Exploiting Chordal Structure in Systems of Polynomial Equations

by

Diego Fernando Cifuentes Pardo

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Author .....
Department of Electrical Engineering and Computer Science
August 29, 2014

Certified by .....
Pablo Parrilo
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Theses

Exploiting Chordal Structure in Systems of Polynomial Equations

by

Diego Fernando Cifuentes Pardo

Submitted to the Department of Electrical Engineering and Computer Science
on August 29, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Chordal structure and bounded treewidth allow for efficient computation in linear algebra, graphical models, constraint satisfaction and many other areas. Nevertheless, it has not been studied whether chordality might also help solve systems of polynomials. We propose a new technique, which we refer to as chordal elimination, that relies in elimination theory and Gröbner bases. Chordal elimination can be seen as a generalization of sparse linear algebra. Unlike the linear case, the elimination process may not be exact. Nonetheless, we show that our methods are well-behaved for a large family of problems. We also use chordal elimination to obtain a good sparse description of a structured system of polynomials. By mantaining the graph structure in all computations, chordal elimination can outperform standard Gröbner basis algorithms in many cases. In particular, its computational complexity is linear for a restricted class of ideals. Chordal structure arises in many relevant applications and we propose the first method that takes full advantage of it. We demonstrate the suitability of our methods in examples from graph colorings, cryptography, sensor localization and differential equations.

Thesis Supervisor: Pablo Parrilo
Title: Professor

Acknowledgments

I would like to thank my advisor Pablo Parrilo for his guidance throughout this thesis. He is an accomplished researcher whose insightful comments have greatly helped shape this work. He challenged me to always do better. I am very thankful to Dina Katabi, who introduced me to many interesting projects and who offered me her support during my first year at MIT.

I am profoundly grateful to my professors from Los Andes University and from the Colombian Math Olympiads that helped me become an MIT student; especially to Federico Ardila, who continues motivating young students to pursue a career in Mathematics.

I would also like to thank all my friends, roommates and fellow Colombians at MIT that have made this time here very exciting. I am particularly grateful with Maru, who did not only read this document, but has showed me utmost support, has shared my enthusiasm for burrito and has made these years unforgettable.

Finally, I am very grateful for the love and support from my mother, brother, cousins, aunt, uncle and grandparents. I miss them all very much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Systems of polynomial equations can be used to model a large variety of applications. In most cases the systems arising have a particular sparsity structure, and exploiting such structure can greatly improve their efficiency. When all polynomials have degree one, we have the special case of systems of linear equations, which are represented with matrices. In such case, it is well known that under a *chordal structure* many matrix algorithms can be done efficiently [11, 27, 30].

Chordal graphs have many peculiar properties. The reason why they appear in solving linear equation is that chordal graphs have a *perfect elimination ordering*. If we apply Gaussian elimination to a symmetric matrix using such order, the sparsity structure of the matrix is preserved, i.e. no zero entries of the matrix become nonzero. Furthermore, if the matrix is positive semidefinite its Cholesky decomposition also preserves the structure. This constitutes the key to the improvements in systems with chordal structure. Similarly, many hard combinatorial problems can be solved efficiently for chordal graphs [22]. Chordal graphs are also a keystone in constraint satisfaction, graphical models and database theory [4, 13]. We address the question of whether chordality might also help solve nonlinear equations.

The most widely used method to work with nonlinear equations is given by the *Gröbner bases* theory [9]. Gröbner bases are a special representation of the same system

of polynomials which allows to extract many important properties of it. In particular, a Gröbner basis with respect to a *lexicographic order* allows to obtain the *elimination ideals* of the system. These elimination ideals provide a simple way to solve the system both analytically and numerically. Nevertheless, computing lexicographic Gröbner bases is very complicated in general.

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 graph parameter of the graph called the *treewidth* determines the complexity of many other hard problems [6], and it should influence polynomial equations as well. Nevertheless, standard algebraic geometry techniques do not relate to this graph. In this thesiswe link for the first time Gröbner bases theory with this graph structure of the system. We proceed to formalize our statements now.

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 the ordering of the variables $x_0 > x_1 > \cdots > x_{n-1}$. Given a system of polynomials $F = \{f_1, f_2, \ldots, f_m\}$ in the ring R, we associate to it a graph $G(F)$ with vertex set $V = \{x_0, \ldots, x_{n-1}\}$. Such graph is given by a union of cliques: for each f_i we form a clique in all 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 constrained satisfaction problems, $G(F)$ is usually called the primal constraint graph [13].

Throughout this document we fix a polynomial ideal $I \subseteq R$ with a given set of generators F. We associate to I the graph $G(I) := G(F)$, which we assume to be a chordal graph. Even more, we assume that $x_0 > \cdots > x_{n-1}$ is a perfect elimination ordering of the graph (see Definition 2.1). In the event that $G(I)$ is not chordal, the same reasoning applies by considering a chordal completion. We want to compute the elimination ideals of I, denoted as $\text{elim}_l(I)$, 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 $\mathbf{V}(I_l) = \mathbf{V}(\text{elim}_l(I))$.

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

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

Example 1.1 (Gröbner bases may destroy chordality)**.** Let $I = \langle x_0x_2 - 1, x_1x_2 - 1\rangle$, whose associated graph is the path x_0—x_2—x_1. Note that any Gröbner basis must contain the polynomial $p = x_0 - x_1$, breaking the sparsity structure. Nevertheless, we can find some generators for its first elimination ideal $\mathrm{elim}_1(I) = \langle x_1x_2 - 1\rangle$, that preserve such structure.

As evidenced in Example 1.1, a Gröbner basis with the same graph structure might not exist, but we might still be able to find its elimination ideals. In this thesis, we introduce a new method that attempts to find ideals I_l as proposed above. We refer to this method as *chordal elimination* and it is presented in Algorithm 3.1. 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.

As opposed to sparse Gaussian elimination, in the general case chordal elimination does may not lead to the right elimination ideals. Nevertheless, we can give inner and outer approximations to the actual elimination ideals, as shown in Theorem 3.3. Therefore, we can also give guarantees when the elimination ideals found are right. We prove that for a large family of problems, which includes the case of linear equations, chordal elimination gives the *exact* elimination ideals. In particular, Theorem 4.7 shows that generic dense ideals belong to this family.

The aim of chordal elimination is to obtain a good description of the ideal (e.g. a Gröbner basis), but at the same time preserve 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 big 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 basis. We argue that such representation is given by finding the elimination ideals of all *maximal cliques* of the graph. We attempt to find this *best sparse description* in Algorithm 5.1. In case 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 $\mathbf{V}(I) = \{p_1, \ldots, p_k\}$ is finite, the complexity depends intrinsically on the size of the projection $|\pi_l(p_1, \ldots, p_k)|$. As such, it performs much better if such set is small. In Theorem 6.4 we show complexity bounds for certain family of ideals were this condition is met. Specifically, we show that chordal elimination is $O(n)$ when we fix the size of the maximal clique κ. This parameter κ is usually called the *treewidth* of the graph. This linear behavior is reminiscent to many other graph problems (e.g. Hamiltonian circuit, vertex colorings, vertex cover) which are NP-hard in general, but are tractable for fixed κ [6]. Our methods provide an algebraic alternative for some of these problems.

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, we can easily prove that solving zero dimensional quadratic equations of treewidth 2 is already NP-hard, as seen in the following example.

Example 1.2 (Polynomials of treewidth 2 are hard)**.** Let $A = \{a_1, \ldots, a_n\} \subseteq \mathbb{Z}^n$ be arbitrary, the Subset Sum problem asks for a nonempty subset of A with zero sum. It is known to be NP-complete. The following is a polynomial formulation of this problem:

$$
\begin{aligned}
& s_0 = s_n = 0 && \\
& s_i = s_{i-1} + a_i x_i, && \text{for } 1 \leq i \leq n \\
& x_i^2 = x_i, && \text{for } 1 \leq i \leq n
\end{aligned}
$$

The natural ordering of the variables $s_0 > x_1 > s_1 > \cdots > x_n > s_n$ is a perfect

elimination ordering of the associated graph, that has treewidth 2.

Chordal structure arises in many different applications and we believe that algebraic geometry algorithms should take advantage of it. The last part of this thesismentions some of such applications that include cryptography, sensor localization and differential equations. In all these cases we show the advantages of chordal elimination over standard Gröbner bases algorithms, using both lex and degrevlex term orderings. In some cases, we show that we can also find a lex Gröbner basis faster than with degrevlex ordering, by making use of chordal elimination. This contradicts the standard heuristic preferring degrevlex ordering.

We now summarize our contributions. We present a new algorithm that exploits chordal structure in systems of polynomial equations. Even though this method might fail in some cases, we show that it works well for a very large family of problems. We also study the question of finding a good sparse description of an ideal with chordal structure, and argue that our methods provide such description. We show that the complexity of our algorithm is linear for a restricted class of problems. We illustrate the experimental improvements of our methods in different applications. To our knowledge, we are the first to relate graph theoretical ideas of chordal structure with computational algebraic geometry.

