S$ we have $x_l > x_i$.

Assume by contradiction that this is not true, and let $x_i$ be the smallest counterexample. Let $x_p$ be the parent of $x_i$. Note that $x_p \notin S$ because of the minimality of $x_i$, and thus $x_p \in T_l$. As mentioned earlier, there is some $x_j \in T_l$ with $x_j > x_i$. As $x_j > x_i$ and $x_p$ is the parent of $x_i$, this means that $x_i$ is in the path of $T$ that joins $x_j$ and $x_p$. However, $x_j, x_p \in T_l$ and $x_i \notin T_l$, so this contradicts that $T_l$ is connected.

Thus, for any $x_i \in S$, we have that $x_i < x_l$. This says that to obtain $J_l$ we do not need to eliminate any of the variables in $S$. Therefore, we can ignore all variables in $S$. Thus, we can assume that $l = n - 1$ and $T_l = T$. This reduces the problem to the specific case considered in the following lemma. ∎

LEMMA 7. *Let $I = \langle f_1, \ldots, f_n \rangle$ such that $f_j$ is generic simplicial for all $j$. Then there is a simplicial polynomial $f \in I_{n-1} = J_{n-1}$.*

*Proof.* We will prove the more general result: for each $l$ there exist $f_1^l, f_2^l, \ldots, f_{n-l}^l \in I_l$ which are all simplicial and generic. Moreover, we will show that if $x_j$ denotes the largest variable of some $f_i^l$, then $f_i^l \in J_j$. Note that as $x_j \leq x_l$, we have $J_j \subseteq I_j \subseteq I_l$. We will explicitly construct such polynomials.

Such a construction is very similar to the chordal elimination algorithm. The only difference is that instead of elimination ideals we use resultants.

Initially, we assign $f_i^0 = f_i$ for $1 \leq i \leq n$. Inductively, we construct the next polynomials,

$$f_i^{l+1} = \begin{cases} \mathrm{Res}_{x_l}(f_0^l, f_{i+1}^l) & \text{if } f_{i+1}^l \text{ involves } x_l, \\ f_{i+1}^l & \text{if } f_{i+1}^l \text{ does not involve } x_l, \end{cases}$$

for $1 \leq i \leq n - l$, where we assume that $f_0^l$ involves $x_l$, possibly after rearranging them. In the event that no $f_i^l$ involves $x_l$, we can ignore such a variable. Notice that Lemma 5 tells us that $f_i^l$ are all generic and simplicial.

We need to show that $f_i^l \in J_j$, where $x_j$ is the largest variable of $f_i^l$. We will prove this by induction on $l$.

The base case is $l = 0$, where $f_i^0 = f_i$ are generators of $I$, and thus Lemma 3 says that $f_i \in J_j$.

Assume that the hypothesis holds for some $l$ and consider some $f := f_i^{l+1}$. Let $x_j$ be its largest variable. Consider first the case where $f = f_{i+1}^l$. By the induction hypothesis, $f \in J_j$, and we are done.

Now consider the case when $f = \mathrm{Res}_{x_l}(f_0^l, f_{i+1}^l)$. In this case the largest variable of both $f_0^l, f_{i+1}^l$ is $x_l$ and thus, using the induction hypothesis, both of them lie in $J_l$. Let $x_p$ be the parent of $x_l$. Using (9) we get $f \in \mathrm{elim}_{l+1}(J_l) \subseteq J_p$. Let us see now that $x_j \leq x_p$. The reason is because $x_j \in X_p$, as $f \in \mathbb{K}[X_p]$ and $x_j$ is its largest variable. Thus, we found an $x_p$ with $x_j \leq x_p < x_l$ and $f \in J_p$. If $x_j = x_p$, we are done. Otherwise, if $x_j < x_p$, let $x_r$ be the parent of $x_p$. As $f$ does not involve $x_p$, we have $f \in \mathrm{elim}_{p+1}(J_p) \subseteq J_r$. In the same way as before we get that $x_j \leq x_r < x_p$ and $f \in J_r$. Note that we can repeat this argument again, until we get that $f \in J_j$. This concludes the induction. ∎

Lemma 6 can be used to show the domination condition and thus certify that chordal elimination succeeds. In particular, we can do this in the special case when all polynomials are simplicial, as we show in the next theorem.

THEOREM 3. *Let $I = \langle f_1, \ldots, f_s \rangle$ be an ideal such that for each $1 \leq i \leq s$, $f_i$ is generic simplicial. Then chordal elimination succeeds.*

*Proof.* For each $l$, let $T_l$ be the largest subtree of $T$ with minimal vertex $x_l$. Equivalently, $T_l$ consists of all the descendants of $x_l$. Let $t_l := |T_l|$, and let $x(f_j)$ denote the largest variable of $f_j$. If for all $x_l$ there are at least $t_l$ generators $f_j$ with $x(f_j) \in T_l$, then Lemma 6 implies the domination condition and we are done. Otherwise, let $x_l$ be the largest where this fails. The maximality of $x_l$ guarantees that elimination succeeds up to such a point, i.e., $I_m = \mathrm{elim}_m(I)$ for all $x_m \geq x_l$. We claim that no equation of $I_l$ involves $x_l$, and thus we can ignore it. Proving this claim will conclude the proof.

If $x_l$ is a leaf of $T$, then $t_l = 1$, which means that no generator of $I$ involves $x_l$. Otherwise, let $x_{s_1}, \ldots, x_{s_r}$ be its children. Note that $T_l = \{x_l\} \cup T_{s_1} \cup \cdots \cup T_{s_r}$. We know that there are at least $t_{s_i}$ generators with $x(f_j) \in T_{s_i}$ for each $s_i$, and such a bound has to be exact, as $x_l$ does not have such a property. Thus, for each $s_i$ there are exactly $t_{s_i}$ generators with $x(f_j) \in T_{s_i}$, and there is no generator with $x(f_j) = x_l$. Then, for each $s_i$, when we eliminate all the $t_{s_i}$ variables in $T_{s_i}$ in the corresponding $t_{s_i}$ equations we must get the zero ideal, i.e., $\mathrm{elim}_{s_i+1}(J_{s_i}) = 0$. On the other hand, as there is no generator with $x(f_j) = x_l$, all generators that involve $x_l$ are in some $T_{s_i}$. But we observed that the $l$th elimination ideal in each $T_{s_i}$ is zero, so that $I_l$ does not involve $x_l$, as we wanted. ∎

## 5. Elimination ideals of cliques.

NOTATION. *We will write $I \overset{\mathsf{rad}}{=} J$ whenever we have $\mathbf{V}(I) = \mathbf{V}(J)$.*

Algorithm 2 allows us to compute (or bound) the elimination ideals $I \cap \mathbb{K}[x_l, \ldots, x_{n-1}]$. In this section we will show that once we compute such ideals, we can also compute many other elimination ideals. In particular, we will compute the elimination ideals of the maximal cliques of $G$.

We recall the definition of the cliques $X_l$ from (1). Let $H_l := I \cap \mathbb{K}[X_l]$ be the corresponding elimination ideal. As any clique is contained in some $X_l$, we can restrict our attention to computing $H_l$. In particular, all maximal cliques of the graph are of the form $X_l$ for some $l$.

The motivation behind these clique elimination ideals is to find sparse generators of the ideal that are the closest to a Gröbner basis. Lex Gröbner bases can be very

large, and thus finding a sparse approximation to them might be much faster, as will be seen in section 7. We attempt to find such an "optimal" sparse representation by using chordal elimination.

Specifically, let $gb_{H_l}$ denote a lex Gröbner basis of each $H_l$. We argue that the concatenation $\bigcup_l gb_{H_l}$ constitutes such a closest sparse representation. In particular, the following proposition says that if there exists a lex Gröbner basis of $I$ that preserves the structure, then $\bigcup_l gb_{H_l}$ is also one.

PROPOSITION 2. *Let $I$ be an ideal with graph $G$, and let $gb$ be a lex Gröbner basis. Let $H_l$ denote the clique elimination ideals, and let $gb_{H_l}$ be the corresponding lex Gröbner bases. If $gb$ preserves the graph structure, i.e., $G(gb) \subseteq G$, then $\bigcup_l gb_{H_l}$ is a lex Gröbner basis of $I$.*

*Proof.* It is clear that $gb_{H_l} \subseteq H_l \subseteq I$. Let $m \in \mathrm{in}(I)$ be some monomial; we just need to show that $m \in \mathrm{in}(\bigcup_l g_{H_l})$. As $\mathrm{in}(I) = \mathrm{in}(gb)$, we can restrict $m$ to be the leading monomial $m = \mathrm{lm}(p)$ of some $p \in gb$. By the assumption on $gb$, the variables of $p$ are in some clique $X_l$ of $G$. Thus, $p \in H_l$ so that $m = \mathrm{lm}(p) \in \mathrm{in}(H_l) = \mathrm{in}(gb_l)$. This concludes the proof. ∎

Before computing $H_l$, we will show how to obtain elimination ideals of simpler sets. These sets are determined by the elimination tree of the graph, and we will find the corresponding elimination ideals in section 5.1. After that we will come back to computing the clique elimination ideals in section 5.2. Finally, we will elaborate more on the relationship between lex Gröbner bases and clique elimination ideals in section 5.3.

**5.1. Elimination ideals of lower sets.** Now we will show how to find elimination ideals of some simple sets of the graph, which depend on the elimination tree. To do so, we recall that in chordal elimination we decompose $I_l = J_l + K_{l+1}$, which allows us to compute next $I_{l+1} = \mathrm{elim}_{l+1}(J_l) + K_{l+1}$. Observe that

$$\begin{aligned} I_l &= J_l + K_{l+1} \\ &= J_l + \mathrm{elim}_{l+1}(J_l) + K_{l+1} \\ &= J_l + I_{l+1} \\ &= J_l + J_{l+1} + K_{l+2} \\ &= J_l + J_{l+1} + \mathrm{elim}_{l+2}(J_{l+1}) + K_{l+2}. \end{aligned}$$

Continuing this way, we conclude that

$$I_l = J_l + J_{l+1} + \cdots + J_{n-1}. \tag{10}$$

We will obtain a similar summation formula for other elimination ideals apart from $I_l$.

