Gröbner Bases and Graph Colourings

Nina Toon 2658540

James Zoryk 2663347

Anthony Pearson 2659726

Dennis Li 2658411

Mac Shigeki Chong 2666136

Supervisor: Dr. Sander Dahmen

Bachelor Mathematics
Project Computer Assisted Proofs

Department of Mathematics
Faculty of Sciences
Vrije Universiteit
Amsterdam, Netherlands

March 18, 2023

Contents

Introduction

In this report we present some fundamental concepts about Gröbner bases, from which we implement the Bucherger's Algorithm that we use to investigate systems of multi-variable polynomials that arise from finite graphs and colourings, while throughout discussing the use of computers to assist in computation heavy tasks. The use of Gröbner bases has been documented in many applications such as robotics, sudoku, magic squares and graph colourings [CLO15]. Each of these applications involves solving systems of polynomial equations, in which Gröbner bases play a key role. So what are Gröbner bases and why are they useful? Let F be a field. A Gröbner basis is a generating set for an ideal in the polynomial ring $F[x_1, ..., x_n]$ with properties that are desirable in the context of algorithms. A familiar example is that Gröbner bases are a generalization of row reduced echelon forms in Gaussian elimination [Stu05]. Gröbner bases provides us with a way to deal with questions concerning ideal membership and solving polynomial equations. Any finite set of polynomials may be transformed into a Gröbner basis for an ideal generated by this set of polynomials. This property allows for an algorithmic approach to constructing a Gröbner basis for any polynomial ideal.

In practice, computing a Gröbner basis is intensive and is therefore usually left to the computer, which is better suited at preforming these operations as we show in this report. Even though Mathematica has its own built-in function to compute Gröbner bases, we decided to create our own Mathematica code that computes Gröbner bases. This is in order to better understand the problems and limitations while also verifying the methods that we use in our work. Naturally, given the relevance of computers when dealing with Gröbner bases, we will also discuss the role of computer-assistance in our work. Throughout the report, we provide links to github containing generic code of our implementation of the division algorithm and Buchberger's algorithm, as well as specific codes used in examples. The general github repository is found in https://bit.ly/3ALvT6t.

In this report we introduce the idea of Gröbner bases and explore the role of Gröbner bases in finite graph colouring. The basics of Gröbner bases are discussed in Chapter 2. There we introduce monomial orderings and a multivariable division algorithm, followed by the formal definition of a Gröbner basis and a reduced Gröbner basis. Then we discuss Buchberger's algorithm and our implementation of it in Mathematica. The chapter finishes with an example of how Gröbner bases are used to solve polynomial system equations.

In Chapter 3, we will discuss finite graphs and explore one of the earlier mentioned application, namely graph colourings. Intuitively speaking, graphs are a collection of nodes (vertices) that may or may not be connected to each other through lines (edges). In graph colouring, the objective is to assign a colour to each vertex in the graph such that no two vertices connected through an edge share the same colour. The main idea is to translate the object we are working with into a set of polynomials and work with the ideal generated by said polynomials. Then we use the theory of Gröbner basis to find k-colourings, discuss the permutations of k-colourings and prove k-colourability through ideal membership. We have also done some work on magic squares, but we have decided to not include it due to page limit constraints, however the code code can be found at https://bit.ly/3uqV9hi. Finally, in Chapter 4 we compared our implementation of Buchberger's algorithm in Mathematica with its built-in functions, discussed situations when Mathematica helped us and situations when we could solve the problems ourselves, and questions for further research.

Gröbner bases

This Chapter is dedicated to introducing the main definitions, theorems and concepts relevant to understanding Gröbner bases. We denote by $F[x_1, ..., x_n]$ the ring of polynomials in the variables $x_1, \ldots, x_n$ with coefficients in any field F with characteristic 0. The ideas presented in this report can be extended to non-zero characteristic fields, however we shall not consider them.

2.1 Monomial Orderings and Division Algorithm in $F[x_1, \ldots, x_n]$

The notion of ordering terms in polynomials is essential when working with algorithms. We begin this chapter by defining three related but different monomial orderings and introduce a theorem that provides us with a division algorithm in $F[x_1, ..., x_n]$.

Definition 2.1.1. (Monomial) A monomial is a product of powers of variables. In other words, if $x_1, .., x_n$ are variables and $\alpha = (\alpha_1, ..., \alpha_n)$ with each α_i being a nonnegative integer, then $x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$ is a monomial.

Monomial orderings are useful as it allows for systematic approach to algorithms, like long division. Here is an example. If we were to divide a polynomial $3x + x^2 - 1$ by another polynomial $1 + x$, we assume the monomial ordering $x^2 > x > 1$ and rewrite to get $x^2 + 3x - 1$ and $x + 1$ so that the left most terms are the leading terms. Then continue with long division, which also assumes such ordering and it is important that the same ordering is used throughout the process. In the single variable case, a monomial ordering $x^2 > x > 1$ is so natural that we rarely think about it. However, in the multivariate case we have to decide, for example, whether $xy^2z^3 > x^2yz^3$ or $x^2yz^3 > xy^2z^3$ for variables x, y and z.

Although we omit the formal definition of a monomial ordering here, we will introduce three monomial orderings that are relevant for later sections. The formal definition is found in [CLO15, Definition 2.2.1].

Definition 2.1.2. (Lexicographic Order). Let $\alpha = (\alpha_1, \dots, \alpha_n)$ and $\beta = (\beta_1, \dots, \beta_n)$ be elements of $(\mathbb{Z}_{\geq 0})^n$. We shall write $\alpha >_{lex} \beta$ if the leftmost nonzero entry of the vector difference $\alpha - \beta \in \mathbb{Z}^n$ is positive. Given two monomials $x^\alpha, x^\beta \in F[x_1, \dots, x_n]$ we say $x^\alpha >_{lex} x^\beta$ if and only if $\alpha >_{lex} \beta$.