The document is structured as follows. In chapter 2 we provide a brief introduction to chordal graphs and we recall some ideas from algebraic geometry. In chapter 3 we present our main method, chordal elimination. This method returns some ideals I_l which are an inner approximation to the elimination ideals $\text{elim}_l(I)$. Chapter 4 presents some types of systems under which chordal elimination is exact, in the sense that $\mathbf{V}(I_l) = \mathbf{V}(\text{elim}_l(I))$. In chapter 5, we use chordal elimination to provide a good sparse description of the ideal. We show that this description is close to a lex Gröbner basis of the ideal, by sharing many structural properties with it. In chapter 6 we analyze the computational complexity of the algorithms proposed for a certain class of problems. We show that these algorithms are linear in n for fixed treewidth. Finally,

chapter 7 presents an experimental evaluation of our methods. It is shown that for many applications chordal elimination performs better than standard Gröbner bases algorithms.

1.1 Related work

Chordal graphs

Chordal graphs have many peculiar properties and appear in many different applications. In fact, many hard problems can be solved efficiently when restricted to chordal graphs or nearly chordal graphs (graphs with small treewidth) [22]. In particular, classical graph problems such as vertex colorings, Hamiltonian cycles, vertex covers, weighted independent set, etc. can be solved in linear time for graphs with bounded treewidth [6]. Likewise, constraint satisfaction problems become polynomial-time solvable when the treewidth of the primal constraint graph is bounded [12, 13]. In other words, many hard problems are fixed-parameter-tractable when they are parametrized by the treewidth.

In a similar fashion, many sparse matrix algorithms can be done efficiently when the underlying graph has a chordal structure. If we apply Gaussian elimination to a symmetric matrix using a perfect elimination order of the graph, the sparsity structure of the matrix is preserved, i.e. no zero entries of the matrix become nonzero [30]. In the same way, Cholesky decomposition can also be performed while preserving the structure. This constitutes the key to several techniques in numerical analysis and optimization with chordal structure [11, 27, 28]. We follow a similar approach to generalize these ideas to nonlinear equations.

Structured polynomials

Solving structured systems of polynomial equations is a well-studied question. Many past techniques make use of different types of structure, although they do not address the special case of chordality. In particular, past work explores the use of symmetry,

multi-linear structure, multi-homogeneous structure, and certain types of sparsity [10, 34]. For instance, recent work on Gröbner bases by Faugère et. al. makes use of symmetry [17], and multi-homogeneous structure [18, 19].

An intrinsic measure of complexity for polynomial equations is the number of solutions. Indeed, homotopy methods depend strongly on a good rootcount [25]. The Bezout bound provides a simple bound that depends solely on the degrees of the equations. Taking into account the sparsity leads to the better BKK bound, which uses a polytopal abstraction of the system. This constitutes the key to the theory of sparse resultants and toric geometry [32, 33]. A recent paper adapts such ideas to Gröbner bases [19]. Nonetheless, these type of methods do not take advantage of the chordal structure we study in this thesis. On the other hand, our approach relates the complexity to the treewidth of the underlying graph, as seen in chapter 6.

A different body of methods come from the algebraic cryptanalysis community, where they deal with very sparse equations over small finite fields. One popular approach is converting the problem into a SAT problem and use SAT solvers [2]. A different idea is seen in [29], 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 work for positive dimension and arbitrary fields.

Chapter 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 2.1. Let G be a graph with vertices $x_0, \ldots, x_{n-1}$. 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

$$X_l := \{x_l\} \cup \{x_m : x_m \text{ is adjacent to } x_l,\ x_m < x_l\} \tag{2.1}$$

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

Chordal graphs have many interesting properties. Observe, for instance, that the number of maximal cliques is at most n. The reason is that 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 [30]. A classic and simple algorithm to do so is *Maximum Cardinality Search* (MCS) [5].

This algorithm successively selects a vertex with maximal number of neighbors among previously chosen vertices, as shown in Algorithm 2.1. The ordering obtained is a reversed perfect elimination ordering.

Algorithm 2.1 Maximum Cardinality Search [5]