Consider again the elimination tree $T$. We present another characterization of it.

PROPOSITION 3. *Consider the directed acyclic graph (DAG) obtained by orienting the edges of $G$ with the order of its vertices. Then the elimination tree $T$ corresponds to the transitive reduction of such a DAG. Equivalently, $T$ is the Hasse diagram of the poset associated to the DAG.*

*Proof.* As $T$ is a tree, it is reduced, and thus we just need to show that any arc from the DAG corresponds to a path of $T$. Let $x_i \to x_j$ be an arc in the DAG, and observe that being an arc is equivalent to $x_j \in X_i$. Let $x_p$ be the parent of $x_i$. Then Lemma 2 implies $x_j \in X_p$, and thus $x_p \to x_j$ is in the DAG. Similarly, if $x_r$

is the parent of $x_p$, then $x_r \to x_j$ is another arc. By continuing this way we find a path $x_i, x_p, x_r, \ldots$ in $T$ that connects $x_i \to x_j$, proving that $T$ is indeed the transitive reduction. ∎

DEFINITION 8. *We say a set of variables $\Lambda$ is a* lower set *if $T|_\Lambda$ is also a tree rooted in $x_{n-1}$. Equivalently, $\Lambda$ is a lower set of the poset associated to the DAG of Proposition* 3.

Observe that $\{x_l, x_{l+1}, \ldots, x_{n-1}\}$ is a lower set, as when we remove $x_0, x_1, \ldots$ we are pruning some leaf of $T$. The following lemma gives a simple property of lower sets.

LEMMA 8. *If $X$ is a set of variables such that $G|_X$ is a clique, then $T|_X$ is contained in some branch of $T$. In particular, if $x_l > x_m$ are adjacent, then any lower set containing $x_l$ must also contain $x_m$.*

*Proof.* For the first part, note that the DAG induces a poset on the vertices, and restricted to $X$ we get a linear order. Thus, in the Hasse diagram, $X$ must be part of a chain (branch). The second part follows by considering the clique $X = \{x_l, x_m\}$ and using the previous result. ∎

The next lemma tells us how to obtain the elimination ideals of any lower set.

LEMMA 9. *Let $I$ be an ideal, let $V = \mathbf{V}(I)$, and assume that the domination condition holds for chordal elimination. Let $\Lambda \subseteq \{x_0, \ldots, x_{n-1}\}$ be a lower set. Then,*

$$I \cap \mathbb{K}[\Lambda] \overset{\text{rad}}{=} \sum_{x_i \in \Lambda} J_i$$

*and the corresponding variety is $\pi_\Lambda(V)$, where $\pi_\Lambda : \mathbb{K}^n \to \mathbb{K}^\Lambda$ is the projection onto $\Lambda$.*

*Proof.* See section A.2 in the appendix. ∎

**5.2. Cliques elimination algorithm.** Lemma 9 tells us that we can very easily obtain the elimination ideal of any lower set. We return now to the problem of computing the elimination ideals of the cliques $X_l$, which we denoted as $H_l$. Before showing how to get them, we need a simple lemma.

LEMMA 10. *Let $G$ be a chordal graph, and let $X$ be a clique of $G$. Then there is a perfect elimination ordering $v_0, \ldots, v_{n-1}$ of $G$ such that the last vertices of the ordering correspond to $X$, i.e., $X = \{v_{n-1}, v_{n-2}, \ldots, v_{n-|X|}\}$.*

*Proof.* We can apply MCS (Algorithm 1) to the graph, choosing at the beginning all the vertices of clique $X$. As the graph is chordal, this gives a reversed perfect elimination ordering. ∎

THEOREM 4. *Let $I$ be a zero-dimensional ideal with chordal graph $G$. Assume that the domination condition holds for chordal elimination. Then we can further compute ideals $H_l \in \mathbb{K}[X_l]$ such that $H_l \overset{\text{rad}}{=} I \cap \mathbb{K}[X_l]$, preserving the structure.*

*Proof.* We will further prove that the corresponding variety is $\pi_{X_l}(V)$, where $V = \mathbf{V}(I)$ and $\pi_{X_l} : \mathbb{K}^n \to \mathbb{K}^{X_l}$ is the projection onto $X_l$. We proceed by induction on $l$.

The base case is $l = n-1$. As chordal elimination is successful, $I_{n-1} \overset{\text{rad}}{=} \text{elim}_{n-1}(I)$ and the variety is $\pi_{n-1}(V)$, so we can set $H_{n-1} = I_{n-1}$.

Assume that we found $H_m$ for all $x_m < x_l$. Let $\Lambda$ be a lower set with largest element $x_l$. By Lemma 9, we can compute an ideal $I_\Lambda$ with $\mathbf{V}(I_\Lambda) = \pi_\Lambda(V)$. Note

that $X_l \subseteq \Lambda$ because of Lemma 8. Thus, we should use as $H_l$ the ideal $I_\Lambda \cap \mathbb{K}[X_l]$. Naturally, we will use chordal elimination to approximate this ideal. For a reason that will be clear later, we modify $I_\Lambda$, appending to it the ideals $H_r$ for all $x_r \in \Lambda \setminus \{x_l\}$. Observe that this does not change the variety.

Consider the induced graph $G|_\Lambda$, which is also chordal as $G$ is chordal. Thus, Lemma 10 implies that there is a perfect elimination ordering $\sigma$ of $G|_\Lambda$, where the last clique is $X_l$. We can now use Algorithm 2 in the ideal $I_\Lambda$ using such ordering of the variables to find an ideal $H_l$ that approximates $I_\Lambda \cap \mathbb{K}[X_l]$. We will show now that this elimination is successful and thus $H_l \overset{\text{rad}}{=} I_\Lambda \cap \mathbb{K}[X_l]$.

Let $X_j^\sigma \subseteq G|_\Lambda$ denote the cliques as defined in (1) but using the new ordering $\sigma$ in $G|_\Lambda$. Similarly, let $I_j^\sigma = J_j^\sigma + K_{j+1}^\sigma$ denote the clique decompositions used in chordal elimination with such an ordering. Let $x_m$ be one variable that we need to eliminate to obtain $H_l$, i.e., $x_m \in \Lambda \setminus X_l$. Let us assume that $x_m$ is such that $X_m^\sigma$ is a maximal clique of $G|_\Lambda$. As the maximal cliques do not depend on the ordering, it means that $X_m^\sigma = X_r$ for some $x_r < x_l$, and thus we already found an $H_r$ with $H_r \overset{\text{rad}}{=} I \cap \mathbb{K}[X_m^\sigma]$. Observe that $H_r \subseteq J_m^\sigma$ by recalling that we appended $H_r$ to $I_\Lambda$ and using Lemma 3. As $H_r$ is zero-dimensional, $J_m^\sigma$ is also zero-dimensional for all such $x_m$. Therefore, Corollary 2 says that the domination condition holds and chordal elimination succeeds.

Finally, let us prove that $H_l \overset{\text{rad}}{=} I \cap \mathbb{K}[X_l]$. Observe that as the domination condition holds in the elimination above (to get $H_l$), we have

$$\mathbf{V}(H_l) = \mathbf{V}(I_\Lambda \cap \mathbb{K}[X_l]) = \pi_{X_l}(\mathbf{V}(I_\Lambda)).$$

As $\mathbf{V}(I_\Lambda) = \pi_\Lambda(V)$, we obtain that $\mathbf{V}(H_l) = \pi_{X_l}(V)$. On the other hand, we also have $H_l \subseteq I \cap \mathbb{K}[X_l]$, so that

$$\mathbf{V}(H_l) \supseteq \mathbf{V}(I \cap \mathbb{K}[X_l]) \supseteq \pi_{X_l}(V).$$

Therefore, the three terms above must be equal. ∎

Observe that the above proof hints at an algorithm for computing $H_l$. However, the proof depends on the choice of some lower set $\Lambda$ for each $x_l$. To avoid eliminations we want to use a lower set $\Lambda$ as small as possible. By making a good choice we can greatly simplify the procedure, and we get, after some observations made in Corollary 4, Algorithm 3. Note that this procedure recursively computes the clique elimination ideals: for a given node $x_l$ it only requires $J_l$ and the clique elimination ideal of its parent $x_p$.

COROLLARY 4. *Let $I$ be a zero-dimensional ideal with chordal graph $G$. Assume that the domination condition holds for chordal elimination. Then Algorithm 3 correctly computes the clique elimination ideals, i.e., $H_l \overset{\text{rad}}{=} I \cap \mathbb{K}[X_l]$.*

*Proof.* We refer the reader to the proof of Theorem 4. For a given $x_l$, let $x_p$ be its parent, and let $P_l$ denote the directed path in $T$ from $x_l$ to the root $x_{n-1}$. It is easy to see that $P_l$ is a lower set and that $P_l = P_p \cup \{x_l\}$. We will see that Algorithm 3 corresponds to selecting the lower set $\Lambda$ to be this $P_l$ and reusing the eliminations performed to get $H_p$ when we compute $H_l$.