Definition 2.1.3. (Graded Lex order) Let $\alpha, \beta \in (\mathbb{Z}_{\geq 0})^n$. Then we say $\alpha >_{grlex} \beta$ if

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \text{ or } |\alpha| = |\beta| \text{ and } \alpha >_{lex} \beta$$

Definition 2.1.4. (Graded Reverse Lex Order) Let $\alpha, \beta \in (\mathbb{Z}_{\geq 0})^n$. Then we say $\alpha >_{grevlex} \beta$ if

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \text{ or } |\alpha| = |\beta| \text{ and the rightmost nonzero entry of } \alpha - \beta \in \mathbb{Z}^n \text{ is negative.}$$

We use the short notation Lex, Grlex and Grevlex respectively. We note that all of these orderings on $(\mathbb{Z}_{\geq 0})^n$ are well-orderings, where the proof for the lex ordering can be found under [CLO15, Proposition 2.2.4]. The choice of monomial ordering can have a significant effect on how fast an algorithm completes, depending on the problem at hand, which we will experiment with in the chapter about graph colourings.

Example 2.1.1. Here is an example to illustrate the three monomial orderings. Suppose we have the monomials $x_1^4x_2x_3$, $x_1^3x_2^3x_3$ and $x_1^3x_2x_3^3$. The ordering according to the three monomial orderings are as follows.

Lexicographic	Graded Lex	Graded Reverse Lex
$x_1^4x_2x_3 > x_1^3x_2^3x_3 > x_1^3x_2x_3^3$	$x_1^3x_2^3x_3 > x_1^3x_2x_3^3 > x_1^4x_2x_3$	$x_1^3x_2x_3^3 > x_1^3x_2^3x_3 > x_1^4x_2x_3$

We can see that in this example, each monomial ordering gives a different order.

For ease of notation we shall introduce the following definitions.

Definition 2.1.5. Let $f \in F[x_1, \dots, x_n]$ and write $f = \sum_\alpha a_\alpha x^\alpha$. We define $\text{multideg}(f) = \max(\alpha \mid a_\alpha \neq 0)$ where the max is taken with respect to a chosen ordering. Then define

$$\text{LM}(f) = x^{\text{multideg}(f)},\ \text{LC}(f) = a_{\text{multideg}(f)},\ \text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f).$$

The following theorem provides us with a division algorithm, where the proof and the explicit algorithm can be found under [CLO15, Theorem 2.3.3]. See [CLO15, Example 2.3.1] and [CLO15, Example 2.3.2] for some examples on how the algorithm works.

Theorem 2.1.1. (Division Algorithm) Let $F = (f_1, \dots, f_s)$ be an ordered s-tuple of polynomials in $F[x_1, \dots, x_n]$. Then every $f \in F[x_1, \dots, x_n]$ can be written as

$$f = q_1 f_1 + \cdots + q_s f_s + r,$$

where $q_i, r \in F[x_1, \dots, x_n]$. With the condition that either $r = 0$ or r can be written as a linear combination of monomials in $F[x_1, \dots, x_n]$ none of which is divisible by any of $\text{LT}(f_1), \dots, \text{LT}(f_s)$. We call r a remainder of f on division by F, and we shall denote it by $\overline{f}^F$. Furthermore if $q_i f_i \neq 0$ then, $\text{multideg}(f) \geq \text{multideg}(q_i f_i)$.

We stress that the remainder obtained from the division algorithm is not unique. See https://bit.ly/35GYKxf for our implementation of the division algorithm.

2.2 Gröbner Basis and Buchberger's Algorithm

At this point, we have the necessary definitions to define a Gröbner basis.

Definition 2.2.1. (Gröbner Basis) Fix some monomial ordering and let $I \subset F[x_1, \dots, x_n]$ be an ideal. A finite subset $\mathcal{G} = \{g_1, \dots, g_t\}$ of I is called a Gröbner Basis if

$$\langle \mathrm{LT}(g_1), \dots, \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle,$$

where

$$\mathrm{LT}(I) = \{\mathrm{LT}(f) \mid f \in I \backslash \{0\}\}.$$

By convention, the Gröbner basis for the zero ideal is the empty set. Then [CLO15, Corollary 2.5.6] tells us that every ideal $I \subset F[x_1, ..., x_n]$ has a Gröbner basis and any Gröbner basis for the ideal I generates I. We give an example of the use of Gröbner bases in solving a polynomial system of equations in Example 2.2.1. Note that polynomial ideals do not have a unique Gröbner basis, because any set that contains a Gröbner basis is also a Gröbner basis. However, a special kind of Gröbner basis called the reduced Gröbner basis may be interpreted as the "simplest" Gröbner basis of a polynomial ideal and is also unique (with respect to some monomial ordering) to a given polynomial ideal.

Definition 2.2.2. (Reduced Gröbner basis) A reduced Gröbner basis for an ideal $I \subset F[x_1, \dots, x_n]$ is a Gröbner basis $\mathcal{G}$ for I such that:

- $\mathrm{LC}(p) = 1$ for all $p \in \mathcal{G}$.
- For all $p \in \mathcal{G}$, no monomial of p lies in $\langle \mathrm{LT}(\mathcal{G} \backslash \{p\}) \rangle$.

By [CLO15, Theorem 2.7.5], any Gröbner basis has a reduced Gröbner basis, and it is unique (with respect to some fixed monomial ordering). This property is especially useful because it allows us to check if two ideals are equal as two ideals are equal if and only if their reduced Gröbner basis are equal.

We will now introduce Buchberger's criterion, which is used to check whether or not some set is indeed a Gröbner basis for a polynomial ideal.

Theorem 2.2.1. (Buchberger's Criterion). Let $I \subset F[x_1, \dots, x_n]$ be an ideal. Then a basis $G = \{g_1, \dots, g_t\}$ of I is a Gröbner basis of I if and only if for all pairs $i \neq j$ we have $\overline{S(g_i, g_j)}^G = 0$, where

$$S(f, g) := \frac{\mathrm{lcm}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(f)} \cdot f - \frac{\mathrm{lcm}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(g)} \cdot g.$$

The proof of this can be found under [CLO15, Theorem 2.6.6].