Input: A chordal graph G with vertex set V
Output: A reversed perfect elimination ordering σ

```
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 2.2. 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].

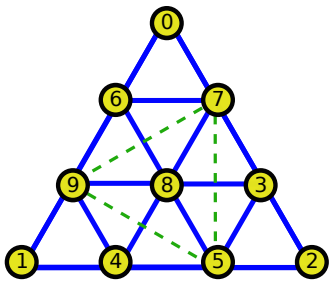

Figure 2-1: 10-vertex graph (blue, solid) and a chordal completion (green, dashed).

Example 2.1. Let G be the blue/solid graph in Figure 2-1. 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 document that the graph $G(I)$ is chordal and the ordering of its vertices (inherited from the polynomial ring) is a perfect elimination ordering. However, for a non-chordal graph G the same results hold by considering a chordal completion.

2.2 Algebraic geometry

We use standard tools from algebraic geometry, following the notation from [9]. For sake of completeness, we include here a brief definition of the main concepts needed. However, we assume some familiarity with Gröbner bases and elimination theory.

Definition 2.3. A *field* $\mathbb{K}$ is a set together with two binary operations, addition "+" and multiplication "$\cdot$", that satisfy the following axioms:

- both addition and multiplication are commutative and associative.
- multiplication distributes over addition.
- there exist an additive identity 0 and a multiplicative identity 1.
- every (nonzero) element has an additive (multiplicative) inverse.

Simple examples of fields include the rationals $\mathbb{Q}$, the reals $\mathbb{R}$ and the complex numbers $\mathbb{C}$. Another example are *finite fields* (or Galois fields), denoted as $\mathbb{F}_q$, where q is a power of a prime. If p is a prime then $\mathbb{F}_p$ is simply arithmetic modulo p.

A field is said to be *algebraically closed* if every univariate nonconstant polynomial with coefficients in $\mathbb{K}$ has a root. In the examples above, $\mathbb{C}$ is the only one algebraically closed. However, any field $\mathbb{K}$ can be embedded into an algebraically closed field $\overline{\mathbb{K}}$.

Definition 2.4. The *polynomial ring* $R = \mathbb{K}[x_0, \ldots, x_{n-1}]$ over a field $\mathbb{K}$, is the set of all polynomials in variables $x_0, \ldots, x_{n-1}$ and coefficients in $\mathbb{K}$, together with ordinary addition and multiplication of polynomials.

Definition 2.5. Let R be a polynomial ring. An *ideal* is a set $\emptyset \subsetneq I \subseteq R$ such that:

- $p + q \in I$ for all $p, q \in I$.
- $p \cdot r \in I$ for all $p \in I$ and all $r \in R$.

Given an arbitrary set of polynomials $F = \{f_1, f_2, \ldots\} \subseteq R$ (possibly infinite) we can associate to it an ideal. The *ideal generated* by F is

$$\langle F \rangle = \langle f_1, f_2, \ldots \rangle := \{f_{i_1} r_1 + \cdots + f_{i_k} r_k : \text{for } k \in \mathbb{N} \text{ and } f_{i_j} \in F,\, r_j \in R,\, 1 \leq j \leq k\}$$

The following definition is the starting point of a correspondence between algebraic objects (ideals) and geometry (varieties).

Definition 2.6. Let $I \subseteq R$ be an ideal. The *variety* of I is the set

$$\mathbf{V}(I) := \{(s_0, \ldots, s_{n-1}) \in \mathbb{K}^n : p(s_0, \ldots, s_{n-1}) = 0 \text{ for all } p \in I\}$$

We say that I is *zero dimensional* if $\mathbf{V}(I)$ is finite.
Conversely, given an arbitrary set $S \subseteq \mathbb{K}^n$, the *vanishing ideal* of S is

$$\mathbf{I}(S) := \{p \in R : p(s_0, \ldots, s_{n-1}) = 0 \text{ for all } (s_0, \ldots, s_{n-1}) \in S\}$$

It is easy to check that if $I = \langle F \rangle$ for some system of polynomials F, then $\mathbf{V}(I)$ is precisely the zero set of F.

We define now some standard operations on ideals to obtain other ideals.

Definition 2.7. Let $I \subseteq R$ be an ideal. The *radical* of I is the following ideal

$$\sqrt{I} := \{p : p^k \in I \text{ for some integer } k \geq 1\}$$

The l-th *elimination ideal* is the set

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

Given ideals I, J the *ideal quotient* is the set

$$(I : J) := \{r \in R : rJ \subseteq I\}$$

There is a geometric interpretation to all the operations defined above. We assume from now that $\mathbb{K}$ is algebraically closed. To begin with, Hilbert's Nullstellensatz states:

$$\mathbf{I}(\mathbf{V}(I)) = \sqrt{I} \subseteq I$$

In other words, $\sqrt{I}$ is the largest ideal that has the same zero set as I. We say that I is radical if $\sqrt{I} = I$. The property above implies a one to one correspondence between radical ideals and varieties.

We denote $\pi_l : \mathbb{K}^n \to \mathbb{K}^{n-l}$ to the *projection* onto the last $n - l$ coordinates. There is a correspondence between elimination ideals and projections:

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

where $\overline{S}$ denotes the *smallest variety* containing S. It is also called the *Zariski closure* of S, as we can define a topology where varieties are the closed sets.

Finally, the geometric analog of the ideal quotient is *set difference*:

$$\mathbf{I}(S) : \mathbf{I}(T) = \mathbf{I}(S \setminus T)$$

for arbitrary sets S, T.

We now proceed to define Gröbner bases. A monomial is a term of the form $x_0^{a_0} x_1^{a_1} \cdots x_{n-1}^{a_{n-1}}$. We write it in vector notation as x^α, where $\alpha = (a_0, \ldots, a_{n-1})$.

Definition 2.8. Let R be a polynomial ring. A *monomial term ordering* $\succ$ is a total (or linear) order on the monomials of R such that:

- if $x^\alpha \succ x^\beta$ then $x^\gamma x^\alpha \succ x^\gamma x^\beta$, for any monomials $x^\alpha, x^\beta, x^\gamma$.
- $\succ$ is a well-ordering.

The simplest example of a monomial ordering is the *lexicographic* (or *lex*) term order with $x_0 \succ \cdots \succ x_{n-1}$. In this order, $x^\alpha \succ x^\beta$ if and only if the leftmost nonzero entry of $\alpha - \beta$ is positive. For instance, $x_0 x_1^2 \succ x_1^3 x_2^4$ and $x_0^3 x_1^2 x_2^4 \succ x_0^3 x_1^2 x_2$.

Another important monomial ordering is the *degree reverse lexicographic* (or *degrevlex*) term order.

Definition 2.9. Let $\succ$ be a monomial term ordering of R. The *leading monomial* $lm(p)$ of a polynomial $p \in R$ is the largest monomial of p with respect to $\succ$. Let I be an ideal, its *initial ideal* is the set $in(I) := \langle lm(p) : p \in I \rangle$

Definition 2.10. Let $\succ$ be a monomial term ordering of R, let I be an ideal. A finite set $gb_I \subseteq I$ is a *Gröbner basis* of I (with respect to $\succ$) if $in(I) = \langle lm(p) : p \in gb_I \rangle$.
A Gröbner basis gb_I is said to be *minimal* if the leading coefficient of each $p \in gb_I$ is 1 and if the monomial $lm(p)$ is not in the ideal $\langle lm(q) : q \in gb_I \setminus \{p\} \rangle$.
A Gröbner basis gb_I is said to be *reduced* if the leading coefficient of each $p \in gb_I$ is 1 and if no monomial of p is in the ideal $\langle lm(q) : q \in gb_I \setminus \{p\} \rangle$.

It is a consequence of Hibert's Basis Theorem that there always exists a Gröbner basis. Even more, there is a unique reduced Gröbner basis. The oldest method to compute Gröbner bases is Buchberger's algorithm, and most modern algorithms are based on it.

An important property of lex Gröbner bases is its relation to elimination ideals. Given a lex Gröbner basis gb_I, then

$$\mathrm{elim}_l(I) = \langle gb_I \cap \mathbb{K}[x_l, \ldots, x_{n-1}] \rangle$$

Even more, $gb_I \cap \mathbb{K}[x_l, \ldots, x_{n-1}]$ is a lex Gröbner basis of it. This allows to find elimination ideals by means of Gröbner bases computation. Usually Gröbner bases for lex orderings are more complex than for degrevlex orderings. Thus, a standard approach is to first find a degrevlex Gröbner basis, and then use a term order conversion algorithm (e.g. FGLM [16]) to obtain the lex Gröbner basis.

Finally, we give a brief description of resultants. Let $f, g \in \mathbb{K}[x_0, \ldots, x_{n-1}]$ be polynomials of positive degree d, e in x_0. Assume they have the form

$$\begin{aligned} f &= a_d x_0^d + a_{d-1} x_0^{d-1} + \cdots + a_1 x_0 + a_0, & a_d \neq 0 \\ g &= b_e x_0^e + b_{e-1} x_0^{e-1} + \cdots + b_1 x_0 + b_0, & b_e \neq 0 \end{aligned}$$

where a_i, b_j are polynomials in $\mathbb{K}[x_1, \ldots, x_{n-1}]$. The *resultant* of f, g with respect to x_0 is the resultant of a $(d+e) \times (d+e)$ matrix:

$$\mathrm{Res}_{x_0}(f,g) := \det \begin{bmatrix} a_d & a_{d-1} & \cdots & a_1 & a_0 & & & \\ & a_d & a_{d-1} & & a_1 & a_0 & & \\ & & \ddots & & & \ddots & \ddots & \\ & & & & & & a_1 & a_0 \\ b_e & b_{e-1} & \cdots & b_1 & b_0 & & & \\ & b_e & b_{e-1} & & b_1 & b_0 & & \\ & & \ddots & & & \ddots & \ddots & \\ & & & & & & b_1 & b_0 \end{bmatrix}$$

where the first e rows only involve a_i's and the following d rows only involve b_j's. The main property of the resultant is that it is zero if and only if f, g have a common divisor with positive degree in x_0. Resultants are also related to elimination ideals:

$$\mathrm{Res}_{x_0}(f,g) \in \mathrm{elim}_1(\langle f, g \rangle)$$

Chapter 3

Chordal elimination

In this chapter, we present our main method, chordal elimination. As mentioned before, 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 ideally it will be close to it. In fact, we will find inner and outer approximations to the ideal, as will be seen later. In the case that both approximations are the same we will be sure that the elimination ideal was computed correctly. The following example illustrates why elimination may not be exact.

