Sección de Matemáticas
Universidad de La Laguna

Karen Rodríguez Torres

Estudio de códigos binarios mediante Bases de Gröbner

Study of binary codes through Gröbner Bases

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Marzo de 2023

Dirigido por
Ignacio García Marco

Ignacio García Marco
Departamento de
Matemáticas,Estadística e I.O.
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Agradecer, principalmente a Nacho, gracias infinitas por tu paciencia, dedicación, por tu trabajo constante y por esa ilusión que tienes por las matemáticas, eres un gran ejemplo a seguir. Siguiendo, gracias a mi entorno, en especial a esas personas que me dió la carrera el primer año y que siguen conmigo actualmente. Por último, pero no menos importante, gracias a mi compañero de vida, gracias a tu apoyo incondicional he podido sacar fuerzas y afrontar una carrera tan dura y a la vez tan bonita como lo es matemáticas.

Karen Rodríguez Torres
La Laguna, 10 de marzo de 2023

Resumen · Abstract

Resumen

La Teoría de Códigos es una rama de la Matemática y de la Informática que tiene como objetivo enviar de forma eficiente un mensaje mediante un canal afectado de ruido, para que sea recuperado por el receptor sin errores. Con los códigos lineales se pretende introducir herramientas del Álgebra Lineal para resolver de forma eficiente problemas ligados a los códigos, como el problema de decodificación. Los parámetros principales de un código lineal son la longitud, la dimensión y la distancia mínima, los dos primeros van ligados a la eficiencia y el tercero a la capacidad correctora del código.

Dado un código binario, vamos a estudiar dos problemas. El primero, calcular su distancia mínima, y el segundo, la decodificación. Para ello, le asociaremos a todo código binario un ideal del anillo de polinomios y resolveremos los problemas anteriores haciendo uso de técnicas basadas en bases de Gröbner.

Palabras clave: *Códigos binarios, Códigos lineales, Distancia mínima, Bases de Gröbner, Decodificación*

Abstract

Coding theory is a branch of Mathematics and Computer Science that aims to efficiently send a message through a channel affected by noise, so that it can be recovered by the receiver without errors. With linear codes, the goal is to introduce tools from Linear Algebra to efficiently solve problems related to codes, such as the decoding problem. The main parameters of a linear code are length, dimension, and minimum distance. The first two are related to efficiency and the third to the correcting capacity of the code.
Given a binary code, we will study two problems. The first is to calculate its minimum distance, and the second is decoding. To do this, we will associate an ideal of the polynomial ring to every binary code and solve the above problems using Gröbner bases.

Keywords: *Binary codes, Linear codes, Minimum distance, Gröbner basis, Decoding*

Contenido

Introducción

La Teoría de Códigos es una rama de la Matemática y la Informática que se ocupa de la creación y análisis de códigos para la transmisión y almacenamiento de información. Los códigos se utilizan para representar información en un formato que pueda ser transmitido a través de un canal de comunicación, como una red de computadoras, y recuperado sin errores en el extremo receptor.

Ésta, se desarrolló en el siglo XX, y su historia se remonta a los primeros días de la telegrafía sin hilos y la transmisión de señales a larga distancia. En los años 30 y 40, los matemáticos e ingenieros empezaron a estudiar los problemas de transmisión de información y a desarrollar teorías y algoritmos para mejorar la eficiencia y la confiabilidad de la transmisión de información. El comienzo de la Teoría de Códigos y de la Teoría de la Información se ubica en el artículo "*A mathematical theory of communication*", escrito por Claude Shannon en el año 1948. En esta publicación, se define la capacidad del canal y se demuestra que siempre que no se supere esa capacidad, se puede comunicar información de manera fiable. Shannon asume que en cualquier canal habrá interferencias y propone una estrategia para lidiar con ellas, la redundancia. Pese a la innovación, la demostración de Shannon presentaba un inconveniente, y es que no era constructiva. Después de esta publicación, un hecho importante fue la creación del código Hamming, que permite corregir errores en la transmisión de información. Este código fue desarrollado por Richard Hamming, que había trabajado con Shannon en los Laboratorios Bell, en 1950 y se convirtió en un estándar en la industria de la transmisión de datos. Desde entonces, la teoría de códigos ha experimentado un desarrollo constante y ha evolucionado para incluir nuevos tipos de códigos, como códigos turbo y códigos LDPC, y nuevas aplicaciones, como la compresión de datos y la transmisión de información en redes inalámbricas.