In the proof of Theorem 4, to get $H_l$ we need a perfect elimination ordering (PEO) $\sigma_l$ of $G|_\Lambda$ that ends in $X_l$. This order $\sigma_l$ determines the eliminations performed in $I_\Lambda$. Let $\sigma_p$ be a PEO of $G|_{P_p}$, whose last vertices are $X_p$. Let us see that we can extend $\sigma_p$ to obtain the PEO $\sigma_l$ of $G|_{P_l}$. Let $C := X_p \cup \{x_l\}$, and observe that $X_l \subseteq C$ due

**Algorithm 3** Compute elimination ideals of cliques.

**Input:** An ideal $I$, given by generators with chordal graph $G$

**Output:** Ideals $H_l$ such that $H_l \overset{\text{rad}}{=} I \cap \mathbb{K}[X_l]$

```
1: procedure CliquesElim(I, G)
2:     get cliques X_0, ..., X_{n-1} of G
3:     get J_0, ..., J_{n-1} from ChordElim(I, G)
4:     H_{n-1} = J_{n-1}
5:     for l = n - 2 : 0 do
6:         x_p = parent of x_l
7:         C = X_p ∪ {x_l}
8:         I_C = H_p + J_l
9:         order = MCS(G|_C, start = X_l)
10:        H_l = ChordElim(I_C^order, G|_C^order)
11:    return H_0, ..., H_{n-1}
```

to Lemma 2, and thus $P_l = P_p \cup C$. Let $\sigma_C$ be a PEO of $G|_C$ whose last vertices are $X_p$ (using Lemma 10). We will argue that the following ordering works:

$$\sigma_l := (\sigma_p \setminus X_p) + \sigma_C.$$

By construction, the last vertices of $\sigma_l$ are $X_l$, so we just need to show that it is indeed a PEO of $G|_{P_l}$. Let $v \in P_l$, and let $X_v^{\sigma_l}$ be the vertices adjacent to it that follow $v$ in $\sigma_l$. We need to show that $X_v^{\sigma_l}$ is a clique. There are two cases: $v \in C$ or $v \notin C$. If $v \in C$, then $X_v^{\sigma_l}$ is the same as with $\sigma_C$, so that it is a clique because $\sigma_C$ is a PEO. If $v \notin C$, we will see that $X_v^{\sigma_l}$ is the same as with $\sigma_p$, and thus it is a clique. Consider the partition $X_v^{\sigma_l} = (X_v^{\sigma_l} \setminus X_p) \cup (X_v^{\sigma_l} \cap X_p)$, and note that the part that is not in $X_p$ depends only on $\sigma_p$. The part in $X_p$ is just $adj(v) \cap X_p$, i.e., its neighbors in $X_p$, given that we put $\sigma_C$ at the end of $\sigma_l$. Observe that the same happens for $\sigma_p$, i.e., $X_v^{\sigma_p} \cap X_p = adj(v) \cap X_p$, by construction of $\sigma_p$. Thus $X_v^{\sigma_l} = X_v^{\sigma_p}$, as we wanted.

The argument above shows that given any PEO of $P_p$ and any PEO of $C$, we can combine them into a PEO of $P_l$. This implies that the eliminations performed to obtain $H_p$ can be reused to obtain $H_l$, and the remaining eliminations correspond to $G|_C$. Thus, we can obtain these clique elimination ideals recursively, as it is done in Algorithm 3. ∎

Computing a Gröbner basis for all maximal cliques in the graph might be useful, as it decomposes the system of equations into simpler ones. We can extract the solutions of the system by solving the subsystems in each clique independently and "glueing" them. We elaborate on this now.

LEMMA 11. *Let $I$ be an ideal, and let $H_j = I \cap \mathbb{K}[X_j]$ be the cliques elimination ideals. Then,*

$$I = H_0 + H_1 + \cdots + H_{n-1}.$$

*Proof.* As $H_j \subseteq I$ for any $x_j$, we have $H_0 + \cdots + H_{n-1} \subseteq I_l$. On the other hand, let $f \in I$ be one of its generators. By definition of $G$, the variables of $f$ must be contained in some $X_j$, so we have $f \in H_j$. This implies $I \subseteq H_0 + \cdots + H_{n-1}$. ∎

Lemma 11 gives us a strategy for solving zero-dimensional ideals. Note that $H_j$ is also zero-dimensional. Thus, we can compute the elimination ideals of the maximal cliques, solve each $H_j$ independently, and finally merge the solutions. We illustrate this now.

*Example* 9. Let $G$ be the blue/solid graph in Figure 1, and let $I$ be given by

$$\begin{aligned} x_i^3 - 1 &= 0, && 0 \le i \le 8, \\ x_9 - 1 &= 0, && \\ x_i^2 + x_i x_j + x_j^2 &= 0, && (i,j) \text{ blue/solid edge.} \end{aligned}$$

Note that the graph associated to the above equations is precisely $G$. However, to use chordal elimination we need to consider the chordal completion $\overline{G}$, which includes the three green/dashed edges of Figure 1. In such a completion, we identify seven maximal cliques:

$$\begin{gathered} X_0 = \{x_0,x_6,x_7\},\ X_1 = \{x_1,x_4,x_9\},\ X_2 = \{x_2,x_3,x_5\}, \\ X_3 = \{x_3,x_5,x_7,x_8\},\ X_4 = \{x_4,x_5,x_8,x_9\}, \\ X_5 = \{x_5,x_7,x_8,x_9\},\ X_6 = \{x_6,x_7,x_8,x_9\}. \end{gathered}$$

With Algorithm 3 we can find the associated elimination ideals. Some of them are

$$\begin{aligned} H_0 &= \langle x_0 + x_6 + 1, x_6^2 + x_6 + 1, x_7 - 1\rangle, \\ H_5 &= \langle x_5 - 1, x_7 - 1, x_8^2 + x_8 + 1, x_9 - 1\rangle, \\ H_6 &= \langle x_6 + x_8 + 1, x_7 - 1, x_8^2 + x_8 + 1, x_9 - 1\rangle. \end{aligned}$$

Denoting $\zeta = e^{2\pi i/3}$, the corresponding varieties are

$$\begin{aligned} H_0 : \{x_0, x_6, x_7\} &\to \left\{\zeta, \zeta^2, 1\right\},\ \left\{\zeta^2, \zeta, 1\right\}, \\ H_5 : \{x_5, x_7, x_8, x_9\} &\to \{1, 1, \zeta, 1\},\ \left\{1, 1, \zeta^2, 1\right\}, \\ H_6 : \{x_6, x_7, x_8, x_9\} &\to \left\{\zeta^2, 1, \zeta, 1\right\}, \left\{\zeta, 1, \zeta^2, 1\right\}. \end{aligned}$$

There are only two solutions to the whole system, one of which corresponds to the values on the left and the other to the values on the right.

From the example above we can see that to obtain a solution of $I$ we have to match solutions from different cliques $H_l$. We can do this matching iteratively following the elimination tree. Any partial solution is guaranteed to extend, as the elimination was successful. Let us see now an example where this matching gets a bit more complex.

*Example* 10. Consider again the blue/solid graph in Figure 1, and let $I$ be given by:

$$\begin{aligned} x_i^4 - 1 &= 0, && 0 \le i \le 8, \\ x_9 - 1 &= 0, && \\ x_i^3 + x_i^2 x_j + x_i x_j^2 + x_j^3 &= 0, && (i,j) \text{ blue/solid edge.} \end{aligned}$$

The graph (and cliques) is/are the same as in Example 9, but this time the variety is larger. This time we have $|\mathbf{V}(H_0)| = 18$, $|\mathbf{V}(H_5)| = 27$, $|\mathbf{V}(H_6)| = 12$. These numbers are still small. However, when we merge all partial solutions we obtain $|\mathbf{V}(I)| = 528$.

**5.3. Lex Gröbner bases and chordal elimination.** To finalize this section, we will show the relationship between lex Gröbner bases of $I$ and lex Gröbner bases of the clique elimination ideals $H_l$. We will see that both share many structural properties. This justifies our claim that these polynomials are the closest sparse

representation of $I$ to a lex Gröbner basis. In some cases, the concatenation of the clique Gröbner bases might already be a lex Gröbner basis of $I$. This was already seen in Proposition 2, and we will see now another situation where this holds. In other cases, a lex Gröbner basis can be much larger than the concatenation of the clique Gröbner bases. As we can find $H_l$ while preserving sparsity, we can outperform standard Gröbner bases algorithms in many cases, as will be seen in section 7.

We focus on radical zero-dimensional ideals $I$. Note that this radicality assumption is not restrictive, as we have always been concerned with $\mathbf{V}(I)$, and we can compute $\sqrt{H_l}$ for each $l$. We recall now that in many cases (e.g., generic coordinates) a radical zero-dimensional ideal has a very special type of Gröbner basis. We say that $I$ is in *shape position* if the reduced lex Gröbner basis has the structure

$$x_0 - g_0(x_{n-1}),\ \ x_1 - g_1(x_{n-1}), \ldots,\ x_{n-2} - g_{n-2}(x_{n-1}),\ \ g_{n-1}(x_{n-1}).$$

Later we will prove the following result for ideals in shape position.

PROPOSITION 4. *Let $I$ be a radical zero-dimensional ideal in shape position. Let $gb_{H_l}$ be a lex Gröbner basis of $H_l$. Then $\bigcup_l gb_{H_l}$ is a lex Gröbner basis of $I$.*

If the ideal is not in shape position, then the concatenation of such smaller Gröbner bases might not be a Gröbner basis for $I$. Indeed, in many cases any Gröbner basis for $I$ is extremely large, while the concatenated polynomials $gb_{H_l}$ are relatively small, as they preserve the structure. This will be seen in the application studied in section 7.1, where we will show how much simpler $\bigcup_l gb_{H_l}$ can be compared to a full Gröbner basis.

Even when the ideal is not in shape position, the concatenated polynomials already have some of the structure of a lex Gröbner basis of $I$, as we will show. Therefore, it is usually simpler to find such a Gröbner basis starting from such concatenated polynomials. In fact, in section 7.1 we show that by doing this we can compute a lex Gröbner basis faster than a degrevlex Gröbner basis.