Perhaps unsurprisingly, Buchberger's criterion is a fundamental building block for Buchberger's Algorithm for computing Gröbner bases from a generating set.

2.2.1 Buchberger's Algorithm

Buchberger's Algorithm is an algorithm used to find a Gröbner basis of an ideal generated by finitely many polynomials. The algorithm itself and a proof of its correctness can be found in [CLO15, Theorem 2.7.2]. We shall see in Section 3.2.1 an example on how Buchberger's algorithm is used to compute Gröbner bases.

Throughout Chapter 3, we will use both our own implementation of the improved Buchberger's algorithm as well as Mathematica's built-in function `GroebnerBasis[ ]` for computing Gröbner bases. Unless specified otherwise we will initially use our own implementation of the division algorithm and Buchberger's algorithm in Mathematica. At first we used the basic Buchberger's algorithm, but after testing it on some random graphs, we noticed that it was extremely slow in producing a Gröbner basis for graphs of more than four vertices. On the other hand, we found that [CLO15, Theorem 2.10.9] provides an improved Buchberger's algorithm. So instead, we implemented the improved Buchberger's algorithm, which can be found in https://bit.ly/3AYUFjO with proof at [CLO15, Theorem 2.10.9].

Throughout the text we will also give small examples of what one can input into Mathematica to achieve our results. These examples will often have sets and polynomials that were referred to in the text with no reference to how we constructed them in Mathematica. In these cases we refer to https://bit.ly/3AKSUq0 to see how we constructed the sets used. Additionally, we note that when recording the runtime of a program, within each subsection, the same computer is used.

2.2.2 Varieties

Definition 2.2.3. Let $f_1, \dots, f_s \in F[x_1, \dots, x_n]$, then we define,

$$V(f_1, \dots, f_s) := \{(a_1, \dots, a_n) \in F^n \mid f_i(a_1, \dots, a_n) = 0 \text{ for all } 1 \leq i \leq s\}.$$

The set $V(f_1, \dots, f_s)$ is called the variety defined by $f_1, ..., f_s$.

The following theorem relates varieties in some polynomial ideal to its Gröbner basis. The proof can be found in [CLO15, Theorem 1.4.4].

Theorem 2.2.2. Suppose $\langle f_1, \dots, f_s \rangle = \langle g_1, \dots, g_t \rangle \subset F[x_1, \dots, x_n]$ for some polynomials $f_i, g_j \in F[x_1, \dots, x_n]$. Then $V(f_1, \dots, f_s) = V(g_1, \dots, g_t)$.

Theorem 2.2.2 tells us that finding the common roots of a set of polynomials amounts to finding common roots of its corresponding (reduced) Gröbner basis.

Example 2.2.1. Consider the following set of polynomials in $\mathbb{R}[x, y]$: $P = \{x^2 - y^2, x^3 - y^2 - 1\}$. One method to find the common roots of the polynomials in P is to do some substitutions and then solve for the variables. Alternatively, we find a reduced Gröbner basis for the ideal $\langle P \rangle$ first. A Gröbner computation gives the Gröbner basis $\mathcal{G} = \{x + y^2 - y^4 + 1, y^6 - y^4 - 2y^2 - 1\}$. Notice that the Gröbner basis contains two polynomials as in the original set P, however one of the polynomials is dependent only on the variable y. In essence this simplifies the system so it becomes easier to solve than the system $P = 0$. By Theorem 2.2.2 it holds that $V(\mathcal{G}) = V(P)$, meaning we may solve for the common roots of $\mathcal{G}$ to find the common roots of P. One may see that $P = 0$ could be solved by hand, however if the number of expressions and variables is increased this becomes tedious work thus the reduced Gröbner basis becomes extremely useful. See the Mathematica notebook found in https://tinyurl.com/3b9v334d for graphs and the solution set.

Graph Colourings

3.1 Definition of graphs and graph colouring

We shall henceforth assume that the field that we are working in is $\mathbb{C}$. A graph is a pair of sets $G = (V, E)$, where V is the set containing vertices and E the set containing edges. An (undirected) edge is an unordered pair $\{x, y\}$ that we think of as connecting some vertices $x, y \in V$. Therefore E is some subset

$$E \subset \{\{x, y\} | x, y \in V, \ x \neq y\}.$$

The order of a graph is the number of vertices $|V|$. Two vertices x and y are adjacent to each other if $\{x, y\} \in E$. A graph is called complete if every vertex is adjacent to every other vertex in the graph, and will be denoted by K_n where n is the order of the graph. Note, in this report we shall only consider finite graphs.

In graph colouring, the objective is to assign a colour to each vertex in V such that no two adjacent vertices share the same colour. A k-colouring is an assigning of colours to every vertex of a graph G using at most k amount of colours, such that no two adjacent vertices share a colour. It is not necessary to use all k colours to be considered a k-colouring. If there exists a k-colouring for a graph G, then G is k-colourable. The minimum amount of colours needed to colour a graph G is called the chromatic number, and is denoted by $\chi(G)$

We will translate graph colourings into a problem involving polynomial rings and Gröbner bases. Aside from finding k-colourings, we consider several questions about graph colourings, which we attempt to answer using Gröbner basis theory:

- What is the role of Gröbner bases in finding k-colourings?
- How much can Mathematica handle in terms of graph size?
- How much does the number of colours affect computational speed?
- How many (essentially) different ways are there to colour a graph?

3.2 Polynomial representation of graph colourings

To translate graph colourings into a problem involving polynomial rings and Gröbner bases, we need to give the vertices and edges of a given graph some polynomial representation. More specifically, we want a set of polynomials G_k such that an element of the variety $V(G_k)$ corresponds to a k-colouring, $V(G_k)$ contains all possible k-colourings, and there is no k-colouring if and only if $V(G_k)$ is empty. We use the following polynomial representation.