El problema de comunicación con códigos correctores de errores sigue el esquema de la Figura 0.1. En él se suele utilizar como alfabeto el cuerpo finito de q elementos $\mathbb{F}_q$. Supongamos que queremos enviar un mensaje $m \in \mathbb{F}_q^k$, el codificador es la aplicación inyectiva $\text{Enc} : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$ con $n > k$ donde la idea es añadir información redundante. El conjunto $Im(\text{Enc}) \subseteq \mathbb{F}_q^n$, que es la

imagen de la aplicación, es lo que se denomina código. El mensaje codificado, se envía a través del canal, el cual, usualmente, se supone que no tiene pérdidas (si se emite un símbolo siempre se recibe un símbolo del mismo alfabeto que estamos utilizando), no tiene memoria (el error en cada símbolo no depende de los símbolos previamente enviados) y es simétrico (la probabilidad de error de cada símbolo es la misma). Como el canal no tiene pérdidas, el receptor recibe un mensaje de igual longitud pero suceptible de traer entradas erróneas. Aquí comienza el proceso de decodificación, en el que el receptor debe recuperar el mensaje enviado a partir del mensaje recibido; esto se podrá conseguir siempre y cuando no se hayan producido demasiados errores durante su transmisión. Dado que el canal no tiene memoria y es simétrico, lo más probable es que el mensaje original sea el mensaje modificado tal que difiera en el menor número de entradas posible con el mensaje recibido; esto es lo que se conoce como decodificación por máxima verosimilitud.

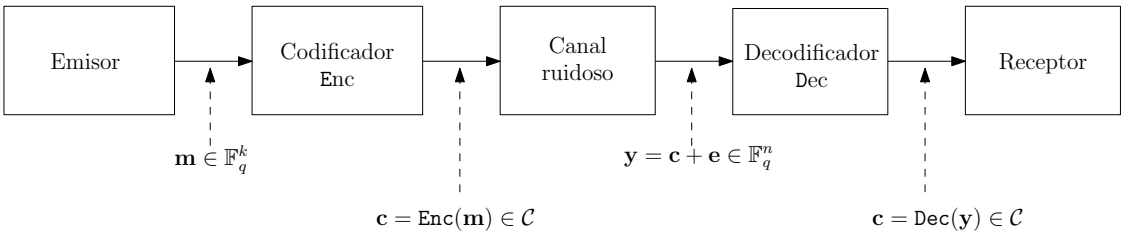


Figura 0.1. Esquema de comunicación.

El proceso de decodificación es el más delicado de todo el proceso, consiste en recibir una palabra del código perturbada y devolver una palabra del código a la distancia mínima más cercana. La decodificación se puede realizar mediante una variedad de técnicas, como la decodificación por fuerza bruta, la decodificación con diferencia de códigos, la decodificación basada en matrices o la decodificación por medio de bases de Gröbner. La selección de una técnica de decodificación depende de las necesidades específicas de una aplicación y puede tener un impacto significativo en la eficiencia y precisión de la decodificación.

En concreto, los códigos lineales son subespacios vectoriales de $\mathbb{F}_q^n$ y son un tipo de códigos que se basan en el Álgebra Lineal para su construcción y análisis. En la Teoría de Códigos, se utiliza el Álgebra Lineal para describir las relaciones entre las palabras codificadas y las palabras originales, y para analizar las propiedades de corrección de errores de los códigos. Además utilizamos vectores y matrices para analizar esos códigos. Por ejemplo, la matriz Generatriz sirve para codificar de manera eficiente y, además sólo viendo su tamaño vamos a conocer la dimensión y la longitud del código. El Álgebra Lineal también se utiliza para

construir códigos con propiedades específicas, como la capacidad de corregir un número determinado de errores o para aumentar la eficiencia en la transmisión de información. Por ejemplo, los códigos de Reed-Solomon y los códigos de BCH son códigos lineales que se utilizan comúnmente en aplicaciones como la transmisión de datos por satélite, la transmisión de audio y video, y la transmisión de información en medios de almacenamiento. Los principales parámetros de un código lineal $\mathcal{C} \subseteq \mathbb{F}_q^n$ incluyen:

- Longitud: La longitud de $\mathcal{C} \subseteq \mathbb{F}_q^n$ es $n = n(\mathcal{C})$ y hace referencia al número de bits en una palabra codificada.
- Dimensión: La dimensión de un código lineal $\mathcal{C}$ se denota por $k(\mathcal{C})$ y es su dimensión como $\mathbb{F}_q$-espacio vectorial.
- Distancia mínima: La distancia mínima de un código lineal $\mathcal{C}$ es la mínima distancia entre dos palabras del código, y coincide con el peso mínimo de una palabra no nula. La distancia mínima la denotaremos como $d(\mathcal{C})$. Este valor está ligado a la capacidad de corrección de errores del código.