Example 3.1 (Incremental elimination may fail)**.** Consider the ideal $I = \langle x_0x_2 + 1, x_1^2 + x_2, x_1 + x_2, x_2x_3\rangle$. The associated graph is the tree in Figure 3-1. We show now an incremental attempt to eliminate variables in a similar way as in Gaussian elimination.

First we consider only the polynomials involving x_0, there is only one: $x_0x_2 + 1$. Thus, we cannot eliminate x_0. We are left with the ideal $I_1 = \langle x_1^2+x_2, x_1+x_2, x_2x_3\rangle$. We now consider the polynomials involving x_1; there are two: $x_1^2 + x_2, x_1 + x_2$. Eliminating x_1, we obtain $x_2^2 + 2$. We get the ideal $I_2 = \langle x_2^2 + x_2, x_2x_3\rangle$. We cannot eliminate x_2

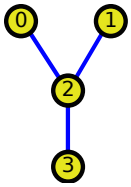

Figure 3-1: Simple 3-vertex tree.

from these equations. We got the following approximations for the elimination ideals:

$$I_1 = \langle x_1^2 + x_2, x_1 + x_2, x_2 x_3 \rangle$$
$$I_2 = \langle x_2^2 + x_2, x_2 x_3 \rangle$$
$$I_3 = \langle 0 \rangle$$

The unique reduced Gröbner basis for this system is $gb = \{x_0 - 1, x_1 - 1, x_2 + 1, x_3\}$, and thus none of the elimination ideals were computed correctly. The problem in this case is that the first equation implies that $x_2 \neq 0$, which could be used to reduce the equations obtained later.

Example 3.1 shows the basic idea we follow. It also shows that we might not obtain the exact elimination ideals. However, we can always characterize what was wrong; in Example 3.1 the problem was the implicit equation $x_2 \neq 0$. This allows us to provide bounds for the elimination ideals computed, and give guarantees when the elimination ideals are computed correctly.

3.1 Squeezing elimination ideals

Our goal now is to find inner and outer approximations (or bounds) to the elimination ideals of I. We start with the first elimination ideal, and then proceed to further elimination ideals. To do so, the key will be the Closure Theorem [9, Chapter 3].

Definition 3.1. 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

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

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

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

Theorem 3.1 (Closure Theorem)**.** *Let* $I = \langle f_1, \ldots, f_s \rangle \subseteq \mathbb{K}[x_0, \ldots, x_{n-1}]$. *Denote* $V := \mathbf{V}(I) \subseteq \mathbb{K}^n$ *and let* $W = \mathbf{V}(\text{coeff}_1(I)) \subseteq \mathbb{K}^{n-1}$ *be the variety of the coefficient ideal. Let* $\text{elim}_1(I)$ *be the first elimination ideal, and let* $\pi : \mathbb{K}^n \to \mathbb{K}^{n-1}$ *be the natural projection. Then:*

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

The next lemma provides us with an inner approximation I_1 to the first elimination ideal $\text{elim}_1(I)$. It also describes an outer approximation to it, which depends on I_1 and some variety W. If the two bounds are equal, this implies that we obtain the exact elimination ideal.

Lemma 3.2. *Let* $J = \langle f_1, \ldots, f_s \rangle \subseteq \mathbb{K}[x_0, \ldots, x_{n-1}]$, *let* $K = \langle g_1, \ldots, g_r \rangle \subseteq \mathbb{K}[x_1, \ldots, x_{n-1}]$ *and let*

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

Let $W := \mathbf{V}(\text{coeff}_1(J) + K) \subseteq \mathbb{K}^{n-1}$ *and let* $I_1 := \text{elim}_1(J) + K \subseteq \mathbb{K}[x_1, \ldots, x_{n-1}]$.

Then

$$\mathbf{V}(\mathrm{elim}_1(I)) = \overline{\pi(\mathbf{V}(I))} \subseteq \mathbf{V}(I_1) \tag{3.1}$$

$$\mathbf{V}(I_1) - W \subseteq \pi(\mathbf{V}(I)) \tag{3.2}$$

Proof. We first show equation (3.1). The Closure Theorem says that $\mathbf{V}(\mathrm{elim}_1(I)) = \overline{\pi(\mathbf{V}(I))}$. We will show that $\pi(\mathbf{V}(I)) \subseteq \mathbf{V}(I_1)$. Nevertheless, as $\mathbf{V}(I_1)$ is closed, that would imply equation (3.1).

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 $\overline{\pi(\mathbf{V}^n(J))} = \mathbf{V}(\mathrm{elim}_1(J))$. Combining everything we conclude:

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

We now show equation (3.2). The Closure Theorem states that $\mathbf{V}(\text{elim}_1(J)) - \mathbf{V}(\text{coeff}_1(J)) \subseteq \pi(\mathbf{V}(J))$. Then

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

This concludes the proof. □

The ideal I_1 is an approximation to the ideal $\text{elim}_1(I)$. The variety W provides a bound on the error of such approximation. In particular, if W is empty then I_1 and $\text{elim}_1(I)$ determine the same variety. Observe that to compute both I_1 and W we only do operations on the generators of J, which are the polynomials that involve x_0, we never deal with K. As a result, we are indeed preserving the chordal structure of the system. We elaborate more on this later.

As the set difference of varieties corresponds to ideal quotient, we can express the bounds above in terms of ideals:

$$\sqrt{I_1} : \mathbf{I}(W) \supseteq \sqrt{\text{elim}_1(I)} \supseteq \sqrt{I_1}$$

where $\mathbf{I}(W)$ is the radical ideal associated to W.

We can generalize the previous lemma to further elimination ideals, as we do now.

Theorem 3.3 (Chordal elimination)**.** *Let $I \subseteq \mathbb{K}[x_0, \ldots, x_{n-1}]$ be a sparse ideal. Consider the following procedure:*

i. Let $I_0 := I$ and $l := 0$.

ii. 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 [1] such that $I_l = J_l + K_{l+1}$.

iii. Let $W_{l+1} = \mathbf{V}(\text{coeff}_{l+1}(J_l) + K_{l+1}) \subseteq \mathbb{K}^{n-l-1}$.

[1] Note that we are not fully defining the ideals J_l, K_{l+1}, we have some flexibility.

iv. *Let* $I_{l+1} := \text{elim}_{l+1}(J_l) + K_{l+1} \subseteq \mathbb{K}[x_{l+1}, \ldots, x_{n-1}]$

v. *Go to ii with* $l := l + 1$.

Then the following equation holds for all l.

$$\mathbf{V}(\text{elim}_l(I)) = \overline{\pi_l(\mathbf{V}(I))} \subseteq \mathbf{V}(I_l) \tag{3.3}$$

$$\mathbf{V}(I_l) - (\pi_l(W_1) \cup \cdots \cup \pi_l(W_l)) \subseteq \pi_l(\mathbf{V}(I)) \tag{3.4}$$

Proof. We prove it by induction on l. The base case is Lemma 3.2. Assume that the result holds for some l and let's show it for $l + 1$.

By induction hypothesis $I_l, W_1, \ldots, W_l$ satisfy equations (3.3), (3.4). Lemma 3.2 with I_l as input tell us that I_{l+1}, W_{l+1} satisfy:

$$\overline{\pi(\mathbf{V}(I_l))} \subseteq \mathbf{V}(I_{l+1}) \tag{3.5}$$

$$\mathbf{V}(I_{l+1}) - W_{l+1} \subseteq \pi(\mathbf{V}(I_l)) \tag{3.6}$$

where $\pi : \mathbb{K}^{n-l} \to \mathbb{K}^{n-l-1}$ is the natural projection. 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 equation (3.3).

We also have

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

which proves equation (3.4). □

Observe that the procedure described in Theorem 3.3 is not fully defined (see step ii). However, we will use this procedure as the basis to construct the chordal elimination algorithm in section 3.2.

Theorem 3.3 gives us lower and upper bounds for the elimination ideals. We can reformulate these bounds in terms of ideals:

$$\sqrt{I_L} : \mathbf{I}(W) \supseteq \sqrt{\mathrm{elim}_L(I)} \supseteq \sqrt{I_L} \tag{3.7}$$

where $W := \pi_L(W_1) \cup \cdots \cup \pi_L(W_L)$, so that

$$\mathbf{I}(W) = \mathrm{elim}_L(\mathbf{I}(W_1)) \cap \cdots \cap \mathrm{elim}_L(\mathbf{I}(W_L)) \tag{3.8}$$

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

3.2 Chordal elimination algorithm

As mentioned earlier, the procedure in Theorem 3.3 is not fully defined yet. In particular, the procedure in Theorem 3.3 does not specify how to obtain the decomposition $I_l = J_l + K_{l+1}$. It only specifies that $J_l \subseteq \mathbb{K}[x_l, \ldots], K_{l+1} \subseteq \mathbb{K}[x_{l+1}, \ldots]$. Indeed, there are several valid approaches to do so, in the sense that they preserve the chordal structure in the system. We now describe the approach that we follow to obtain the chordal elimination algorithm.

We recall the definition of the cliques X_l from equation (2.1). 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 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*.

Example 3.2. 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 sets $J_0 = \langle f, g\rangle$, $K_1 = \langle h\rangle$.

Observe that the clique decomposition attempts to include as many polynomials in J_l, while guaranteeing that we do not change the sparsity pattern. It is easy to see that the procedure in Theorem 3.3, using this clique decomposition, preserves chordality. We state that now.

Proposition 3.4. *Let I be an ideal with chordal graph G. If we use the algorithm in Theorem 3.3 with 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. □

After these comments, we solved the ambiguity problem of step ii in Theorem 3.1. However, there are still some issues while computing the "error" variety W of equation (3.8). We discuss that now.

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 depends on the specific set of generators that we are using. In particular, some set of generators might lead to a larger/worse variety W_{l+1} than others. This problem is inherent to the Closure theorem, and it is discussed in [9, Chapter 3]. It turns out that a lex Gröbner basis of J_l is an optimal set of generators, as shown in [9].

Algorithm 3.1 presents the chordal elimination algorithm, that uses the clique decomposition. The output of the algorithm is the inner approximation I_L to the L-th elimination ideal and the ideals associated to the varieties $W_1, \ldots, W_L$, that satisfy (3.7).

We should add another remark regarding the lower bound W of (3.7). Algorithm 3.1 provides us varieties W_l for each l. However, if we want to find the outer approximation