Given a graph $G = (V, E)$ of order n, we think of each vertex $v_i \in V$ as a variable for $1 \leq i \leq n$, where its value will determine the colour of the vertex. Suppose we want to colour G with k-colours. For each of the k colours we associate to it a unique kth root of unity. Then, for a vertex v_i we consider the polynomial $V_i^k(v_i) = v_i^k - 1$ where the choice of roots determines the colour of v_i. Now we need a way of encoding the fact that if $\{v_i, v_j\} \in E$ then $v_i \neq v_j$. This is done with the following polynomial

$$E_{i,j}^k(v_i, v_j) = \sum_{m=1}^{k} v_i^{k-m} v_j^{m-1} \left(= \frac{v_i^k - v_j^k}{v_i - v_j} \right) \tag{3.1}$$

where the second equality makes sense algebraically in the ring $\mathbb{C}(v_i, \ldots, v_n)$, but not if we want to compute with it when $v_i = v_j$. We shall prove that this polynomial encodes such information.

Lemma 3.2.1. Let $G = (V, E)$ be a graph. Two vertices $v_i, v_j \in V$ do not share the same colour if and only if the polynomial $E_{i,j}^k(v_i, v_j) = 0$.

Proof. Suppose first that v_i and v_j do not share the same colour, then by definition we have $v_i \neq v_j$ so we can make sense of the right most expression in (3.1). Since v_i and v_j are kth roots of unity we have both $v_i^k = 1$, $v_j^k = 1$ and consequently $v_i^k - v_j^k = 0$ so that $E_{i,j}^k(v_i, v_j) = 0$.

Conversely we wish to show that if v_i and v_j are coloured the same then $E_{i,j}(v_i, v_j) \neq 0$. In this case we get $v_i = v_j$ so,

$$E_{i,j}^k(v_i, v_j) = \sum_{m=1}^{k} v_i^{k-m} v_j^{m-1} = \sum_{m=1}^{k} v_i^{k-1} = k v_i^{k-1} \neq 0.$$

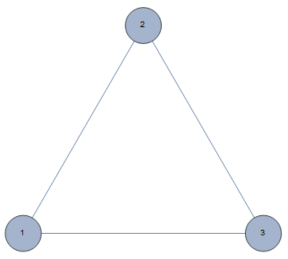
$\square$

Suppose we wanted to colour a graph $G = (V, E)$ with k colours. The earlier discussions suggest the related sets of polynomials are:

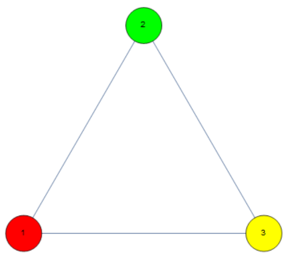
$$\mathcal{V}_G^k = \{V_i^k \mid v_i \in V\}, \ \mathcal{E}_G^k = \{E_{i,j}^k \mid \{v_i, v_j\} \in E\}.$$

We shall define $G_k := \mathcal{V}_G^k \cup \mathcal{E}_G^k$ which is completely determined by each graph G and the k colours. The variety $V(G_k)$ is empty if and only if G is not k-colourable. Then to find a k-colouring of G (if it exists), it suffices to find any element of $V(G_k)$ and associate each root with a colour (if $V(G_k)$ is non-empty).

3.2.1 Example graph colouring



(a) Graph of K_3.



(b) Colouring of K_3.

Figure 3.1: Graph K_3 and a 3-colouring

Before we move onto larger graphs, we will focus our attention on K_3 (see Figure 3.1a) for a simple example to see how Gröbner bases are used in practice. It is immediately clear that $\chi(K_3) = 3$, so by previous discussions we wish to find the variety $V((K_3)_3)$. To do this we shall compute a Gröbner basis of the ideal generated by

$$(K_3)_3 = \{x_1^3 - 1, \ x_2^3 - 1, \ x_3^3 - 1, \ x_1^2 + x_1x_2 + x_2^2, \ x_1^2 + x_1x_3 + x_3^2, \ x_2^2 + x_2x_3 + x_3^2\}$$

where we will denote the ith element of $(K_3)_3$ by f_i. Using Theorem 2.2.1 we know that $(K_3)_3$ is a Gröbner basis if and only if all pairs $S(f_i, f_j)$ have remainder zero under division by $(K_3)_3$. Furthermore since $\overline{S(f_i, f_j)}^{(K_3)_3} \in \langle (K_3)_3 \rangle$ as $S(f_i, f_j) \in \langle (K_3)_3 \rangle$, simply appending $\overline{S(f_i, f_j)}^{(K_3)_3}$ to $(K_3)_3$ does not change the ideal that it generates. Then of course if $\overline{S(f_i, f_j)}^{(K_3)_3}$ is already in $(K_3)_3$ then division by this set will give a remainder of zero, so we are one step closer to a Gröbner basis. We remark that this is exactly what Buchberger's algorithm does, by continually appending $\overline{S(f_i, f_j)}^{(K_3)_3}$ one can, in theory, generate a Gröbner basis of any ideal, albeit very slowly. Indeed this is what we shall do to generate a Gröbner basis for our example with K_3, while assuming the lexicographic ordering.

$$\begin{aligned} S(f_1, f_5) &= \frac{x_1^3}{x_1^3}(x_1^3 - 1) - \frac{x_1^3}{x_1^2}(x_1^2 + x_1x_3 + x_3^2) = -x_1^2x_3 - x_1x_3^2 - 1, \\ &= (-1)f_3 + (-x_3)f_4 + (x_3)f_6 + (x_1x_2x_3 - x_1x_3^2 - x_2x_3^2 - 2). \end{aligned}$$