THEOREM 5. *Let $I$ be a radical zero-dimensional ideal. For each $x_l$ let $gb_{I_l}$ and $gb_{H_l}$ be minimal lex Gröbner bases for the elimination ideals $I_l = \mathrm{elim}_l(I)$ and $H_l = I \cap \mathbb{K}[X_l]$. Denoting* deg *as the degree, the following sets are equal:*

$$D_{I_l} = \{\deg_{x_l}(p) : p \in gb_{I_l}\},$$
$$D_{H_l} = \{\deg_{x_l}(p) : p \in gb_{H_l}\}.$$

*Proof.* See section A.2 in the appendix. ∎

COROLLARY 5. *Let $I$ be a radical zero-dimensional ideal; then for each $x_l$ we have that $x_l^d \in \mathrm{in}(I)$ if and only if $x_l^d \in \mathrm{in}(H_l)$, using lex ordering.*

*Proof.* Let $gb_{I_l}, gb_{H_l}$ be minimal lex Gröbner bases of $I_l, H_l$. As $I$ is zero-dimensional, there are $d_l, d_H$ such that $x_l^{d_l}$ is the leading monomial of some polynomial in $gb_{I_l}$, and $x_l^{d_H}$ is the leading monomial of some polynomial in $gb_{H_l}$. All we need to show is that $d_l = d_H$. This follows by noting that $d_l = \max\{D_{I_l}\}$ and $d_H = \max\{D_{H_l}\}$, following the notation from Theorem 5. ∎

*Proof of Proposition* 4. As $I$ is in shape position, its initial ideal has the form

$$\mathrm{in}(I) = \langle x_0, x_1, \ldots, x_{n-2}, x_{n-1}^d \rangle$$

for some $d$. For each $x_l > x_{n-1}$, Corollary 5 implies that $gb_{H_l}$ contains some $f_l$ with leading monomial $x_l$. For $x_{n-1}$, the corollary says that there is an $f_{n-1} \in gb_{H_{n-1}}$ with leading monomial $x_{n-1}^d$. Then $\mathrm{in}(I) = \langle \mathrm{lm}(f_0), \ldots, \mathrm{lm}(f_{n-1}) \rangle$ and as $f_l \in H_l \subseteq I$, these polynomials form a Gröbner basis of $I$. ∎

**6. Complexity analysis.** Solving systems of polynomials in the general case is hard even for small treewidth, as was shown in Example 1. Therefore, we need some additional assumptions to ensure tractable computation. In this section we study the complexity of chordal elimination for a special type of ideals, where we can prove such tractability.

Chordal elimination shares the same limitations as other elimination methods. In particular, for zero-dimensional ideals its complexity is intrinsically related to the size of the projection $|\pi_l(\mathbf{V}(I))|$. Thus, we will make certain assumptions on the ideal that allow us to bound the size of this projection. The following concept will be key.

DEFINITION 9 ($q$-domination). *We say that a polynomial $f$ is* $(x_i, q)$-dominated *if its leading monomial has the form $x_i^d$ for some $d \leq q$. Let $I = \langle f_1, \ldots, f_s \rangle$; we say that $I$ is* $q$-dominated *if for each $x_i$ there is a generator $f_j$ that is $(x_i, q)$-dominated.*

We will assume that $I$ satisfies this $q$-dominated condition. Observe that Corollary 3 holds, and thus chordal elimination succeeds. Note that this condition also implies that $I$ is zero-dimensional.

It should be mentioned that the $q$-dominated condition applies to finite fields. Let $\mathbb{F}_q$ denote the finite field of size $q$. If we are interested in solving a system of equations in $\mathbb{F}_q$ (as opposed to its algebraic closure), we can add the equations $x_i^q - x_i$. Even more, by adding such equations we obtain the radical ideal $\mathbf{I}(\mathbf{V}_{\mathbb{F}_q}(I))$ [23].

We need to know the complexity of computing a lex Gröbner basis. To simplify the analysis, we assume from now on that the generators of the ideal have been *preprocessed* to avoid redundancies. Specifically, we make the assumption that the polynomials have been reduced so that no two of them have the same leading monomial and no monomial is divisible by $x_i^{q+1}$. Note that the latter assumption can be made because the ideal is $q$-dominated. These conditions allow us to bound the number of polynomials.

LEMMA 12. *Let $I = \langle f_1, \ldots, f_s \rangle$ be a preprocessed $q$-dominated ideal. Then $s = O(q^n)$.*

*Proof.* As $I$ is $q$-dominated, for each $0 \leq i < n$ there is a generator $g_i$ with leading monomial $x_i^{d_i}$ with $d_i \leq q$. The leading monomials of all generators, other than the $g_i$'s, are not divisible by $x_i^q$. There are only $q^n$ monomials with degrees less than $q$ in any variable. As the leading monomials of the generators are different, the result follows. ∎

The complexity of computing a Gröbner basis for a zero-dimensional ideal is known to be single exponential in $n$ [30]. This motivates the following definition.

DEFINITION 10. *Let $\alpha$ be the* smallest constant *such that the complexity of computing a Gröbner basis is $\widetilde{O}(q^{\alpha n})$ for any (preprocessed) $q$-dominated ideal. Here $\widetilde{O}$ ignores polynomial factors in $n$.*

A rough estimate of $\alpha$ is stated next. The proof in [23] is for the case of $\mathbb{F}_q$, but the only property used there is that the ideal is $q$-dominated.

PROPOSITION 5 (see [23]). *Buchberger's algorithm in a $q$-dominated ideal requires $O(q^{6n})$ field operations.*

We should mention that the complexity of Gröbner bases has been actively studied, and different estimates are available. For instance, Faugère et al. [19] show that for generic ideals the complexity is $\widetilde{O}(D^\omega)$, where $D$ is the number of solutions and $2 < \omega < 3$ is the exponent of matrix multiplication. Thus, if we only considered

generic polynomials we could interpret such a condition as saying that $\alpha \leq \omega$. However, even if the generators of $I$ are generic, our intermediate calculations are not generic, and thus we cannot make such an assumption.

Nevertheless, to obtain good bounds for chordal elimination we need a slightly stronger condition than $I$ being $q$-dominated. Let $X_1, \ldots, X_r$ denote the *maximal cliques* of the graph $G$, and let

$$\hat{H}_j = \langle f : f \text{ generator of } I,\ f \in \mathbb{K}[X_j] \rangle. \tag{11}$$

Note that $\hat{H}_j \subseteq I \cap \mathbb{K}[X_j]$. We assume that each (maximal) $\hat{H}_j$ is $q$-dominated. Note that such a condition is also satisfied in the case of finite fields. The following lemma shows the reason why we need this assumption.

LEMMA 13. *Let $I$ be such that for each maximal clique $X_j$ the ideal $\hat{H}_j$ (as in (11)) is $q$-dominated. Then in Algorithm 2 we have that $J_l$ is $q$-dominated for any $x_l$.*

*Proof.* See section A.3 in the appendix. ∎

It should be mentioned that whenever we have a zero-dimensional ideal $I$ such that each $\hat{H}_j$ is also zero-dimensional, the same results apply by letting $q$ be the largest degree in a Gröbner basis of any $\hat{H}_j$.

Now we derive complexity bounds, in terms of field operations, for chordal elimination under the assumptions of Lemma 13. We use the following *parameters*: $n$ is the number of variables, $s$ is the number of equations, and $\kappa$ is the clique number (or treewidth), i.e., the size of the largest clique of $G$.

THEOREM 6. *Let $I$ be such that each (maximal) $\hat{H}_j$ is $q$-dominated. In Algorithm 2, the complexity of computing $I_l$ is $\widetilde{O}(s + lq^{\alpha\kappa})$. We can find all elimination ideals in $\widetilde{O}(nq^{\alpha\kappa})$. Here $\widetilde{O}$ ignores polynomial factors in $\kappa$.*

*Proof.* In each iteration there are essentially only two relevant operations: decomposing $I_l = J_l + K_{l+1}$ and finding a Gröbner basis for $J_l$.

For each $x_l$, Lemma 13 tells us that $J_l$ is $q$-dominated. Thus, we can compute a lex Gröbner basis of $J_l$ in $\widetilde{O}(q^{\alpha\kappa})$. Here we assume that the initial $s$ equations were preprocessed and note that the following equations are also preprocessed as they are obtained from minimal Gröbner bases. To obtain $I_l$ we compute at most $l$ Gröbner bases, which we do in $\widetilde{O}(lq^{\alpha\kappa})$.

It remains only to bound the time of decomposing $I_l = J_l + K_{l+1}$. Note that if we do this decomposition in a naive way, we will need $\Theta(ls)$ operations. But we can improve such a bound easily. For instance, assume that in the first iteration we compute for every generator $f_j$ the largest $x_l$ such that $f_j \in J_l$. Thus, $f_j$ will be assigned to $K_{m+1}$ for all $x_m > x_l$, and then it will be assigned to $J_l$. We can do this computation in $\widetilde{O}(s)$. We can repeat the same process for all polynomials $p$ that we get throughout the algorithm. Let $s_l$ be the number of generators of $\mathrm{elim}_{l+1}(J_l)$. Then we can do all decompositions in $\widetilde{O}(s + s_0 + s_1 + \cdots + s_{l-1})$. We just need to bound $s_l$.

It follows from Lemma 12 that for each clique $X_l$, the size of any minimal Gröbner basis of arbitrary polynomials in $X_l$ is at most $q^\kappa + \kappa$. As the number of generators of $\mathrm{elim}_{l+1}(J_l)$ is bounded by the size of the Gröbner basis of $J_l \subseteq \mathbb{K}[X_l]$, we have $s_l = \widetilde{O}(q^\kappa)$. Thus, we can do all decompositions in $\widetilde{O}(s + lq^\kappa)$.

Hence, the total cost to compute $I_l$ is

$$\widetilde{O}(s + lq^\kappa + lq^{\alpha\kappa}) = \widetilde{O}(s + lq^{\alpha\kappa}).$$

In particular, we can compute $I_{n-1}$ in $\widetilde{O}(s+nq^{\alpha\kappa})$. Note that as each of the original $s$ equations is in some $X_l$, Lemma 12 implies that $s = O(nq^{\kappa})$. Thus, we can find all elimination ideals in $\widetilde{O}(nq^{\alpha\kappa})$. ∎

*Remark.* Note that to compute the bound $W$ of (7) we need to use chordal elimination $l$ times, so the complexity is $O(ls + l^2 q^{\alpha\kappa})$.

COROLLARY 6. *Let $I$ be such that each (maximal) $\hat{H}_j$ is $q$-dominated. The complexity of Algorithm 3 is $\widetilde{O}(nq^{\alpha\kappa})$. Thus, we can also describe $\mathbf{V}(I)$ in $\widetilde{O}(nq^{\alpha\kappa})$. Here $\widetilde{O}$ ignores polynomial factors in $\kappa$.*

*Proof.* The first part of the algorithm is chordal elimination, which we can do in $O(nq^{\alpha\kappa})$, as shown above. Observe also that MCS runs in linear time, so we can ignore it. The only missing part is computing the elimination ideas of $I_C$, where $C = X_p \cup \{x_l\}$. As $|C| \leq \kappa + 1$, the cost of chordal elimination is $\widetilde{O}(\kappa q^{\alpha\kappa}) = \widetilde{O}(q^{\alpha\kappa})$. Thus, the complexity of Algorithm 3 is still $\widetilde{O}(nq^{\alpha\kappa})$.