Algorithm 3.1 Chordal Elimination Ideal

Input: An ideal I with chordal graph G and an integer L
Output: Ideal I_L and ideals $W_1, \ldots, W_L$ satisfying (3.7)

1: **procedure** ChordElim(I, G, L)
2: $I_0 = I$
3: **for** $l = 0 : L-1$ **do**
4: get clique X_l of G
5: $J_l, K_{l+1} =$ Decompose(I_l, X_l)
6: FindElim&Coeff(J_l)
7: $I_{l+1} = \text{elim}_{l+1}(J_l) + K_{l+1}$
8: $W_{l+1} = \text{coeff}_{l+1}(J_l) + K_{l+1}$
9: **return** $I_L, W_1, \ldots, W_L$
10: **procedure** Decompose(I_l, X_l)
11: $J_l = \langle f : f \text{ generator of } I_l \text{ and } f \in \mathbb{K}[X_l] \rangle$
12: $K_l = \langle f : f \text{ generator of } I_l \text{ and } f \notin \mathbb{K}[X_l] \rangle$
13: **procedure** FindElim&Coeff(J_l)
14: append to J_l its lex Gröbner basis
15: $\text{elim}_{l+1}(J_l) = \langle f : f \text{ generator of } J_l \text{ with no } x_l \rangle$
16: $\text{coeff}_{l+1}(J_l) = \langle \text{leading coefficient of } f : f \text{ generator of } J_l \rangle$

to $\text{elim}_L(I)$ we need to compute the projections $\pi_L(W_l)$, as seen in (3.7). In the event that $W_l = \emptyset$, then the projection is trivial. This is the case that we focus on for the rest of the thesis. Otherwise, we need to compute the L-th elimination ideal of $\mathbf{I}(W_l)$. As $\mathbf{I}(W_l)$ preserves the chordal structure of I, it is natural to use chordal elimination again on each $\mathbf{I}(W_l)$. Let $W_{l,L}$ be the outer approximation to $\pi_L(W_l)$ that we obtain by using chordal elimination, i.e.

$$\mathbf{I}(W_{l,L}) := \text{ChordElim}(\mathbf{I}(W_l), L)$$

Thus $W_{l,L} \supseteq \pi_L(W_l)$, so that (3.7) still holds with $\hat{W} := W_{1,L} \cup \cdots \cup W_{L,L}$.

Finally, observe that in line 14 of Algorithm 3.1 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 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 we

append to them this Gröbner basis. We will explain the reason to do that in section 3.3.

3.3 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 3.2. 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 x_l is *a descendant* of x_p. Note that T is rooted at x_{n-1}.

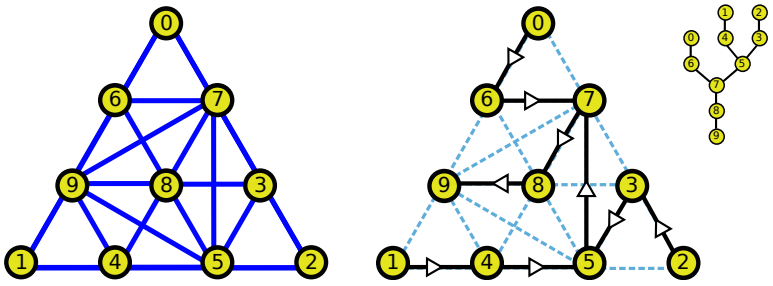

Figure 3-2: Chordal graph G and its elimination tree T.

Figure 3-2 shows an example of the elimination tree of a given graph. It is easy 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 tree.

Lemma 3.5. *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 equation (2.1).

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 property, we must have $C \subseteq X_p$. □

A consequence of the lemma above is the following relation:

$$\text{elim}_l(I \cap \mathbb{K}[X_l]) \subseteq I \cap \mathbb{K}[X_p] \tag{3.9}$$

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

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

There is a simple geometric interpretation of equation (3.9). 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, equation (3.9) 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 3.1. To preserve the property above (i.e. every partial solution of X_p extends to X_p), we would like to have the following relation:

$$\text{elim}_{l+1}(J_l) \subseteq J_p \tag{3.10}$$

It turns out that there is a very simple way to have 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 3.1 we *append* a Gröbner basis to J_l.

We prove now that equation (3.10) holds. We need one lemma before.

Lemma 3.6. *In Algorithm 3.1, let $f \in I_l$ be one of its generators. Assume that $x_m \leq x_l$*

is such that all variables of f are in 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 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 $g \in K_{l+1}$. In the second case, f is a generator of I_{l+1} and using of the induction hypothesis we get $f \in J_m$. In the first case, as $f \in \mathbb{K}[X_m]$ then all variables of f are less or equal to x_m, and thus strictly smaller than x_l. Following Algorithm 3.1, 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, then $x_m \in X_l$. Thus, X_{lm} is a clique of $G|_{\{x_m,\ldots,x_{n-1}\}}$ and it 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 3.7. *In Algorithm 3.1, for any x_l if x_p denotes its parent in T, then equation* (3.10) *holds.*

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 $X_l \backslash \{x_l\} \subseteq X_p$, where we used Lemma 3.5. From Lemma 3.6 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 (3.10).

Example 3.3. 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 3.1), 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 $x_2^2 \notin \hat{J}_2$, even though $x_2^2 \in J_0$. Moreover, the ideal J_0 is zero dimensional, but $\hat{J}_2$ has positive dimension. Thus, equation (3.10) does not hold.

Chapter 4

Exact elimination

Notation. We are mainly interested in the zero sets of the ideals. To simplify the writing, we abuse notation by writing $I_1 = I_2$ whenever we have $\mathbf{V}(I_1) = \mathbf{V}(I_2)$.

In chapter 3 we showed an algorithm that gives us an approximate elimination ideal. In this chapter we are interested in finding conditions under which such algorithm returns the actual elimination ideal. We will say that our estimated elimination ideal I_l is *exact* whenever we have $\mathbf{V}(I_l) = \mathbf{V}(\text{elim}_l(I))$. Following the convention above, we abuse notation and simply write $I_l = \text{elim}_l(I)$

Theorem 3.3 gives us lower and upper bounds on the actual elimination ideals. Clearly, if the two bounds are the same we actually obtain the exact elimination ideal. The simplest case in which the two bounds obtained are equal is when $W_l = \emptyset$ for all l. We clearly state this condition now.

Lemma 4.1 (Exact elimination)**.** *Let I be an ideal and assume that in Algorithm 3.1 we have that for each $l < L$ there is a $f_l \in J_l$ such that its leading monomial is a pure power of x_l. Then $W_l = \emptyset$ and $I_l = \text{elim}_l(I)$ for all $l \leq L$.*

Proof. Assume that the leading monomial of f_l is x_l^d for some d. As $f_l \in J_l$, then x_l^d is in the initial ideal $in(J_l)$. Thus, there must be a g that is part of the Gröbner basis of J_l such that its leading monomial is $x_l^{d'}$ for some $d' \leq d$. The coefficient u_t that

corresponds to such g is $u_t = 1$, and therefore $W_l = \mathbf{V}(1) = \emptyset$. Thus the two bounds in Theorem 3.3 are the same and we conclude that the elimination ideal is exact. □

Remark. Note that the polynomial f_l in the lemma need not be a generator of J_l.

The previous lemma is quite simple, but it will allow us to find some classes of ideals in which we can perform exact elimination, while preserving the structure. We now present a simple consequence of the lemma above.

Corollary 4.2. *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 $W_l = \emptyset$ and $I_l = \mathrm{elim}_l(I)$ for all l.*

Proof. Let x_j be arbitrary, and let $x_l \geq x_j$ be such that $X_j \subseteq X_l$ and X_l is a maximal clique. As $J_l \subseteq \mathbb{K}[X_l]$ is zero dimensional then its Gröbner basis must have a polynomial f with leading monomial of the form $x_j^{d_j}$. Such polynomial becomes part of the generators of J_l. From Lemma 3.6 we obtain that $f \in J_j$, and thus Lemma 4.1 applies. □

Corollary 4.3. *Let I be an ideal and assume that for each $l < L$ there is a generator of I with leading monomial $x_l^{d_l}$ for some d_l. Then $W_l = \emptyset$ and $I_l = \mathrm{elim}_l(I)$ for all $l \leq L$.*

Proof. Let f be a generator of I such that its leading monomial is $x_l^{d_l}$. It follows from Lemma 3.6 that $f \in J_l$, and thus Lemma 4.1 applies. □

The previous corollary presents a first class of ideals for which we are guaranteed to obtain exact 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.

However, in many cases there might be some l such that the condition above does not hold. In particular, if $l = n - 2$ the only way that such condition holds is if there is a polynomial that only involves x_{n-2}, x_{n-1}. We will now see that under some weaker conditions on the ideal, we will still be able to apply Lemma 4.1.

Definition 4.1. Let $f \in \mathbb{K}[x_0, \ldots, x_{n-1}]$ be such that for each $0 \leq l < n$ the monomial m_l of f with largest degree in x_l is of the form $m_l = x_l^{d_l}$ for some d_l. We say that f is *simplicial*.

The reason to call such polynomial simplicial is that the standard simplex

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

where X_f are the variables of f, is a face of the Newton polytope of f and they are the same if f is homogeneous. Note, for instance, that linear equations are always simplicial.

We will also assume that the coefficients of $x_l^{d_l}$ are generic, in a sense that will be clear in the next lemma.

Lemma 4.4. *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 as univariate polynomials in x. As q_2 is simplicial, for each $x_i \in X_2 \setminus x$ the monomial with 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 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 3.2). The following lemma tells us that if among the descendants of x_l in T there are many simplicial polynomials, then there is an simplicial element in J_l, and thus we can apply Lemma 4.1.

Lemma 4.5. *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 leading variable $x(f_{i_j}) \in T_l$ for $0 \leq j < t$. Then there is a $f_l \in J_l$ generic simplicial.*