Los códigos binarios son aquellos códigos lineales que usan $\mathbb{F}_2$ como alfabeto. El uso del sistema binario en teoría de códigos es debido a la sencillez de tener que trabajar con dos dígitos, la flexibilidad de representar cualquier número y la facilidad de que los ordenadores ya tienen implementado este sistema. En este trabajo estudiaremos dos problemas difíciles desde el punto de vista computacional en Teoría de Códigos: el cálculo de la distancia mínima y el problema de la decodificación, haciendo especial énfasis en los códigos binarios. Cabe mencionar que ambos problemas están clasificados como $\mathcal{NP}$-duros [3] incluso para códigos binarios, así que no se espera encontrar algoritmos rápidos (polinomiales) que los resuelvan.

En este trabajo, primero hablaremos de la decodificación por fuerza bruta de cualquier código y, de la decodificación por descenso de gradiente de un código binario. La decodificación por fuerza bruta implica comprobar enumerando todas las posibilidades, si hay una palabra en el código próxima a la recibida como entrada. Es un método exhaustivo que garantiza la corrección de la decodificación, pero también es computacionalmente muy costoso y no es viable para códigos de longitud significativa. La decodificación por fuerza bruta se utiliza principalmente en situaciones en las que se conocen pocos detalles sobre la codificación original o cuando se requiere una garantía absoluta de corrección. Sin embargo, debido a su costo computacional, la decodificación por fuerza bruta es muy poco viable para la decodificación en tiempo real en aplicaciones prácticas.

Otro de los algoritmos de decodificación de los que hablaremos, será de la decodificación por descenso de gradiente. Es un método que se usa con códigos binarios, y es un poco costoso porque necesitamos saber el conjunto de palabras minimales del código. Éste, es un algoritmo veroz (*greedy* en inglés) y consiste en, dado $x \in \mathbb{F}_q^n$, buscar una palabra minimal m tal que el peso de $x + m$ sea

menor que el de x e iterar el proceso hasta que no se pueda hacer descender más el peso. Al final, como demostraremos, este proceso devolverá una de las palabras del código más próxima a la dada inicialmente.

Tras esto presentaremos los resultados principales de esta memoria, que proponen métodos de decodificación y del cálculo de la distancia mínima en códigos binarios basados en el cálculo de bases de Gröbner en el anillo de polinomios en varias variables $\mathbb{K}[x_1, \ldots, x_n]$ sobre un cuerpo $\mathbb{K}$.

Las bases de Gröbner son una herramienta matemática utilizada en Álgebra Computacional que se ideó con el objetivo de resolver sistemas de ecuaciones polinómicas. Las introdujo Bruno Buchberger en su Tesis Doctoral en 1965. Desde su aparición se han encontrado innumerables aplicaciones de las bases de Gröbner en Matemáticas y Ciencias de la Computación. Algunos de los usos más comunes incluyen, la resolución de sistemas de ecuaciones polinómicas, útil en aplicaciones en álgebra computacional, como la teoría de números y la criptografía, análisis de sistemas dinámicos, incluyendo el control automático, la robótica y la dinámica de sistemas mecánicos, investigación en geometría algebraica, para resolver problemas relacionados con la geometría de objetos algebraicos, como curvas y superficies.

Una base de Gröbner es un conjunto particular de polinomios generadores de un ideal, que permite determinar la pertenencia de un polinomio a un ideal. Una vez que se tiene una base de Gröbner para un ideal, se puede resolver el problema de pertenencia en un ideal utilizando un algoritmo de la división. Este algoritmo reduce un polinomio a su forma canónica utilizando las relaciones entre los polinomios de la base de Gröbner. Si un polinomio se reduce a cero, significa que pertenece al ideal, mientras que si no se reduce a cero, significa que no pertenece al ideal. Estudiaremos en detalle las bases de Gröbner reducidas, que aportan un sistema generador canónico del ideal y sirven, por ejemplo, para comprobar si dos ideales son iguales. Haremos énfasis en el estudio de las bases de Gröbner de ideales binomiales porque serán de utilidad en el estudio de los códigos binarios.