We now prove the second part. As mentioned, Corollary 3 applies for $q$-dominated ideals, so all eliminations are successful. From Lemma 11 and the following remarks we know that the elimination ideals $H_l$, found with Algorithm 3, give a natural description of $\mathbf{V}(I)$. ∎

The bounds above tell us that for a fixed $\kappa$, we can find all clique elimination ideals, and thus describe the variety, in $O(n)$. This is reminiscent of many graph problems (e.g., Hamiltonian circuit, vertex colorings, vertex cover) which are NP-hard in general, but are linear for fixed treewidth [6]. Similar results hold for some types of CSPs [16]. These types of problems are said to be fixed-parameter-tractable (FPT) with treewidth as the parameter.

Our methods provide an algebraic solution to some classical graph problems. In particular, we show an application of the bounds above for finding graph colorings. It is known that the coloring problem can be solved in linear time for bounded treewidth [6]. We can prove the same result by encoding colorings into polynomials.

COROLLARY 7. *Let $G$ be a graph, and let $\bar{G}$ be a chordal completion with largest clique of size $\kappa$. We can describe all $q$-colorings of $G$ in $\widetilde{O}(nq^{\alpha\kappa})$.*

*Proof.* It is known that graph $q$-colorings can be encoded with the following system of polynomials:

$$x_i^q - 1 = 0, \qquad i \in V, \tag{12a}$$

$$x_i^{q-1} + x_i^{q-2}x_j + \cdots + x_i x_j^{q-2} + x_j^{q-1} = 0, \qquad (i,j) \in E, \tag{12b}$$

where $V, E$ denote the vertices and edges, and where each color corresponds to a different square root of unity [3, 28]. Note that the ideal $I_G$ given by these equations satisfies the $q$-dominated condition stated before. The chordal graph associated to such an ideal is $\bar{G}$. The result follows from Corollary 6. ∎

To conclude, we emphasize the differences between our results and results using similar methods in graph theory and constraint satisfaction. First, note that for systems of polynomials we do not know a priori a discrete set of possible solutions. And even if the variety is finite, the solutions may not have a rational (or radical) representation. In addition, by using Gröbner bases methods we take advantage of many well-studied algebraic techniques. Finally, even though our analysis here assumes zero dimensionality, we can use our methods in underconstrained systems, and if they are close to satisfying the $q$-dominated condition, they should perform well. Indeed, in section 7.3 we test our methods on underconstrained systems.

**7. Applications.** In this section we show numerical evaluations of the approach proposed in some concrete applications. Our algorithms were implemented using Sage [41]. Gröbner bases are computed with Singular's interface [17], except when $\mathbb{K} = \mathbb{F}_2$ for which we use PolyBoRi's interface [7]. Chordal completions of small graphs ($n < 32$) are found using Sage's vertex separation algorithm. The experiments were performed on an i7 PC with 3.40GHz, 15.6 GB RAM, running Ubuntu 12.04.

We will show the performance of chordal elimination compared to the Gröbner bases algorithms from Singular and PolyBoRi. In all of the applications we give here, chordal elimination is successful because of the results of section 4. It can be seen below that in all of the applications, as the problem gets bigger our methods perform better than the algorithms from Singular and PolyBoRi.

As mentioned, chordal elimination shares some of the limitations of other elimination methods, and it performs the best under the conditions studied in section 6. We show two examples that meet such conditions in sections 7.1 and 7.2. The first case relates to the coloring problem, which was already mentioned in Corollary 7. The second case is an application to cryptography, where we solve equations over the finite field $\mathbb{F}_2$.

Sections 7.3 and 7.4 show cases where the conditions from section 6 are not satisfied. We use two of the examples from [34], where the authors study a similar chordal approach for semidefinite programming (SDP) relaxations. Gröbner bases are not as fast as SDP relaxations, as they contain more information, so we use smaller scale problems. The first example is the sensor localization problem and the second is given by discretizations of differential equations.

**7.1. Graph colorings.** We consider the $q$-colorings equations from (12) over the field $\mathbb{K} = \mathbb{Q}$. We fix the graph $G$ of Figure 4 and vary the number of colors $q$. Such a graph was considered in [28] to illustrate a characterization of uniquely colorable graphs using Gröbner bases. We use a different ordering of the vertices that determines a simpler chordal completion (the clique number is 5).

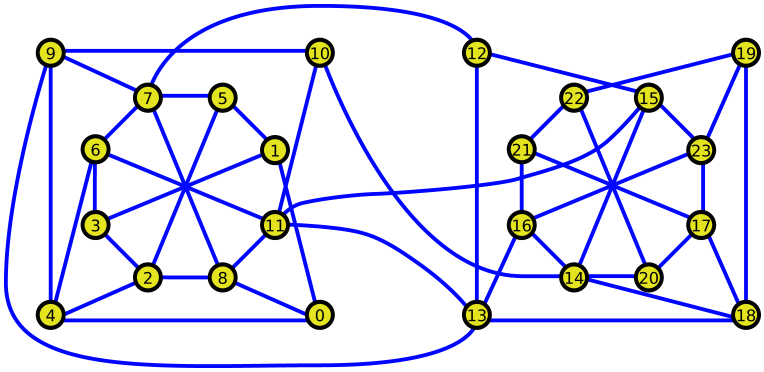


FIG. 4. *Graph with a unique* 3*-coloring* [28].

Table 1 shows the performance of Algorithms 2 and 3, compared to Singular's default Gröbner basis algorithm using degrevlex order (lex order takes much longer). It can be seen how the cost of finding a Gröbner basis increases very rapidly as we increase $q$, as opposed to our approach. In particular, for $q = 4$ we could not find a Gröbner basis after 60000 seconds (16.7 hours), but our algorithms ran in less than one second. The underlying reason for such a long time is the large size of the solution

set (number of 4-colorings), which is $|\mathbf{V}(I)| = 572656008$. Therefore, it is expected that the size of its Gröbner basis is also very large. On the other hand, the projection on each clique is much smaller, $|\mathbf{V}(H_l)| \leq 576$, and thus the corresponding Gröbner bases (found with Algorithm 3) are also much simpler.

TABLE 1
*Performance (in seconds) on* (12) *(graph of Figure* 4*) for Algorithms* 2 *and* 3*, computing a degrevlex Gröbner basis with the original equations (Singular). One experiment was interrupted after* 60000 *seconds.*

| $q$ | Variables | Equations | Monomials | ChordElim | CliquesElim | DegrevlexGB |
|---|---|---|---|---|---|---|
| 2 | 24 | 69 | 49 | 0.058 | 0.288 | 0.001 |
| 3 | 24 | 69 | 94 | 0.141 | 0.516 | 5.236 |
| 4 | 24 | 69 | 139 | 0.143 | 0.615 | $>$ 60000 |
| 5 | 24 | 69 | 184 | 0.150 | 0.614 | - |
| 6 | 24 | 69 | 229 | 0.151 | 0.638 | - |

We repeat the same experiments, this time with the blue/solid graph of Figure 1. Table 2 shows the results. This time we also show the cost of computing a lex Gröbner basis, using as input the clique elimination ideals $H_l$. Again, we observe that chordal elimination is much faster than finding a Gröbner basis. We also see that we can more quickly find a lex Gröbner basis than a degrevlex by using the output from chordal elimination.

TABLE 2
*Performance (in seconds) on* (12) *(blue/solid graph of Figure* 1*) for Algorithms* 2 *and* 3*, computing a lex Gröbner basis with input $H_l$, and computing a degrevlex Gröbner basis with the original equations (Singular).*

| $q$ | Vars | Eqs | Mons | ChordElim | CliquesElim | LexGB from $H_l$ | DegrevlexGB |
|---|---|---|---|---|---|---|---|
| 5 | 10 | 28 | 75 | 0.035 | 0.112 | 0.003 | 0.003 |
| 10 | 10 | 28 | 165 | 0.044 | 0.130 | 0.064 | 0.202 |
| 15 | 10 | 28 | 255 | 0.065 | 0.188 | 4.539 | 8.373 |
| 20 | 10 | 28 | 345 | 0.115 | 0.300 | 73.225 | 105.526 |

**7.2. Cryptography.** We consider the parametric family $SR(n, r, c, e)$ of variants of the advanced encryption standard (AES) from [8]. Such a cipher can be embedded into a structured system of polynomial equations over $\mathbb{K} = \mathbb{F}_2$, as shown in [8]. Note that as the field is finite the analysis from section 6 holds.

We compare the performance of Algorithm 2 to PolyBoRi's default Gröbner bases algorithm, using both lex and degrevlex order. As the input to the cipher is probabilistic, for the experiments we seed the pseudorandom generator in fixed values of 0, 1, 2. We fix the values $r = 1, c = 2, e = 4$ for the experiments and vary the parameter $n$, which corresponds to the number of identical blocks used for the encryption.

Table 3 shows the results of the experiments. We observe that for small problems, standard Gröbner bases outperform chordal elimination, particularly using degrevlex order. Nevertheless, chordal elimination scales better, being faster than both methods for $n = 10$. In addition, standard Gröbner bases have higher memory requirements, which is reflected in the many experiments that aborted for this reason.