Proof. Let's ignore all f_t such that its leading 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's 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 leading 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 it 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 don't 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 the specific case considered in the following lemma. □

Lemma 4.6. *Let $I = \langle f_1, \ldots, f_n \rangle$ such that f_j is generic simplicial for all j. Then there is a $p \in I_{n-1} = J_{n-1}$ generic simplicial.*

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 $x_j \leq x_l$ then $J_j \subseteq I_j \subseteq I_l$. We will explicitly construct such polynomials.

Such 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:

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

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, then we can ignore such variable. Notice that Lemma 4.4 tell 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 3.6 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 that $f = \text{Res}_{x_l}(f_0^l, f_{i+1}^l)$. 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 equation (3.10) we get $f \in \text{elim}_{l+1}(J_l) \subseteq J_p$. Let's see now that $x_j \leq x_p$. The reason is that $x_j \in X_p$, 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, then $f \in \text{elim}_{p+1}(J_p) \subseteq J_r$. 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. □

As mentioned before, we can combine Lemma 4.5 and Lemma 4.1 to obtain guarantees of exact elimination. In the special case when all polynomials are simplicial we

can also prove that elimination is exact, which we now show.

Theorem 4.7. *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. Following the Algorithm 3.1, we obtain $W_l = \emptyset$ and $I_l = \mathrm{elim}_l(I)$ for all l.*

Proof. We will apply Lemma 4.5 and then conclude with Lemma 4.1. 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 l there are at least t_l generators f_j with $x(f_j) \in T_l$ then Lemma 4.5 applies and we are done. Otherwise, let x_l be the largest where such condition fails. The maximality of x_l guarantees that elimination is exact up to such 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 $T_l = \{x_l\} \cup T_{s_1} \cup \ldots \cup T_{s_r}$. 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 bound has to be exact as x_l does not have such 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 $x(f_j) = x_l$, then 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. □

Chapter 5

Elimination ideals of cliques

Algorithm 3.1 allows us to compute (or bound) the elimination ideals $I \cap \mathbb{K}[x_l, \ldots, x_{n-1}]$. In this chapter 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 equation (2.1). 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.

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 chapter 7. We attempt to find such "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 $\cup_l gb_{H_l}$ constitutes such closest sparse representation. In particular, the following proposition says that if there exists a lex Gröbner basis of I that preserves the structure, then $\cup_l gb_{H_l}$ is one also.

Proposition 5.1. *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 $\cup_l gb_{H_l}$ is a lex Gröbner basis of I.

Proof. It is clear that $gb_{H_l} \subseteq H_l \subseteq I$. Let $m \in in(I)$ be some monomial, we just need to show that $m \in in(\cup_l g_{H_l})$. As $in(I) = in(gb)$, we can restrict m to be the leading monomial $m = lm(p)$ 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 $m = lm(p) \in in(H_l) = in(gb_l)$. 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 relation between lex Gröbner bases and clique elimination ideals in section 5.3.

5.1 Elimination ideals of lower sets

We will show now 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 $I_{l+1} = \text{elim}_{l+1}(J_l) + K_{l+1}$. Observe that

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

Continuing this way we conclude:

$$I_l = J_l + J_{l+1} + \dots J_{n-1} \tag{5.1}$$

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 5.2. *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 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 3.5 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 $x_i, x_p, x_r, \dots$ in T that connects $x_i \to x_j$, proving that T is indeed the transitive reduction. □

Definition 5.1. We say a set of variables Λ is a *lower set* if $T|_\Lambda$ is also a tree rooted in x_{n-1}. Equivalently, Λ is a lower set of the poset associated to the DAG of Proposition 5.2.

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

Lemma 5.3. *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 5.4. *Let I be an ideal and assume that Lemma 4.1 holds and thus chordal elimination is exact. Let $\Lambda \subseteq \{x_0, \ldots, x_{n-1}\}$ be a lower set. Then*

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

Proof. 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 $H_\Lambda = J_\Lambda$.

The base case is when $\Lambda = \{x_l, \ldots, x_{n-1}\}$. Note that as x_l is fixed, such Λ is indeed the largest possible lower set. In such case $J_\Lambda = I_l$ as seen in (5.1), and as we are assuming that chordal elimination is exact, then $H_\Lambda = I_l$ as well.

Assume that the result holds for $k+1$ and let's show it for some Λ 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 Λ', obtaining that $H_{\Lambda'} = J_{\Lambda'}$. Note that

$$H_\Lambda = H_{\Lambda'} \cap \mathbb{K}[\Lambda] = J_{\Lambda'} \cap \mathbb{K}[\Lambda] = (J_m + J_\Lambda) \cap \mathbb{K}[\Lambda] \tag{5.2}$$

Observe that the last expression is reminiscent of Lemma 3.2, but in this case we are eliminating x_m. To make it look the same, let's change the term order to $x_m > x_l > x_{l+1} > \cdots > x_{n-1}$. Note that such change has no effect inside X_m, and thus the ideal J_m remains the same. The approximation given in Lemma 3.2 to the elimination ideal in (5.2) is

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

Let x_p be the parent of x_m in T. Then equation (3.10) says that $\text{elim}_{m+1}(J_m) \subseteq J_p$, where we are using that the term order change maintains J_m. Observe that $x_p \in \Lambda$ by the construction of x_m, and then $J_p \subseteq J_\Lambda$. Thus, the approximation to the elimination ideal in (5.2) can be simplified:

$$\text{elim}_{m+1}(J_m) + J_\Lambda = J_\Lambda \tag{5.3}$$

By assumption, Lemma 4.1 holds and thus there is a monomial in $in(J_m)$ that only involves x_m. As the term order change maintains $in(J_m)$, then the lemma still holds and then the expressions in equations (5.2) and (5.3) are the same. Therefore, $H_\Lambda = J_\Lambda$, as we wanted. □

Remark. Note that the equation $I \cap \mathbb{K}[\Lambda] \supseteq \sum_{x_i \in \Lambda} J_i$ holds even if chordal elimination is not exact.

5.2 Cliques elimination algorithm

Lemma 5.4 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 5.5. *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 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 Maximum Cardinality Search (Algorithm 2.1) 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 5.6. *Let I be a zero dimensional ideal with chordal graph G. Assume that we can chordal elimination is exact. Then we can further compute the cliques elimination ideals $H_l = I \cap \mathbb{K}[X_l]$, preserving the structure.*

Proof. We will show an inductive construction of H_l.

The base case is $l = n - 1$ and, as $H_{n-1} = I_{n-1}$, the assertion holds.

Assume that we found H_m for all $x_m < x_l$. Let Λ be a lower set with largest element x_l. By Lemma 5.4, we can compute $I_\Lambda := I \cap \mathbb{K}[\Lambda]$. Note that $X_l \subseteq \Lambda$ because of Lemma 5.3. Thus, $H_l = I_\Lambda \cap \mathbb{K}[X_l]$.

Consider the induced graph $G|_\Lambda$, which is also chordal as G is chordal. Thus, Lemma 5.5 implies that there is a perfect elimination ordering σ of $G|_\Lambda$ where the last clique is X_l. We can now use Algorithm 3.1 in the ideal I_Λ using such ordering of the variables to compute H_l. It just remains to prove that such computation is exact.

Let $X_j^\sigma \subseteq G|_\Lambda$ denote the cliques as defined in (2.1) but using the new ordering σ 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 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's assume that x_m is such that X_m^σ 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 $I \cap \mathbb{K}[X_m^\sigma] = H_r$. Observe that $J_m^\sigma \supseteq H_r$, and H_r is zero dimensional. Then J_m^σ is zero dimensional for all such x_m and then Corollary 4.2 says that chordal elimination is exact. Thus we can compute H_l preserving the chordal structure. □

Observe that the above proof hints to an algorithm to compute H_l. However, the proof depends on the choice of some lower set Λ for each x_l. To avoid eliminations we want to use a lower set Λ as small as possible. By making a good choice we can greatly simplify the procedure and we get, after some observations made in Corollary 5.7, the Algorithm 5.1. 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 5.7. *Let I be a zero dimensional ideal with chordal graph G. Assume that chordal elimination is exact. Then Algorithm 5.1 correctly computes the clique elimination ideals $H_l = I \cap \mathbb{K}[X_l]$, while preserving the structure.*

Algorithm 5.1 Cliques Elimination Ideals

Input: An ideal I with chordal graph G
Output: Cliques elimination ideals $H_l = I \cap \mathbb{K}[X_l]$