Then from this one can see that $R_1 := \overline{S(f_1, f_5)}^{(K_3)_3} = x_1x_2x_3 - x_1x_3^2 - x_2x_3^2 - 2 \neq 0$ so we redefine $(K_3)_3$ by appending R_1 to it, from here the rest of the computation is identical so we only show the end result. We have $R_2 := \overline{S(f_2, f_5)}^{(K_3)_3} = x_1x_2 - x_1x_3 - x_2x_3 - 2x_3^2 \neq 0$, $R_3 := \overline{S(f_2, R_1)}^{(K_3)_3} = -3x_1x_3 - 3x_2x_3 - 3x_3^2 \neq 0$, and $R_4 := \overline{S(f_5, R_1)}^{(K_3)_3} = 3x_1 + 3x_2 + 3x_3 \neq 0$ which we append all to $(K_3)_3$. We will not do it here. However one can check by hand or running it through a computer that our new $(K_3)_3$ satisfies the condition in Theorem 2.2.1 to be a Gröbner basis. To avoid confusion we shall denote this new set by $\mathcal{G}$ below

$$\begin{aligned} \mathcal{G} = \{&x_1^3 - 1,\ x_2^3 - 1,\ x_3^3 - 1,\ x_1^2 + x_1x_2 + x_2^2,\ x_1^2 + x_1x_3 + x_3^2,\ x_2^2 + x_2x_3 + x_3^2, \\ &x_1x_2x_3 - x_1x_3^2 - x_2x_3^2 - 2,\ x_1x_2 - x_1x_3 - x_2x_3 - 2x_3^2,\ -3x_1x_3 - 3x_2x_3 - 3x_3^2,\ 3x_1 + 3x_2 + 3x_3\}. \end{aligned}$$

One can begin to solve for zeroes from here, however it is much easier to consider the reduced Gröbner basis as it does not take much more work to get to it. We can first start by removing the leading coefficients, then considering the second condition of a reduced Gröbner basis we look at

$$\mathrm{LT}(\mathcal{G}) = \{x_1^3,\ x_2^3,\ x_3^3,\ x_1^2,\ x_2^2,\ x_1x_2x_3,\ x_1x_2,\ x_1x_3,\ x_1\}.$$

Since we only need $\mathrm{LT}(\mathcal{G})$ to generate $\langle \mathrm{LT}((K_3)_3)\rangle$, we can remove any elements that can already be generated from other elements, with which we achieve

$$\mathrm{LT}(\mathcal{G}) = \{x_3^3,\ x_2^2,\ x_1\}.$$

Then it turns out that

$$\mathcal{G} = \{x_3^3 - 1,\ x_2^2 + x_2x_3 + x_3^2,\ x_1 + x_2 + x_3\}$$

is a reduced Gröbner basis for $\langle (K_3)_3\rangle$. We remark that this method of eliminating terms from a Gröbner basis does not necessarily produce a reduced Gröbner basis, which can be seen in https://bit.ly/3IY8wtr under the self-made function `MonomialReduce`, so one has to do more work in general. In any case, the first condition is clearly satisfied and one can easily check that the other condition is also satisfied. Then comparing solving for common zeroes in $(K_3)_3$ versus $\mathcal{G}$, we immediately see that using $\mathcal{G}$ is substantially easier. This combined with Theorem 2.2.2 should be convincing evidence for why one should consider computing a Gröbner basis if one wants to solve a polynomial system. Indeed we can see that x_3 can be any of the 3rd roots of unity so let us take $x_3 = 1$. Then we have that x_2 must satisfy $x_2^2 + x_2 + 1 = 0$ which has roots $\exp(2\pi i/3)$ or $\exp(-2\pi i/3)$ so let us choose $\exp(2\pi i/3)$. Finally x_1 must satisfy $x_1 + 1 + \exp(2\pi i/3) = 0$ and it is easily seen that $x_1 = \exp(-2\pi i/3)$ is the solution, the last 3rd root of unity. Then assigning a colour to each 3rd root of unity we achieve a colouring of K_3, see Figure 3.1b.

3.3 Finding k-colourings of large graphs

Throughout this section we use the built-in Gröbner basis function because our implementation of the (improved) Buchberger's algorithm is not fast enough to get result in a reasonable time. For a given graph G, the time it takes the in-built function to compute a Gröbner basis for $\langle G_k\rangle$ is extremely sporadic. For instance with 30 vertices and 50 edges, it would take over an hour for one permutation of the edges, then for another it would finish in 4 seconds (see https://bit.ly/3J0Onmj). However we have not experienced this behavior for any other number of vertices and edges, indeed for the most part anything above 50 vertices took longer than 10 minutes, but that is not to say there does not exist some permutation of 100 vertices and 200 edges for which a colouring can be computed in under a second. On the other hand, it did seem that increasing the amount of colours always increased the amount of time it takes for the computation, which reflects the figure in 3.2. Therefore it is extremely difficult to give a definite answer to the limitations of Mathematica's in-built Gröbner basis function as it seems to be extremely case dependant.

Taking into account the last disscusion, if we would like to have a graph that is as large as possible we should look for ones that are 3-colourable; then it would seem appropriate to look at graph with a large amount of vertices but a small amount of corresponding edges. We have settled upon a randomized graph with $|V| = 40$ and $|E| = 70$ using the code `While@Not@ConnectedGraphQ[d = RandomGraph[{40, 70}]`, the outputs of the code can be seen in https://bit.ly/3sdEif2. Mathematica can find a reduced Gröbner basis, B, for $\langle G_3\rangle$ in just under

30 seconds. Since we only want to find one 3-colouring of G we can plug `FindInstance[B,Variables[B]]` into Mathematica to find a colouring of G, and indeed it finds such a colouring in just over 18 minutes.

In general, for a given graph G, one can find a k-colouring of G (provided it exists) by computing a reduced Gröbner basis of $\langle G_k \rangle$ and solving for the common zeroes in this Gröbner basis. Of course the time in which such a method can be completed can take an extremely long time, but in principle this always works.

3.4 Comparing speed for different amounts of colours

Based on the polynomials for graph colouring discussed in Section 3.2, it seems reasonable to think that increasing the number of colours will complicate its Gröbner basis computation. We would like to get a feel for this, which we shall do by comparing runtimes. Furthermore, the built-in function `GroebnerBasis[ ]`in Mathematica has the option to use either "Buchberger" or "GroebnerWalk" method to compute the Gröbner basis. The monomial order of the function has then three options, namely Lexicographic, Graded Lex and Graded Reverse Lex. Therefore we chose the "Buchberger" method and constructed a simple program in Mathematica (see https://tinyurl.com/2p85fnz6) using increasing colourings k, while recording the runtimes of each of these options to get the results. Though the number of colourings is easily computable by hand, we use the complete graph K_3 for demonstration. The results are shown in Figure 3.2.