TABLE 3

*Performance (in seconds) on the equations of $SR(n, 1, 2, 4)$ for Algorithm 2, and computing (lex/degrevlex) Gröbner bases (PolyBoRi). Three different experiments (seeds) are considered for each $n$. Some experiments aborted due to insufficient memory.*

| $n$ | Variables | Equations | Seed | ChordElim | LexGB | DegrevlexGB |
|---|---|---|---|---|---|---|
| 4 | 120 | 216 | 0 | 517.018 | 217.319 | 71.223 |
| | | | 1 | 481.052 | 315.625 | 69.574 |
| | | | 2 | 507.451 | 248.843 | 69.733 |
| 6 | 176 | 320 | 0 | 575.516 | 402.255 | 256.253 |
| | | | 1 | 609.529 | 284.216 | 144.316 |
| | | | 2 | 649.408 | 258.965 | 133.367 |
| 8 | 232 | 424 | 0 | 774.067 | 1234.094 | 349.562 |
| | | | 1 | 771.927 | $>$ 1500, aborted | 369.445 |
| | | | 2 | 773.359 | 1528.899 | 357.200 |
| 10 | 288 | 528 | 0 | 941.068 | $>$ 1100, aborted | 1279.879 |
| | | | 1 | 784.709 | $>$ 1400, aborted | 1150.332 |
| | | | 2 | 1124.942 | $>$ 3600, aborted | $>$ 2500, aborted |

**7.3. Sensor network localization.** We consider the *sensor network localization* problem, also called *graph realization* problem, given by the equations

$$\|x_i - x_j\|^2 = d_{ij}^2, \qquad (i,j) \in \mathcal{A}, \tag{13a}$$

$$\|x_i - a_k\|^2 = e_{ik}^2, \qquad (i,k) \in \mathcal{B}, \tag{13b}$$

where $x_1, \ldots, x_n$ are unknown sensor positions, $a_1, \ldots, a_m$ are some fixed anchors, and $\mathcal{A}, \mathcal{B}$ are some sets of pairs which correspond to sensors that are close enough. We consider the problem over the field $\mathbb{K} = \mathbb{Q}$. Observe that the set $\mathcal{A}$ determines the graph structure of the system of equations. Note also that the equations are simplicial (see Definition 7), and thus Theorem 3 says that chordal elimination succeeds. However, the conditions from section 6 are not satisfied.

We generate random test problems in a way similar to that in [34]. First, we generate $n = 20$ random sensor locations $x_i^*$ from the unit square $[0, 1]^2$. The $m = 4$ fixed anchors are $(1/2 \pm 1/4, 1/2 \pm 1/4)$. We fix a proximity threshold $D$ which we set to either $D = 1/4$ or $D = 1/3$. Set $\mathcal{A}$ is such that every sensor is adjacent to at most three more sensors and $\|x_i - x_j\| \leq D$. Set $\mathcal{B}$ is such that every anchor is related to all sensors with $\|x_i - a_k\| \leq D$. For every $(i,j) \in \mathcal{A}$ and $(i,k) \in \mathcal{B}$ we compute $d_{ij}, e_{ik}$.

We compare the performance of Algorithm 2 and Singular's algorithms. We consider Singular's default Gröbner bases algorithms, with both degrevlex and lex orderings, and the FGLM algorithm if the ideal is zero-dimensional.

We use two different values for the proximity threshold: $D = 1/4$ and $D = 1/3$. For $D = 1/4$ the system of equations is underconstrained (positive dimensional), and for $D = 1/3$ the system is overconstrained (zero-dimensional). We will observe that in both cases chordal elimination performs well. Degrevlex Gröbner bases perform slightly better in the overconstrained case, but poorly in the underconstrained case. Lex Gröbner bases do not compete with chordal elimination in either case.

Table 4 summarizes the results. We used 50 random instances for the underconstrained case ($D = 1/4$) and 100 for the overconstrained case ($D = 1/3$). We can see that in the underconstrained case neither lex nor degrevlex Gröbner bases ever finish within 1000 seconds. On the other hand, chordal elimination completes more than half of the instances. For the overconstrained case, the lex Gröbner basis algorithm continues to perform poorly. On the other hand, degrevlex Gröbner bases and the FGLM algorithm have slightly better statistics than chordal elimination.

TABLE 4

*Statistics of experiments performed on random instances of* (13). *We consider two situations:* 50 *cases of underconstrained systems* ($D = 1/4$) *and* 100 *cases of overconstrained systems* ($D = 1/3$). *Experiments are interrupted after* 1000 *seconds.*

| $D$ | Repet. | Vars | Eqs | ChordElim | LexGB | DegrevlexGB | LexFGLM | |
|---|---|---|---|---|---|---|---|---|
| 1/4 | 50 | 40 | $39 \pm 5$ | 478.520 | 1000 | 1000 | - | Mean time (s) |
| | | | | 56% | 0% | 0% | - | Completed |
| 1/3 | 100 | 40 | $48 \pm 6$ | 298.686 | 1000 | 219.622 | 253.565 | Mean time (s) |
| | | | | 73% | 0% | 81% | 77% | Completed |

Despite the better statistics of degrevlex and FGLM in the overconstrained case, one can identify that for several of such instances chordal elimination performs much better. This can be seen in Figure 5, where we observe the scatter plot of the performance of both FGLM and Algorithm 2. In about half of the cases (48) both algorithms are within one second, and for the rest, in 29 cases FGLM is better, and in 23 chordal elimination is better. To understand the difference between these two groups, we can look at the clique number of the chordal completions. Indeed, the 23 cases where chordal elimination is better have a mean clique number of 5.48, compared to 6.97 of the 29 cases where FGLM was better. This confirms that chordal elimination is a suitable method for cases with chordal structure, even in the overconstrained case.

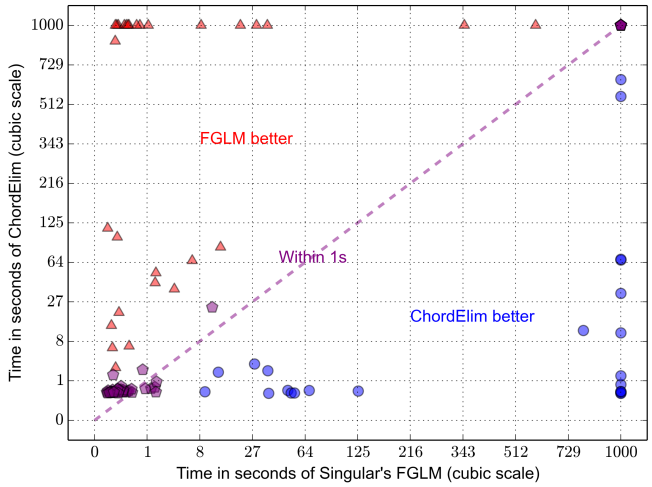


FIG. 5. *Scatter plot of the time used by Singular's FGLM and Algorithm* 2 *on* 100 *random overconstrained* ($D = 1/3$) *instances of* (13). *Darker points indicate overlap.*

**7.4. Differential equations.** Now we consider the following equations over the field $\mathbb{K} = \mathbb{Q}$:

$$0 = 2x_1 - x_2 + \frac{1}{2}h^2(x_1 + t_1)^3, \tag{14a}$$

$$0 = 2x_i - x_{i-1} - x_{i+1} + \frac{1}{2}h^2(x_i + t_i)^3 \qquad \text{for } i = 2, \ldots, n-1, \tag{14b}$$

$$0 = 2x_n - x_{n-1} + \frac{1}{2}h^2(x_n + t_n)^3, \tag{14c}$$

with $h = 1/(n+1)$ and $t_i = i/(n+1)$. Such equations were used in [34] and arise from discretizing the following differential equation with boundary conditions:

$$x'' + \frac{1}{2}(x+t)^3 = 0, \qquad x(0) = x(1) = 0.$$

Note that these polynomials are simplicial (see Definition 7), and thus chordal elimination succeeds because of Theorem 3. Even more, the equations $J_l$ obtained in chordal elimination form a lex Gröbner basis. However, the results from section 6 do not hold. Nevertheless, we compare the performance of chordal elimination to Singular's default Gröbner basis algorithm with lex order. We also consider Singular's FGLM implementation.

TABLE 5
*Performance (in seconds) on* (14) *for Algorithm* 2*, and computing a lex Gröbner basis with two standard methods (Singular's default and FGLM).*

| $n$ | Variables | Equations | ChordElim | LexGB | LexFGLM |
|---|---|---|---|---|---|
| 3 | 3 | 3 | 0.008 | 0.003 | 0.007 |
| 4 | 4 | 4 | 0.049 | 0.044 | 0.216 |
| 5 | 5 | 5 | 1.373 | 1.583 | 8.626 |
| 6 | 6 | 6 | 76.553 | 91.155 | 737.989 |
| 7 | 7 | 7 | 7858.926 | 12298.636 | 43241.926 |

Table 5 shows the results of the experiments. The fast increase in the timings observed is common to all methods. Nevertheless, it can be seen that chordal elimination performs faster and scales better than standard Gröbner bases algorithms. Even though the degrevlex term order is much simpler in this case, the FGLM algorithm is not efficient enough to obtain a lex Gröbner basis.

## Appendix A. Additional proofs.

### A.1. Proofs from section 3.

*Proof of Theorem* 2. We prove it by induction on $l$. The base case is Lemma 1. Assume that the result holds for some $l$, and let us show it for $l+1$.

By induction hypothesis $I_l, W_1, \ldots, W_l$ satisfy (4) and (5). Lemma 1 with $I_l$ as input tells us that $I_{l+1}, W_{l+1}$ satisfy

$$\overline{\pi(\mathbf{V}(I_l))} \subseteq \mathbf{V}(I_{l+1}), \tag{15}$$

$$\mathbf{V}(I_{l+1}) - \mathbf{V}(W_{l+1}) \subseteq \pi(\mathbf{V}(I_l)), \tag{16}$$

where $\pi : \mathbb{K}^{n-l} \to \mathbb{K}^{n-l-1}$ is the projection onto the last factor. Then,

$$\pi_{l+1}(\mathbf{V}(I)) = \pi(\pi_l(\mathbf{V}(I))) \subseteq \pi(\mathbf{V}(I_l)) \subseteq \mathbf{V}(I_{l+1}),$$

and as $\mathbf{V}(I_{l+1})$ is closed, we can take the closure. This shows (4).