```
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}
```

Proof. We refer to the inductive procedure of the proof of Theorem 5.6. 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 5.1 corresponds to selecting the lower set Λ to be this P_l and reusing the eliminations performed to get H_p when we compute H_l.

In the procedure of Theorem 5.6, to get H_l we need a perfect elimination ordering (PEO) σ_l of $G|_\Lambda$ that ends in X_l. This order σ_l determines the eliminations performed in I_Λ. Let σ_p be a PEO of $G|_{P_p}$, whose last vertices are X_p. Let's see that we can extend σ_p to obtain the PEO σ_l of $G|_{P_l}$. Let $C := X_p \cup \{x_l\}$ and observe that $X_l \subseteq C$ due to Lemma 3.5, and thus $P_l = P_p \cup C$. Let σ_C be a PEO of $G|_C$ whose last vertices are X_p (using Lemma5.5). We will argue that the following ordering works:

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

By construction, the last vertices of σ_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 σ_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 σ_C, so that it is a clique because σ_C is a PEO. If

$v \notin C$ then v is not adjacent to x_l as $X_l \subseteq C$, and thus $X_v^{\sigma_l} \subseteq P_l \setminus \{x_l\} = P_p$. Thus, $X_v^{\sigma_l}$ is the same as with σ_p, so that it is a clique because σ_p is a PEO.

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 this clique elimination ideals recursively, as it is done in Algorithm 5.1. □

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. We elaborate on this now.

Lemma 5.8. *Let I be an ideal and let $H_j = I \cap \mathbb{K}[X_j]$ be the cliques elimination ideals. Then*

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

If I is zero dimensional, denoting $I_l = I \cap \mathbb{K}[x_l, \ldots, x_{n-1}]$, then

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

Proof. As $H_j \subseteq I$ for any x_j, then $H_0 + \cdots + H_{n-1} \subseteq I_l$. 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 $I \subseteq H_0 + \cdots + H_{n-1}$.

Let's now prove the decomposition of I_l. For each x_j let gb_{H_j} be a 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 that we just showed 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 $F_l = \bigcup_{x_j \leq x_l} gb_{H_j}$. Note now that as H_j is zero dimensional, then there is a pure power of x_j in gb_{H_j}, and thus Lemma 4.3 says that elimination is exact. Thus

$I_l = \langle F_l \rangle = \sum_{x_j \leq x_l} H_j$, as we wanted. □

Lemma 5.8 gives us a strategy to solve zero dimensional ideals. Note that H_j is also zero dimensional. Thus, we can compute the elimination ideals of the maximal cliques, we solve each H_j independently, and finally we can merge the solutions. We illustrate that now.

Example 5.1. Consider the blue/solid graph in Figure 2-1, and let I be given by:

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

Note that the associated graph $G(I)$ is precisely the blue/solid graph in the figure. However, to use chordal elimination we need to consider the chordal completion of the graph, which includes the three green/dashed edges of the figure. In such 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 5.1 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 them corresponds to the values on the left and the other to the values on the right.

5.3 Lex Gröbner bases and chordal elimination

To finalize this chapter, we will show the relation between lex Gröbner bases of I and lex Gröbner bases of the clique elimination ideals H_l. We will see that both of them 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 5.1, and we will see now that for generic systems this also holds. In other cases, a lex Gröbner bases 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 chapter 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 has a very special type of Gröbner bases. We say that I is in *shape position* if the reduced lex Gröbner bases has the structure:

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

We will prove later the following result for ideals in shape position.

Proposition 5.9. *Let I be a radical zero dimensional ideal in shape position. Let gb_{H_l} be a lex Gröbner basis of H_l. Then the concatenation of all gb_{H_l}'s 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 already 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 of section 7.1, where we will show how much simpler can $\cup_l gb_{H_l}$ 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 Gröbner bases 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.

We use the following result about the structure of elimination Gröbner bases.

Theorem 5.10 ([20]). *Let I be a radical zero dimensional ideal and $V = \mathbf{V}(I)$. 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.

Theorem 5.11. *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 $I_l = I \cap \mathbb{K}[x_l, \ldots, x_{n-1}]$ and*

$H_l = I \cap \mathbb{K}[X_l]$. *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. If $x_l = x_{n-1}$, then $I_l = H_l = I_{n-1}$ and the assertion holds. Otherwise, we want to apply Theorem 5.10, so 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's 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, then Lemma 5.8 implies that $I_l = H_l + I_{l+1}$. 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

$$z \in \pi_{I_l}^{-1}(z_H, z_I) \cap \mathbf{V}(I_l) \iff \rho(z) \in \pi_{H_l}^{-1}(z_H) \cap \mathbf{V}(H_l) \tag{5.4}$$

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 (5.4) holds. This says that $F_{H_l} \subseteq F_{I_l}$, completing the proof. □

Corollary 5.12. *Let I be a radical zero dimensional ideal, then for each x_l we have that $x_l^d \in in(I)$ if and only if $x_l^d \in in(H_l)$, 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 then 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 5.11. □

Proof of Proposition 5.9. As I is in shape position, then its initial ideal has the form

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

for some d. For each $x_l > x_{n-1}$, Corollary 5.12 implies that gb_{H_l} contains some f_l with leading monomial x_l. For x_{n-1}, the corollary says that there is a $f_{n-1} \in gb_{H_{n-1}}$ with leading monomial x_{n-1}^d. Then

$$in(I) = \langle lm(f_0), \ldots, lm(f_{n-2}), lm(f_{n-1}) \rangle$$

and as $f_l \in H_l \subseteq I$, such polynomials form a Gröbner basis of I. □

Chapter 6

Complexity analysis

Solving systems of polynomials in the general case is hard even for small treewidth, as it was shown in Example 1.2. Therefore, we need some additional assumptions to ensure tractable computation. In this chapter 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 of 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.

Let I be a zero dimensional ideal. Let's assume that for each x_i there is a generator with leading monomial of the form $x_i^{d_i}$. Note that Lemma 4.1 holds, and thus chordal elimination is exact.

Definition 6.1. Let $I = \langle f_1, \ldots, f_s \rangle$, we say that I is *q-dominated* if for each x_i there is an f_j with leading monomial $x_i^{d_i}$ with $d_i \leq q$.

It should be mentioned that the conditions above apply for the case of 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) typically we add the equations $x_i^q - x_i$. Even more, by adding such equations we obtain the radical ideal $\mathbf{I}(V_{\mathbb{F}_q}(I))$ [21].

We need to know the complexity of computing a lex Gröbner basis. If the input

of the system is extremely large, the complexity of such task is intrinsically large. To avoid such problem, we assume that the input has been *preprocessed*. Specifically, we make the assumption that the polynomials have been pseudo 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 of the q-dominated condition. This assumption allows us to bound the number of polynomials.

Lemma 6.1. *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 monomial 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 that 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 a single exponential in n [24]. This motivates the following definition.

Definition 6.2. Let α 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 α is stated next. The proof in [21] is for the case of $\mathbb{F}_q$, but the only property that they use is that the ideal is q-dominated.

Proposition 6.2 ([21]). *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. [15] 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 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

$$\hat{H}_j = \langle f : f \text{ generator of } I,\ f \in \mathbb{K}[X_j] \rangle \tag{6.1}$$

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 condition is also satisfied in the case of finite fields. The following lemma shows the reason why we need this assumption.

Lemma 6.3. *Let I be such that for each maximal clique X_j the ideal $\hat{H}_j$ (as in (6.1)) is q-dominated. Then in Algorithm 3.1 we have that J_l is q-dominated for any x_l.*

Proof. Let x_l be arbitrary and let $x_m \in X_l$. We want to find a generator $f \in J_l$ with leading monomial dividing x_m^q. 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 3.6, and thus J_j is q-dominated. Then there must be a generator $f \in J_j$ with leading monomial of the form x_m^d and $d \leq q$.

Let's 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 3.6. As the largest variable of f is x_m, then 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 3.5, 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 3.5 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 that has to eventually terminate. This completes the proof. □

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

We derive now complexity bounds, in terms of field operations, for chordal elimination under the assumptions of Lemma 6.3. We use the following *parameters*: n is the number of variables, s is the number of equations, κ is the clique number (or treewidth), i.e. the size of the largest clique of G.

Theorem 6.4. *Let I be such that each (maximal) $\hat{H}_j$ is q-dominated. In Algorithm 3.1, the complexity of finding 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 κ.*

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 6.3 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 just remains 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 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 $\text{elim}_{l+1}(J_l)$. Then we can do all decompositions in $\widetilde{O}(s + s_0 + s_1 + \ldots + s_{l-1})$. We just need to bound s_l.

It follows from Lemma 6.1 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 $\text{elim}_{l+1}(J_l)$ is bounded by the size of the Gröbner basis of $J_l \subseteq \mathbb{K}[X_l]$, then $s_l = \widetilde{O}(q^\kappa)$. Thus, we can do all decompositions in $\widetilde{O}(s + lq^\kappa)$.

Thus, 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, then Lemma 6.1 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 (3.8) we need to use chordal elimination l times, so the complexity is $O(ls + l^2 q^{\alpha\kappa})$.

Corollary 6.5. *Let I be such that each (maximal) $\hat{H}_j$ is q-dominated. The complexity of Algorithm 5.1 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 κ.*

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 Maximum Cardinality Search runs in linear time, so we can ignore it. The only missing part is to compute the elimination ideas of I_C, where $C = X_p \cup \{x_l\}$. As $|C| \leq \kappa + 1$, then the cost of chordal elimination is $\widetilde{O}(\kappa q^{\alpha\kappa}) = \widetilde{O}(q^{\alpha\kappa})$. Thus the complexity of Algorithm 5.1 is still $\widetilde{O}(nq^{\alpha\kappa})$.

We now prove the second part. As mentioned before, elimination is exact as Lemma 4.1 applies for q-dominated ideals. From Lemma 5.8 and the following remarks we know that the elimination ideals H_l, found with Algorithm 5.1, give a natural description of $\mathbf{V}(I)$. □

The bounds above tell us that for a fixed κ, we can find all clique elimination ideals, and thus describe the variety, in linear time in n. This is reminiscent to many graph problems (e.g. Hamiltonian circuit, vertex colorings, vertex cover) which are NP-hard in general, but are linear for fixed treewidth. In [6] the authors survey such problems. Similar results hold for some types of constraint satisfaction problems [13]. These type 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 now an application of the bounds above for finding graph colorings. It is known that for a fixed treewidth the coloring problem can be solved in linear time [6]. We can prove the same result by encoding colorings into polynomials.