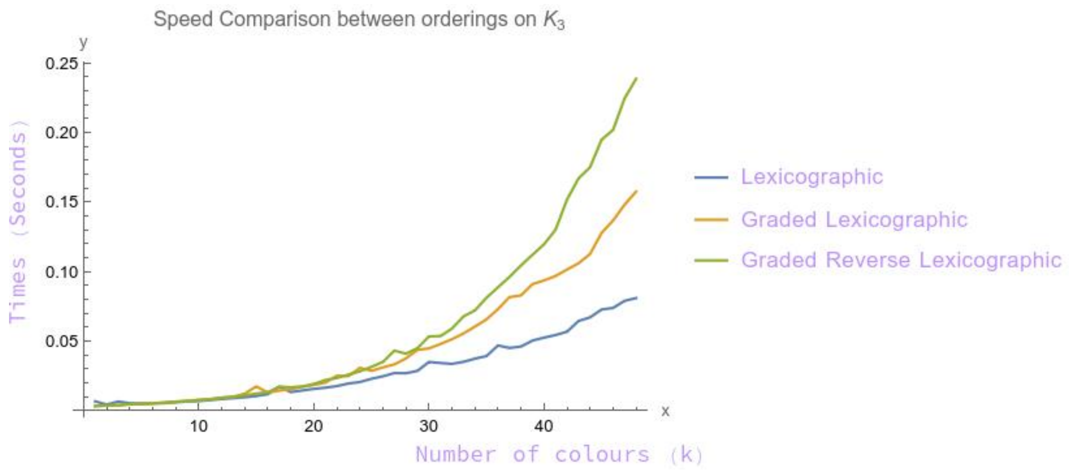


Figure 3.2: Runtime of GroebnerBasis function on K_3 with increasing k, measured in seconds.

If we observe the results in Figure 3.2, increasing the number of colours k can lead to an significantly longer runtime of the program. It is also clear from the figure that the program finishes fastest with lexicographic order, while graded rev lex is slowest for larger k. Although not shown here, as k becomes larger the polynomials become increasingly more complicated. Using the smallest possible number of colours is therefore essential, especially when using large graphs with many edges.

Furthermore, we also attempted to demonstrates the effects of increasing the number of vertices and edges has on the computation speed of the of the built-in function, while also illustrating the increase of complexity. However, due to computer system resources constraints this was not fully achieved. The work can be found here https://tinyurl.com/35c56f52.

3.5 Permutations of colourings

The subject of this section will be determining the number of colourings of a given graph. As an example, if we look back to the previous section and the Gröbner basis $\mathcal{G}$ for K_3, it is clear that a colouring is completely determined by the choice of zero for x_3 and x_2. We have three choices for x_3, namely the 3rd roots of unity, then x_2 satisfies a quadratic polynomial with non-zero determinant so there are two choices for that, and finally x_1 is predetermined by the choice of x_3 and x_2. Therefore we have $3 \cdot 2 = 6$ choices for a 3-colouring of K_3. Furthermore if we have n colours then it is clear that the number of possible n-colourings is $\binom{n}{3} \cdot 6 = n(n-1)(n-2)$. For a more elaborate example we consider the graph shown in Figure 3.3.

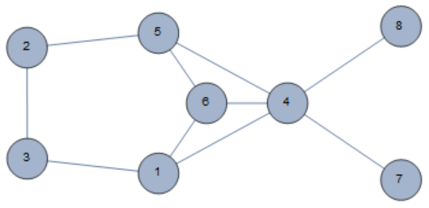


Figure 3.3: Graph G with $|V| = 8$ and $|E| = 10$.

For the remainder of this section we shall assume the lexicographic ordering. Since we see that G contains a K_3 subgraph we shall attempt to make a 3-colouring. Then one can compute the reduced Gröbner basis of $\langle G_3 \rangle$, which we call $\mathcal{G}$, in very much the similar way to the last section, albeit with a lot more steps though it can be automated with computer code:

$$\begin{aligned}\mathcal{G} = \{&x_8^3 - 1,\ x_7^3 - 1,\ x_7x_6^2 - x_8x_6^2 - x_7^2x_6 + x_8^2x_6 - x_7x_8^2 + x_7^2x_8,\ x_6^3 - 1,\\ &- x_7^2 + x_5x_7 + x_6x_7 + x_8^2 - x_5x_8 - x_6x_8,\ x_8^2 - x_5x_8 - x_6x_8 + x_5x_6,\ x_5^2 + x_5x_8 + x_6^2 - x_8^2 + x_6x_8,\\ &x_4 + x_5 + x_6,\ x_3^2 + x_5x_3 - x_6^2 + x_8^2 - x_5x_8 - x_6x_8,\ x_2 + x_3 + x_5,\ x_1 - x_5\}.\end{aligned}$$

We shall introduce a theorem that may help in finding $|V(\mathcal{G})|$. The proof of this can be found under [CLO15, Proposition 9.3.7].

Theorem 3.5.1. Let $I \subset F[x_1, \ldots, x_n]$ be an ideal such that for each i, some power $x_i^{m_i} \in \langle \mathrm{LT}(I) \rangle$. Then $|V(I)| \le m_1 \cdots m_n$.

Then by the definition of a Gröbner basis we have,

$$\langle \mathrm{LT}(I) \rangle = \langle x_8^3,\ x_7^3,\ x_7x_6^2,\ x_6^3,\ x_5x_7,\ x_5x_6,\ x_5^2,\ x_4,\ x_3^2,\ x_2,\ x_1 \rangle$$

Therefore $x_8^3, x_7^3, x_6^3, x_5^2, x_4, x_3^2, x_2, x_1 \in \langle \mathrm{LT}(I) \rangle$ so an upper bound of the size of $V(I)$ is $3 \cdot 3 \cdot 3 \cdot 2 \cdot 2 = 108$. We note that this is a very crude approximation because this is saying that there are three choices for x_8, x_7 and x_6 and two choices for x_5 and x_4, but not all of these choices necessarily result in a valid colouring. Although tedious, we can go through a similar process as to what we did in the K_3 case to find the exact number of colourings.