We also have

$$\begin{aligned}\pi_{l+1}(\mathbf{V}(I)) = \pi(\pi_l(\mathbf{V}(I))) &\supseteq \pi(\mathbf{V}(I_l) - [\pi_l(\mathbf{V}(W_1)) \cup \cdots \cup \pi_l(\mathbf{V}(W_l))]) \\ &\supseteq \pi(\mathbf{V}(I_l)) - \pi[\pi_l(\mathbf{V}(W_1)) \cup \cdots \cup \pi_l(\mathbf{V}(W_l))] \\ &= \pi(\mathbf{V}(I_l)) - [\pi_{l+1}(\mathbf{V}(W_1)) \cup \cdots \cup \pi_{l+1}(\mathbf{V}(W_l))] \\ &\supseteq (\mathbf{V}(I_{l+1}) - \mathbf{V}(W_{l+1})) - [\pi_{l+1}(\mathbf{V}(W_1)) \cup \cdots \cup \pi_{l+1}(\mathbf{V}(W_l))] \\ &= \mathbf{V}(I_{l+1}) - [\pi_{l+1}(\mathbf{V}(W_1)) \cup \cdots \cup \pi_{l+1}(\mathbf{V}(W_{l+1}))],\end{aligned}$$

which proves (5). ∎

### A.2. Proofs from section 5.

*Proof of Lemma* 9. Let $H_\Lambda := I \cap \mathbb{K}[\Lambda]$ and $J_\Lambda := \sum_{x_i \in \Lambda} J_i$. Let $x_l \in \Lambda$ be its largest element. For a fixed $x_l$, we will show by induction on $|\Lambda|$ that $\mathbf{V}(H_\Lambda) = \mathbf{V}(J_\Lambda) = \pi_\Lambda(V)$.

The base case is when $\Lambda = \{x_l, \ldots, x_{n-1}\}$. Note that as $x_l$ is fixed, such a $\Lambda$ is indeed the largest possible lower set. In such a case, $J_\Lambda = I_l$ as seen in (10), and as we are assuming that the domination condition holds, $\mathbf{V}(H_\Lambda) = \mathbf{V}(I_l) = \pi_l(V)$.

Assume that the result holds for $k+1$, and let us show it for some $\Lambda$ with $|\Lambda| = k$. Consider the subtree $T_l = T|_{\{x_l,\ldots,x_{n-1}\}}$ of $T$. As $T_l|_\Lambda$ is a proper subtree of $T_l$ with the same root, there must be an $x_m < x_l$ with $x_m \notin \Lambda$ and such that $x_m$ is a leaf in $T_l|_{\Lambda'}$, where $\Lambda' = \Lambda \cup \{x_m\}$. We apply the induction hypothesis in $\Lambda'$, obtaining that $\mathbf{V}(H_{\Lambda'}) = \mathbf{V}(J_{\Lambda'}) = \pi_{\Lambda'}(V)$.

Now note that $J_m$ is a subset of both $H_{\Lambda'}, J_{\Lambda'}$. Observe also that we want to eliminate $x_m$ from these ideals to obtain $H_\Lambda, J_\Lambda$. To do so, let us change the term order to $x_m > x_l > x_{l+1} > \cdots > x_{n-1}$. Note that such a change has no effect inside $X_m$, and thus the term ordering for $J_m$ remains the same. As the domination condition holds, $J_m$ is $x_m$-dominated, and thus $H_{\Lambda'}, J_{\Lambda'}$ are also $x_m$-dominated. This means that Lemma 4 holds for $H_{\Lambda'}, J_{\Lambda'}$ when we eliminate $x_m$, and then

$$\begin{aligned}\mathbf{V}(\mathrm{elim}_{m+1}(H_{\Lambda'})) &= \pi_{m+1}(\mathbf{V}(H'_\Lambda)) = \pi_\Lambda(V), \\ \mathbf{V}(\mathrm{elim}_{m+1}(J_{\Lambda'})) &= \pi_{m+1}(\mathbf{V}(J'_\Lambda)) = \pi_\Lambda(V).\end{aligned}$$

Notice that $H_\Lambda = \mathrm{elim}_{m+1}(H_{\Lambda'})$, so all we have to do now is show that $J_\Lambda \overset{\mathsf{rad}}{=} \mathrm{elim}_{m+1}(J_{\Lambda'})$. Note that

$$\mathrm{elim}_{m+1}(J_{\Lambda'}) = \mathrm{elim}_{m+1}(J_m + J_\Lambda).$$

Observe that the last expression is reminiscent of Lemma 1, but in this case we are eliminating $x_m$. As mentioned, $J_m$ is $x_m$-dominated, so elimination succeeds. Therefore, we have

$$\mathrm{elim}_{m+1}(J_m + J_\Lambda) \overset{\mathsf{rad}}{=} \mathrm{elim}_{m+1}(J_m) + J_\Lambda.$$

Let $x_p$ be the parent of $x_m$ in $T$. Then (9) says that $\mathrm{elim}_{m+1}(J_m) \subseteq J_p$, where we are using the fact that the term order change maintains $J_m$. Observe that $x_p \in \Lambda$ by the construction of $x_m$, and then $J_p \subseteq J_\Lambda$. Then,

$$\mathrm{elim}_{m+1}(J_m) + J_\Lambda = J_\Lambda.$$

Combining the last three equations, we complete the proof. ∎

For the proof of Theorem 5, we will need two results.

LEMMA 14. *Let $I$ be a zero-dimensional ideal, let $H_j = I \cap \mathbb{K}[X_j]$, and let $I_l = \text{elim}_l(I)$. Then,*

$$I_l \overset{\text{rad}}{=} H_l + H_{l+1} + \cdots + H_{n-1}.$$

*Proof.* For each $x_j$ let $gb_{H_j}$ be a lex Gröbner basis of $H_j$. Let $F = \bigcup_{x_j} gb_{H_j}$ be the concatenation of all $gb_{H_j}$'s. Then the decomposition of $I$ from Lemma 11 says that $I = \langle F \rangle$. Observe now that if we use chordal elimination on $F$, at each step we only remove the polynomials involving some variable; we never generate a new polynomial. Therefore our approximation of the $l$th elimination ideal is given by $F_l = \bigcup_{x_j \le x_l} gb_{H_j}$. Now note that as $H_j$ is zero-dimensional it is also $x_j$-dominated, and thus Corollary 3 says that elimination succeeds. Thus $I_l \overset{\text{rad}}{=} \langle F_l \rangle = \sum_{x_j \le x_l} H_j$. ∎

THEOREM 7 (see [24]). *Let $I$ be a radical zero-dimensional ideal and let $V = \mathbf{V}(I)$. Let $gb$ be a minimal Gröbner basis with respect to an elimination order for $x_0$. Then the set*

$$D = \{\deg_{x_0}(p) : p \in gb\},$$

*where* deg *denotes the degree, is the same as*

$$F = \{|\pi^{-1}(z) \cap V| : z \in \pi(V)\},$$

*where $\pi : \mathbb{K}^n \to \mathbb{K}^{n-1}$ is the projection eliminating $x_0$.*

*Proof of Theorem* 5. If $x_l = x_{n-1}$, then $I_l = H_l$ and the assertion holds. Otherwise, note that $I_l, H_l$ are also radical zero-dimensional, so we can use Theorem 7. Let

$$F_{I_l} = \{|\pi_{I_l}^{-1}(z) \cap \mathbf{V}(I_l)| : z \in \pi_{I_l}(\mathbf{V}(I_l))\},$$
$$F_{H_l} = \{|\pi_{H_l}^{-1}(z) \cap \mathbf{V}(H_l)| : z \in \pi_{H_l}(\mathbf{V}(H_l))\},$$

where $\pi_{I_l} : \mathbb{K}^{n-l} \to \mathbb{K}^{n-l-1}$ and $\pi_{H_l} : \mathbb{K}^{|X_l|} \to \mathbb{K}^{|X_l|-1}$ are projections eliminating $x_l$. Then we know that $D_{I_l} = F_{I_l}$ and $D_{H_l} = F_{H_l}$, so we need to show that $F_{I_l} = F_{H_l}$.

For some $z \in \mathbb{K}^{n-l}$, let us denote $z =: (z_l, z_H, z_I)$, where $z_l$ is the $x_l$ coordinate, $z_H$ are the coordinates of $X_l \setminus x_l$, and $z_I$ are the coordinates of $\{x_l, \ldots, x_{n-1}\} \setminus X_l$. Thus, we have $\pi_{I_l}(z) = (z_H, z_I)$ and $\pi_{H_l}(z_l, z_H) = z_H$.

As $I$ is zero-dimensional, Lemma 14 implies that $I_l \overset{\text{rad}}{=} H_l + I_{l+1}$. Note also that $\mathbf{V}(I_{l+1}) = \pi_{I_l}(\mathbf{V}(I_l))$, as it is zero-dimensional. Then,

$$z \in \mathbf{V}(I_l) \iff (z_l, z_H) \in \mathbf{V}(H_l) \text{ and } (z_H, z_I) \in \pi_{I_l}(\mathbf{V}(I_l)).$$

Thus, for any $(z_H, z_I) \in \pi_{I_l}(\mathbf{V}(I_l))$ we have

$$(z_l, z_H, z_I) \in \mathbf{V}(I_l) \iff (z_l, z_H) \in \mathbf{V}(H_l).$$

Equivalently, for any $(z_H, z_I) \in \pi_{I_l}(\mathbf{V}(I_l))$ we have

$$z \in \pi_{I_l}^{-1}(z_H, z_I) \cap \mathbf{V}(I_l) \iff \rho(z) \in \pi_{H_l}^{-1}(z_H) \cap \mathbf{V}(H_l), \tag{17}$$

where $\rho(z_l, z_H, z_I) := (z_l, z_H)$. Therefore, $F_{I_l} \subseteq F_{H_l}$.