Corollary 6.6. *Let G be a graph and $\bar{G}$ a chordal completion with largest clique κ. 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:

$$x_i^q - 1 = 0, \qquad i \in V \tag{6.2}$$

$$x_i^{q-1} + x_i^{q-2}x_j + \cdots + x_i x_j^{q-2} + x_j^{q-1} = 0, \qquad (i,j) \in E \tag{6.3}$$

where V, E denote the vertices and edges [3, 23]. In such equations each color corresponds to a different square root of unity. Note that the equations in Example 5.1 correspond to 3-colorings of the given graph.

Note that the ideal I_G corresponding to the equations satisfies the q-dominated condition stated before. The chordal graph associated to such ideal is $\bar{G}$. The result follows from Corollary 6.5. □

To conclude, we emphasize the differences between our results to 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 dimensiality, we can use our methods in underconstrained systems and, if they are close to satisfy the q-dominated condition, they should perform well. Indeed, in section 7.3 we test our methods on underconstrained systems.

Chapter 7

Applications

In this chapter we show numerical evaluations of the approach proposed in some concrete applications. Our algorithms were implemented using Sage [31]. Gröbner bases are computed with Singular's interface [14], 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 are performed on an i7 PC with 3.40GHz, 15.6 GB RAM, running Ubuntu 12.04.

We will show the performance of Algorithm 3.1 compared to the Gröbner bases algorithms from Singular and PolyBoRi. In all the applications we give here chordal elimination is exact because of the results of chapter 4. It can be seen below that in all the applications our methods perform better, as the problem gets bigger, than the algorithms from Singular and PolyBoRi.

As mentioned before, chordal elimination has the same limitations as other elimination methods and it performs the best under the conditions studied in chapter 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 6.6. The second case is an application to cryptography, where we solve equations over the finite field $\mathbb{F}_2$.

After that, sections 7.3 and 7.4 show cases were the conditions from chapter 6 are

not satisfied. We use two of the examples from [26], where the authors study a similar chordal approach for semidefinite programming relaxations (SDP). 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 one is given by discretizations of differential equations.

7.1 Graph colorings

We consider the equations (6.2) for q-colorings of a graph, over the field $\mathbb{K} = \mathbb{Q}$. We fix the graph G from Figure 7-1 and vary the number of colors q. Such graph was considered in [23] 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).

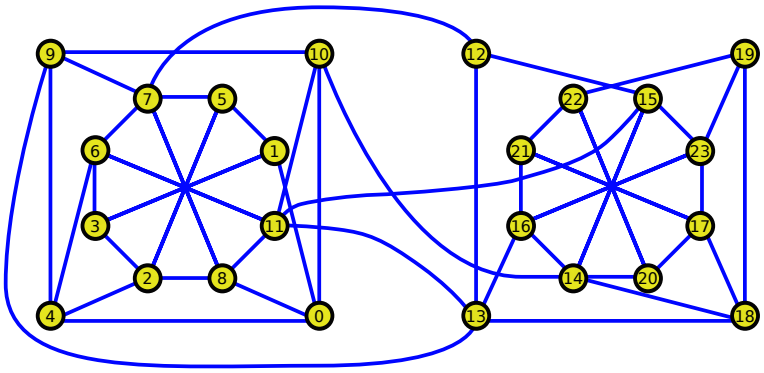

Figure 7-1: Graph with a unique 3-coloring [23].

Table 7.1 shows the performance of Algorithm 3.1 and Algorithm 5.1, 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 run in less than one second. The underlying reason for such long time is the large size of the solution set (number of 4-colorings), which is $|\mathbf{V}(I)| = 572656008$. Therefore, any Gröbner basis

of the ideal will be 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 5.1) are also much simpler.

Table 7.1: Performance (in seconds) on the equations (6.2) (graph of Figure 7-1) for: Algorithm 3.1, Algorithm 5.1, 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 2-1. Table 7.2 shows the results obtained. This time we also show the cost of computing a lex Gröbner bases, 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 find faster a lex Gröbner basis than for degrevlex, by making use of the output from chordal elimination.

Table 7.2: Performance (in seconds) on the equations (6.2) (blue/solid graph of Figure 2-1) for: Algorithm 3.1, Algorithm 5.1, 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 AES variants from [8]. Such cypher can be embedded into a structured system of polynomials equations over $\mathbb{K} = \mathbb{F}_2$ as shown in [8]. Note that as the field is finite the analysis from chapter 6 holds. The parameter n indicates the number of identical blocks used for the encryption. As such, this parameter does not alter the treewidth of the associated graph.

We compare the performance of Algorithm 3.1 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.

Table 7.3: Performance (in seconds) on the equations of $SR(n, 1, 2, 4)$ for: Algorithm 3.1, and computing (lex/degrevlex) Gröbner bases (PolyBoRi). Three different experiments (seeds) are considered for each system. 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

Table 7.3 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.

7.3 Sensor Network Localization

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

$$\|x_i - x_j\|^2 = d_{ij}^2 \qquad (i,j) \in \mathcal{A} \tag{7.1}$$

$$\|x_i - a_k\|^2 = e_{ik}^2 \qquad (i,k) \in \mathcal{B} \tag{7.2}$$

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 4.1) and thus Theorem 4.7 says that chordal elimination is exact. However, the conditions from chapter 6 are not satisfied.

We generate random test problems in a similar way as in [26]. 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 3 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 3.1 and Singular's algorithms. We consider Singular's default Gröbner bases algorithms with both degrevlex and lex orderings, and 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, and poorly in the underconstrained case. Lex Gröbner bases do not compete with chordal elimination in either case.

Table 7.4 summarizes the results obtained. 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 or degrevelex Gröbner basis ever finished within 1000 seconds. On the other hand, chordal elimination succeds in more than half of the instances. For the overconstrained case, lex Gröbner basis algorithm continues to perform poorly. On the other hand, degrevlex Gröbner bases and to FGLM algorithm have slightly better statistics than chordal elimination.

Table 7.4: Statistics of experiments performed on random instances of equations (7.1). 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 ± 5	478.520	1000	1000	-	Mean time (s)
				56%	0%	0%	-	Completed
1/3	100	40	48 ± 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 7-2, where we observe the histogram of the time difference between FGLM and Algorithm 3.1. There we see that in half of the cases (48) both algorithm are within one second and for the rest: in 29 cases FGLM is better, 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.

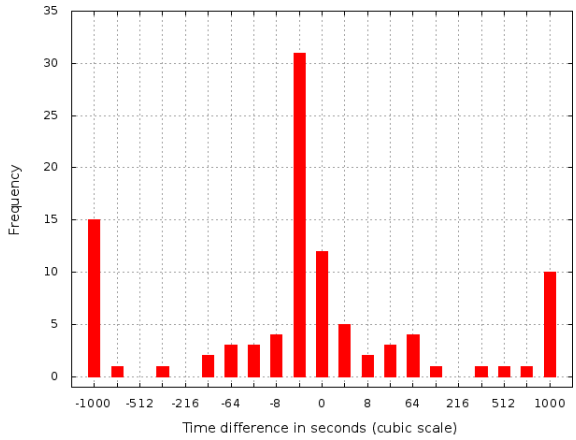

Figure 7-2: Histogram of the time difference between Singular's FGLM and Algorithm 3.1 on 100 random overconstrained ($D = 1/3$) instances of equations (7.1). Positive values correspond to cases where chordal elimination is better.

7.4 Differential Equations

We consider now the following equations over the field $\mathbb{K} = \mathbb{Q}$:

$$0 = 2x_1 - x_2 + \frac{1}{2}h^2(x_1 + t_1)^3 \tag{7.3}$$

$$0 = 2x_i - x_{i-1} - x_{i+1} + \frac{1}{2}h^2(x_i + t_i)^3 \qquad \text{for } i = 2, \ldots, n-1 \tag{7.4}$$

$$0 = 2x_n - x_{n-1} + \frac{1}{2}h^2(x_n + t_n)^3 \tag{7.5}$$

with $h = 1/(n+1)$ and $t_i = i/(n+1)$. Such equations were used in [26], 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 4.1) and thus chordal elimination is exact because of Theorem 4.7. Even more, as x_i is in the initial ideal, the equations J_l obtained in chordal elimination form a lex Gröbner basis. However, the results from chapter 6 do not hold. Nevertheless, we compare the performance of chordal elimination with Singular's default Gröbner basis algorithm with lex order. We also

consider Singular's FGLM implementation.

Table 7.5: Performance (in seconds) on the equations (7.3) for: Algorithm 3.1, 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 7.5 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, FGLM algorithm is not efficient to obtain a lex Gröbner basis.

Bibliography

[1] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[2] Gregory V Bard, Nicolas T Courtois, and Chris Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers. *Cryptology ePrint Archive*, Report 2007/024, 2007.

[3] David Allen Bayer. *The Division Algorithm and the Hilbert Scheme*. PhD thesis, Harvard University, Cambridge, MA, USA, 1982. AAI8222588.

[4] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM (JACM)*, 30(3):479–513, 1983.

[5] Jean RS Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.

[6] Hans L Bodlaender and Arie MCA Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

[7] Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Gröbner-basis computations with boolean polynomials. *Journal of Symbolic Computation*, 44(9):1326 – 1345, 2009. Effective Methods in Algebraic Geometry.

[8] Carlos Cid, Sean Murphy, and Matthew JB Robshaw. Small scale variants of the AES. In *Fast Software Encryption*, pages 145–162. Springer, 2005.

[9] David A Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, 2007.

[10] David A Cox, John B Little, and Donal O'shea. *Using algebraic geometry*, volume 185. Springer, 2005.

[11] Joachim Dahl, Lieven Vandenberghe, and Vwani Roychowdhury. Covariance selection for nonchordal graphs via chordal embedding. *Optimization Methods & Software*, 23(4):501–520, 2008.

[12] Víctor Dalmau, Phokion G Kolaitis, and Moshe Y Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Principles and Practice of Constraint Programming-CP 2002*, pages 310–326. Springer, 2002.

[13] Rina Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[14] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR 3-1-6 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2012.

[15] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. Polynomial systems solving by fast linear algebra. *arXiv preprint arXiv:1304.6039*, 2013.

[16] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[17] Jean-Charles Faugère and Sajjad Rahmany. Solving systems of polynomial equations with symmetries using SAGBI-Gröbner bases. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 151–158. ACM, 2009.

[18] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1, 1): Algorithms and complexity. *Journal of Symbolic Computation*, 46(4):406–437, 2011.

[19] Jean-Charles Faugère, Pierre-Jean Spaenlehauer, and Jules Svartz. Sparse Gröbner bases: the unmixed case. *arXiv preprint arXiv:1402.7205*, 2014.

[20] Shuhong Gao, Virgínia M Rodrigues, and Jeffrey Stroomer. Gröbner basis structure of finite sets of points. *preprint*, 2003.

[21] Sicun Gao. *Counting Zeros over Finite Fields Using Gröbner Bases*. PhD thesis, MS Thesis in Logic and Computation, 2009.

[22] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

[23] Christopher J Hillar and Troels Windfeldt. Algebraic characterization of uniquely vertex colorable graphs. *Journal of Combinatorial Theory, Series B*, 98(2):400–414, 2008.

[24] Yagati N Lakshman. On the complexity of computing a Gröbner basis for the radical of a zero dimensional ideal. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 555–563. ACM, 1990.

[25] Tien-Yien Li. Numerical solution of multivariate polynomial systems by homotopy continuation methods. *Acta numerica*, 6:399–436, 1997.

[26] Jiawang Nie and James Demmel. Sparse SOS relaxations for minimizing functions that are summations of small polynomials. *SIAM Journal on Optimization*, 19(4):1534–1558, 2008.

[27] Vern I Paulsen, Stephen C Power, and Roger R Smith. Schur products and matrix completions. *Journal of functional analysis*, 85(1):151–178, 1989.

[28] Alex Pothen and Sivan Toledo. Elimination structures in scientific computing. *Handbook on Data Structures and Applications, pages*, pages 59–1, 2004.

[29] Håvard Raddum and Igor Semaev. New technique for solving sparse equation systems. *IACR Cryptology ePrint Archive*, 2006:475, 2006.

[30] Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

[31] William Stein et al. Sage: Open source mathematical software, 2008.

[32] Bernd Sturmfels. *Sparse elimination theory*, volume 91. Mathematical Sciences Institute, Cornell University, 1991.

[33] Bernd Sturmfels. *Gröbner bases and convex polytopes*, volume 8. American Mathematical Soc., 1996.

[34] Bernd Sturmfels. *Solving systems of polynomial equations*, volume 97. American Mathematical Soc., 2002.