We see that x_6 satisfies,

$$(x_7 - x_8)x_6^2 - (x_7 + x_8)(x_7 - x_8)x_6 + x_7x_8(x_7 - x_8) = 0.$$

Therefore we have two cases to consider: $x_7 = x_8$ or $x_6^2 - (x_7 + x_8)x_6 + x_7x_8 = 0$. Then one can check that the second case is equivalent to $x_6 = x_7$ or $x_6 = x_8$ so we have three different cases to consider.

Case 1: We first consider the case $x_7 = x_8$. Then from the fifth entry we have $x_5(x_6 - x_7) - x_7(x_6 - x_7)$ so $x_5 = x_7$ or $x_6 = x_7$.

Subcase 1.1 We will consider $x_5 = x_7$, and to make things easier we can compute a Gröbner basis for this specific case.

$$\mathcal{G}_{1.1} = \left\{x_6^3 - 1,\ x_5^2 + x_6x_5 + x_6^2,\ x_4 + x_5 + x_6,\ x_3^2 + x_5x_3 - x_6^2 - x_5x_6,\ x_2 + x_3 + x_5,\ x_1 - x_5\right\}.$$

Then here we are very much in the same case as K_3 since we have three options for x_6, two for x_5 and two for x_3 and the rest are determined by those choices, so we have $3 \cdot 2 \cdot 2 = 12$ possible colourings of G.

Subcase 1.2 Now we consider $x_6 = x_7$ and once again compute a Gröbner basis. It turns out that it is equal to $\mathcal{G}_{1.1}$ so we once again have 12 colourings. Furthermore they are different from the first subcase because $x_6 \neq x_5$, so in total we have 24 colourings.

Case 2: We shall consider $x_6 = x_7$. Then from the fifth entry we have $x_5(x_6 - x_8) - x_8(x_6 - x_8)$ so either $x_5 = x_8$ or $x_6 = x_8$. If $x_6 = x_8$ then we have $x_8 = x_7 = x_6$ which was already considered in the first case. Therefore we look at $x_5 = x_8$, and as before it turns out that the Gröbner basis for this specific case is once again $\mathcal{G}_{1.1}$ so we get 12 more colourings, which are necessarily different from the first case otherwise $x_5 = x_6$.

Case 3: Finally we consider $x_5 = x_7$, then once again the fifth entry gives $x_5(x_6 - x_8) - x_8(x_6 - x_8)$ so either $x_5 = x_8$ or $x_6 = x_8$. The first option is the same as the first case, and as one might guess if we consider the second case, the Gröbner basis reduces to $\mathcal{G}_{1.1}$ so there are 12 more colourings which are different as $x_5 = x_6$ is impossible.

Hence in total we have $24 + 12 + 12 = 48$ total colourings of G. This is easily verifiable with brute force because there are only finitely many possibilities that could work, namely 3^8. Then running through all the possiblities is computationally easy, in fact Mathematica also finds that $|V(\mathcal{G})| = 48$ (the computation by hand was mainly done to verify this) with `Length[Solve[G==0]]` in under two seconds, and has the additional merit of giving all the exact solutions. On another note, if the graphs become large then some easy optimizations include specifying the domain to only include kth roots of unity instead of the entire complex plane, or doing only some of the casework above and leaving the rest to the computer.

3.6 Proving k-colourability through ideal membership

It may be useful to determine the chromatic number of a graph before finding a k-colouring. A method to determine if $\chi(G) = k$ is to prove that G is k-colourable, but not $(k-1)$-colourable. Recall that the graph G is not k-colourable if and only if $V(G_k)$ is empty. Therefore proving k-colourability amounts to finding a reduced Gröbner basis and check that it is not equal to $\{1\}$. Although a natural course of action, it requires the computation of (reduced) Gröbner bases for the ideal $\langle G_k \rangle$, which may not be fast if $|G_k|$ and k is large. Let us consider the problem of proving k-colourability through ideal membership. So we opt for an alternative approach involving ideal membership and see if it is a faster and practical alternative. Let us first define the graph polynomial.

Definition 3.6.1. (Graph Polynomial) Let $G = (V, E)$ be a graph. Then the polynomial

$$P_G := \prod_{\{v_i, v_j\} \in E} (v_i - v_j) \in \mathbb{C}[V]$$

is the graph polynomial of G, where $\mathbb{C}[V]$ denotes the set of polynomials with variables in V and coefficients in the field $\mathbb{C}$.

As a side note, the graph polynomial P_G provides us with a way to quickly determine if some assigning of colours to vertices is a k-colouring. Given a k-colouring for a graph G, $P_G \neq 0$ if and only if no adjacent vertices have the same colour. Using the graph polynomial, the following theorem may be useful in proving the (non-)existence of a k-colouring, for which the proof can be found under [Meh, Theorem 1].

Theorem 3.6.1. Let $G = (V, E)$ be a graph of order n and $P_G = \prod_{\{v_i, v_j\} \in E} (v_i - v_j) \in \mathbb{C}[V]$. Let k be a positive integer and let $I = \langle v_1^k - 1, \ldots, v_n^k - 1 \rangle$ with $v_i \in V$. Then $P_G \notin I$ if and only if G is k-colourable.

Theorem 3.6.1 allows us to turn a graph colouring problem into a problem of ideal membership. The following theorem turns out to be very useful to proof that $P_G \in I$ or $P_G \notin I$. The theorem is proven under [CLO15, Corollary 2.6.2].

Theorem 3.6.2. Let $G = \{g_1, ..., g_t\}$ be a Gröbner basis for an ideal $I \subset F[x_1, ..., x_n]$ and let $f \in F[x_1, ..., x_n]$. Then $f \in I$ if and only if the remainder on division of f by G is zero.

3.6.1 Example

We will apply both theorem first on a small graph so that it can be done by hand, to gain a better understanding. Suppose we have a graph G with $V = \{v_1, v_2, v_3\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}\}$ (see Figure 3.4).