On the other hand, note that if $z_H \in \pi_{H_l}(\mathbf{V}(H_l))$, then there is some $z_I$ such that $(z_H, z_I) \in \pi_{I_l}(\mathbf{V}(I_l))$. Thus, for any $z_H \in \pi_{H_l}(\mathbf{V}(H_l))$ there is some $z_I$ such that (17) holds. This says that $F_{H_l} \subseteq F_{I_l}$, completing the proof. ∎

### A.3. Proofs from section 6.

*Proof of Lemma* 13. Let $x_l$ be arbitrary and let $x_m \in X_l$. We want to find a generator of $J_l$ that is $(x_m, q)$-dominated. Let $x_j \geq x_l$ be such that $X_l \subseteq X_j$ and $X_j$ is a maximal clique. Note that $x_m \in X_j$. Observe that $\hat{H}_j \subseteq J_j$ because of Lemma 3, and thus $J_j$ is $q$-dominated. Then there must be a generator $f \in J_j$ that is $(x_m, q)$-dominated.

Let us see that $f$ is a generator of $J_l$, which would complete the proof. To prove this we will show that $f \in \mathbb{K}[X_l]$, and then the result follows from Lemma 3. As the largest variable of $f$ is $x_m$, all its variables are in $X_j \setminus \{x_{m+1}, \ldots, x_j\} \subseteq X_j \setminus \{x_{l+1}, \ldots, x_j\}$. Thus, it is enough to show that

$$X_j \setminus \{x_{l+1}, x_{l+2}, \ldots, x_j\} \subseteq X_l.$$

The equation above follows by iterated application of Lemma 2, as we will see. Let $x_p$ be the parent of $x_j$ in $T$, and observe that $x_l \in X_p$ as $x_l \leq x_p$ and both are in clique $X_j$. Then Lemma 2 implies that $X_j \setminus \{x_{p+1}, \ldots, x_j\} \subseteq X_p$. If $x_p = x_l$, we are done. Otherwise, let $x_r$ be the parent of $x_p$, and observe that $x_l \in X_r$ as before. Then,

$$X_j \setminus \{x_{r+1}, \ldots, x_j\} \subseteq X_p \setminus \{x_{r+1}, \ldots, x_p\} \subseteq X_r.$$

If $x_r = x_l$, we are done. Otherwise, we can continue this process until it eventually terminates. This completes the proof. ∎

## REFERENCES


[1] S. Arnborg, D. G. Corneil, and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 277–284, doi:10.1137/0608024.

[2] G. V. Bard, N. T. Courtois, and C. Jefferson, *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers*, Report 2007/024, Cryptology ePrint Archive, http://eprint.iacr.org, 2007.

[3] D. A. Bayer, *The Division Algorithm and the Hilbert Scheme*, Ph.D. thesis, Harvard University, Cambridge, MA, 1982.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, *On the desirability of acyclic database schemes*, J. ACM, 30 (1983), pp. 479–513.

[5] J. Blair and B. Peyton, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computation, Springer, Berlin, 1993, pp. 1–29.

[6] H. L. Bodlaender and A. Koster, *Combinatorial optimization on graphs of bounded treewidth*, Comput. J., 51 (2008), pp. 255–269.

[7] M. Brickenstein and A. Dreyer, *PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials*, J. Symbol. Comput., 44 (2009), pp. 1326–1345, doi:10.1016/j.jsc.2008.02.017.

[8] C. Cid, S. Murphy, and M. Robshaw, *Small scale variants of the AES*, in Fast Software Encryption, Springer, Berlin, 2005, pp. 145–162.

[9] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*, Encyclopedia Math. Appl. 138, Cambridge University Press, Cambridge, UK, 2012.

[10] D. A. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer, Berlin, 2007.

[11] V. Dalmau, P. G. Kolaitis, and M. Y. Vardi, *Constraint satisfaction, bounded treewidth, and finite-variable logics*, in Principles and Practice of Constraint Programming (CP 2002), Springer, Berlin, 2002, pp. 310–326.

[12] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng, *A column approximate minimum degree ordering algorithm*, ACM Trans. Math. Software, 30 (2004), pp. 353–376.

[13] J. A. De Loera, J. Lee, S. Margulies, and S. Onn, *Expressing combinatorial problems by systems of polynomial equations and Hilbert's nullstellensatz*, Combin. Probab. Comput., 18 (2009), pp. 551–582, doi:10.1017/S0963548309009894.

[14] J. A. De Loera, S. Margulies, M. Pernpeintner, E. Riedl, D. Rolnick, G. Spencer, D. Stasi, and J. Swenson, *Graph-coloring ideals: Nullstellensatz certificates, Gröbner bases for chordal graphs, and hardness of Gröbner bases*, in Proceedings of the 40th International Symposium on Symbolic and Algebraic Computation (ISSAC'15), ACM, New York, 2015, pp. 133–140, doi:10.1145/2755996.2756639.

[15] R. Dechter, *Bucket elimination: A unifying framework for probabilistic inference*, in Learning in Graphical Models, Springer, Berlin, 1998, pp. 75–104.

[16] R. Dechter, *Constraint Processing*, Morgan Kaufmann, San Francisco, 2003.

[17] W. Decker, G. M. Greuel, G. Pfister, and H. Schönemann, *Singular* 4-0-2*—A Computer Algebra System for Polynomial Computations*, http://www.singular.uni-kl.de, 2015.

[18] I. Z. Emiris and J. F. Canny, *Efficient incremental algorithms for the sparse resultant and the mixed volume*, J. Symbol. Comput., 20 (1995), pp. 117–149, doi:10.1006/jsco.1995.1041.

[19] J. C. Faugère, P. Gaudry, L. Huot, and G. Renault, *Polynomial Systems Solving by Fast Linear Algebra*, preprint, arXiv:1304.6039 [cs.SC], 2013.

[20] J. C. Faugère and S. Rahmany, *Solving systems of polynomial equations with symmetries using SAGBI-Gröbner bases*, in Proceedings of the 34th International Symposium on Symbolic and Algebraic Computation (ISSAC'09), ACM, New York, 2009, pp. 151–158.

[21] J. C. Faugère, M. Safey El Din, and P. J. Spaenlehauer, *Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree* $(1,1)$*: Algorithms and complexity*, J. Symbol. Comput., 46 (2011), pp. 406–437.

[22] J. C. Faugère, P. J. Spaenlehauer, and J. Svartz, *Sparse Gröbner bases: The unmixed case*, in Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC'14), ACM, New York, 2014, pp. 178–185, doi:10.1145/2608628.2608663.

[23] S. Gao, *Counting Zeros over Finite Fields Using Gröbner Bases*, M.S. thesis in Logic and Computation, Carnegie Mellon University, Pittsburgh, PA, 2009.

[24] S. Gao, V. M. Rodrigues, and J. Stroomer, *Gröbner Basis Structure of Finite Sets of Points*, preprint, Department of Mathematical Sciences, Clemson University, Clemson, SC, http://www.math.clemson.edu/~sgao/papers/GBstr.pdf, 2003.

[25] K. Gatemann, *Symbolic solution polynomial equation systems with symmetry*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC'90), ACM, New York, 1990, pp. 112–119, doi:10.1145/96877.96907.

[26] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Ann. Discrete Math. 57, Elsevier B. V., Amsterdam, 2004.

[27] J. Herzog, T. Hibi, F. Hreinsdóttir, T. Kahle, and J. Rauh, *Binomial edge ideals and conditional independence statements*, Adv. Appl. Math., 45 (2010), pp. 317–333, doi:10.1145/96877.96907.

[28] C. J. Hillar and T. Windfeldt, *Algebraic characterization of uniquely vertex colorable graphs*, J. Combin. Theory Ser. B, 98 (2008), pp. 400–414.

[29] B. Huber and B. Sturmfels, *A polyhedral method for solving sparse polynomial systems*, Math. Comp., 64 (1995), pp. 1541–1555.

[30] Y. N. Lakshman, *On the complexity of computing a Gröbner basis for the radical of a zero dimensional ideal*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, ACM, New York, 1990, pp. 555–563.

[31] S. L. Lauritzen and D. J. Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems*, J. Roy. Statist. Soc. Ser. B, 50 (1988), pp. 157–224.

[32] T. Y. Li, *Numerical solution of multivariate polynomial systems by homotopy continuation methods*, Acta Numer., 6 (1997), pp. 399–436.

[33] J. A. Makowsky and K. Meer, *Polynomials of bounded treewidth*, in Foundations of Computational Mathematics, Proceedings of the Smalefest 2000, F. Cucker and M. Rojas, eds., World Scientific, Singapore, 2002, pp. 211–250.

[34] J. Nie and J. Demmel, *Sparse SOS relaxations for minimizing functions that are summations of small polynomials*, SIAM J. Optim., 19 (2008), pp. 1534–1558, doi:10.1137/060668791.

[35] S. Parter, *The use of linear graphs in Gauss elimination*, SIAM Rev., 3 (1961), pp. 119–130, doi:10.1137/1003021.

[36] V. I. Paulsen, S. C. Power, and R. R. Smith, *Schur products and matrix completions*, J. Funct. Anal., 85 (1989), pp. 151–178.

[37] A. Pothen and S. Toledo, *Elimination structures in scientific computing*, in Handbook on Data Structures and Applications, D. Mehta and S. Sahni, eds., CRC Press, Boca Raton, FL, 2004, pp. 59-1–59-29.

[38] H. Raddum and I. Semaev, *New Technique for Solving Sparse Equation Systems*, Report 2006/475, Cryptology ePrint Archive, http://eprint.iacr.org, 2006.
[39] D. J. Rose, *Triangulated graphs and the elimination process*, J. Math. Anal. Appl., 32 (1970), pp. 597–609.
[40] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, Singapore, 2005.
[41] W. A. Stein et al., *SageMath, Free Open-Source Mathematics Software System*, http://www.sagemath.org (accessed 06-20-16).
[42] B. Sturmfels, *Gröbner Bases and Convex Polytopes*, University Lecture Series 8, American Mathematical Society, Providence, RI, 1996.
[43] L. Vandenberghe and M. S. Andersen, *Chordal graphs and semidefinite optimization*, Found. Trends Optim., 1 (2015), pp. 241–433, doi:10.1561/2400000006.
[44] R. Villarreal, *Cohen-Macaulay graphs*, Manuscripta Math., 66 (1990), pp. 277–293, doi:10.1007/BF02568497.
[45] R. Villarreal, *Monomial Algebras*, 2nd ed., Monogr. Res. Notes Math. 8, CRC Press, Boca Raton, FL, 2015.