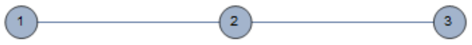


Figure 3.4: Graph used as small example.

Let $P_G = (v_1 - v_2)(v_2 - v_3) = v_1v_2 - v_1v_3 - v_2^2 + v_2v_3$. Let us first prove that G is not 1-colourable. Let $I = \langle v_1 - 1, v_2 - 1, v_3 - 1 \rangle$. Note that the generators of I form a Gröbner basis of I. Using the division algorithm, we divide P_G by the generators of I and writing it as a linear combination, resulting in

$$P_G = (v_1 - 1)(v_2 - v_3) + (v_2 - 1)(v_3 - v_2) \in I$$

Therefore by Theorem 3.6.1, G is not 1-colourable. Now let us prove that G is 2-colourable. Then P_G stays the same, but we let $I = \langle v_1^2 - 1, v_2^2 - 1, v_3^2 - 1 \rangle$. The generators of I are already a Gröbner basis for I.

The only term in P_G that is quadratic is $-v_2^2$, meaning we can write

$$P_G = -(v_2^2 - 1) - 1 + v_1v_2 - v_1v_3 + v_2v_3$$

However, none of the terms are quadratic in any other variables. Therefore dividing P_G with the generators of I gives a non-zero remainder, so by Theorem 3.6.2 $P_G \notin I$ and the graph G is 2-colourable. This example gives us an idea that the higher k is, the easier it is to proof that a graph is k-colourable using Theorem 2.2.1 because P_G is independent of k, while the degrees of the generators of I get higher as k goes up. So division actually gets easier the higher k becomes.

Unfortunately Theorem 3.6.1 does not seem to be something that is feasible in practice for large graphs as we attempted to use it to prove that the graph used in Section 3.3 is 3-colourable. However using `PolynomialReduce[]`, Mathematica crashed before given any results. On the other hand, computing the Gröbner basis and finding the common zeroes only took at most 20 minutes. We predict that a similar issue would occur for larger graphs given that adding vertices and edges only make the objects in Theorem 3.6.1 more complicated. Instead, we consider a medium sized graph. Consider the following graph in as shown in Figure 3.5.

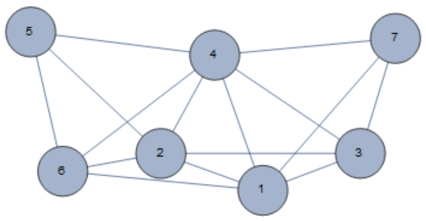


Figure 3.5: Graph with $|V| = 7$ and $|E| = 15$.

We shall first use our code as a test (see https://bit.ly/3gdL8f6). We follow the same process of constructing the polynomial P_G and the ideal I as in the small example. Using our implementation of the division algorithm, the function returned a nonzero remainder for $k = 4$ with graded reverse lex order in 30 minutes. With the lex ordering, the program did not return a remainder within an hour. For $k = 3$, the function did not complete within two hours with both graded reverse lex ordering and lex ordering. Let us compare our implementation with Mathematica's built-in `PolynomialReduce[ ]` function. If we denote by K the generating set for I we can use the Mathematica's built-in function `AbsoluteTiming[PolynomialReduce[P_G,K,Variables[P_G]]]`. We find that for $k = 3$ the program gives a zero remainder in 0.13879 seconds. For $k = 4$ the function returns a nonzero remainder in 0.118682 seconds. So the chromatic number of the graph is 4. For a larger $k = 7$, a nonzero remainder is returned in 0.0756705 seconds. Notice that it is faster for $k = 7$ than $k = 3$. This is likely because in the case of $k = 7$, there is almost no division to be done, unlike when $k = 3$.

Discussion

Finding the Gröbner basis is entirely done with symbolic arithmetic we have no reason to believe that the results are incorrect, unless the function itself is wrong but given the popularity of Mathematica this is unlikely. On the other hand it is not entirely clear how `Solve[ ]` works, however regardless of how accurate the `Solve[ ]` function is, it is easy to verify if the given solutions are valid by checking if they are indeed zeroes of our system. In contrast, although our implementation of the improved Buchberger's algorithm (see https://bit.ly/3IZ1u7y) produced correct results in every instance they were tested (by checking the resulting set satisfies Theorem 2.2.1), the speed at which they produced such correct results was extremely slow in comparison to the built-in functions. For instance it was seen that it took our implementation of the improved Buchberger's algorithm 40 minutes to find a Gröbner basis for Figure 3.5, but less than two seconds for the built-in one.

We saw that the Gröbner basis of the ideal generated by the graph colouring polynomials revealed more insights about the roots than the original set of polynomials. Moreover, it was also faster to find the common roots of the Gröbner basis than the original set of polynomials. After that, we discussed the amount of permutations of graph colourings. Theorem 3.5.1 provided us a way of determining an upper bound for the amount of permutations. Gröbner base revealed insights that assisted us in computing the specific number of permutations. Finally, we turned the problem of proving k-colourability to a problem of ideal membership with Theorem 3.6.1. We saw that for large graphs, it was easier to compute a Gröbner basis than to perform a division on the graph polynomial. For the medium sized graph, using a larger $k = 7$ made performing a division faster than using $k = 4$, its chromatic number.

Given more time we would have liked to significantly improve our implementation of the improved Buchberger's algorithm, for instance in [CLO15, Section 10.3] a new algorithm called the F4 Algorithm is introduced. Furthermore there exists algorithms for computing a reduced Gröbner basis but we could not complete an implementation of this in the given time frame. Then with an improved set of algorithms and code base, we would have liked to investigate the magic squares in non-linear cases, which unfortunately we did not have time to complete for this report.

Bibliography

[Stu05] Bernd Sturmfels. "What is ...a Gröbner basis?" In: *Notices of the American Mathematical Society* 52 (Jan. 2005).

[CLO15] David A. Cox, John B. Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2015.

[Meh] David Mehrle. *A Variety of Graph Coloring Problems*. URL: `https://pi.math.cornell.edu/~dmehrle/notes/old/promys14/graphcoloring.pdf`.