Concluyendo, este trabajo se va a dividir en tres capítulos, el primero, tratará sobre las Bases de Gröbner, el segundo, será una introducción a la Teoría de Códigos y Códigos Lineales, y por último, en el tercer capítulo, le asociaremos a todo código binario $\mathcal{C} \subseteq \mathbb{F}_2^n$ un ideal $I(\mathcal{C})$ binomial en $\mathbb{K}[x_1, \ldots, x_n]$ y veremos cómo utilizar las bases de Gröbner para: (1) calcular la distancia mínima de $\mathcal{C}$ y (2) obtener algoritmos de decodificación. En particular, propondremos dos algoritmos de decodificación por bases de Gröbner, el primero que utiliza el Algoritmo de la división y el segundo, que mejora al anterior, combina las bases de Gröbner con ideas de la decodificación por descenso de gradiente.

A lo largo de esta memoria, se ha hecho uso de algunos ejemplos realizados en `Sagemath` [13], un sistema algebraico computacional que destaca por estar construido sobre paquetes matemáticos ya contrastados como `Singular`, `Sympy`,

PARI/GP o Maxima y por acceder a sus potencias combinadas a través de un lenguaje común basado en Python.

1

Bases de Gröbner

Las bases de Gröbner son una de las principales herramientas en Álgebra Conmutativa Computacional y tienen su origen en la Tesis Doctoral de Bruno Buchberger [6]. Dado un ideal J del anillo de polinomios $\mathbb{K}[x_1, \ldots, x_n]$ sobre un cuerpo $\mathbb{K}$, una base de Gröbner de J es un sistema generador de J con ciertas propiedades adicionales que permiten, entre otros, resolver el problema de pertenencia a J de forma efectiva; es decir, determinar si un $f \in \mathbb{K}[x_1, \ldots, x_n]$ dado pertenece a J o no.

En este capítulo, haremos un recorrido por las bases de Gröbner, indicando brevemente qué son, cómo se pueden calcular y algunas de sus utilidades. En el próximo capítulo, a cada código binario le asociaremos un ideal binomial y estudiaremos el código por medio del ideal asociado. Es por ello que en este capítulo haremos especial énfasis en el estudio de las bases de Gröbner de un ideal binomial (remitimos al lector al artículo [8] para más información sobre ideales binomiales). No incluimos la demostración de la mayoría de resultados de este capítulo, para ello remitimos al lector a [7].

En repetidas ocasiones nos apoyaremos en el Teorema de la base de Hilbert y por ello presentaremos un enunciado del mismo a continuación.

Teorema 1.1. *Sea J un ideal de $\mathbb{K}[x_1, \ldots, x_n]$ anillo de polinomios en n variables sobre un cuerpo $\mathbb{K}$. Si $J = \langle f_i : i \in I \rangle$, entonces $\exists\ i_1, \ldots, i_r \in I$ tales que $J = \langle f_{i_1}, \ldots, f_{i_r} \rangle$*

En particular, todo ideal está finitamente generado y, por tanto, el anillo de polinomios es noetheriano. Para una demostración de este teorema, recomendamos visitar [7, Capítulo 2.5].

1.1. Órdenes monomiales

Sea $\mathbb{K}[x_1, \ldots, x_n]$ el anillo de polinomios en n variables sobre un cuerpo $\mathbb{K}$. Un órden monomial es un orden total sobre el conjunto de monomios de $\mathbb{K}[x_1, \ldots, x_n]$ con ciertas propiedades adicionales.

Definición 1.2. *Un monomio en* $\mathbb{K}[x_1, \ldots, x_n]$ *es un polinomio de la forma* $x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ *con* $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. *Denotamos por* $|\alpha| := \alpha_1 + \ldots + \alpha_n$ *al grado del monomio* x^α.

Definición 1.3. *Un orden monomial en* $\mathbb{K}[x_1, \ldots, x_n]$ *es una relación binaria* " $\geq$ " *en el conjunto de monomios* $M = \{x^\alpha : \alpha \in \mathbb{N}^n\}$ *de* $\mathbb{K}[x_1, \ldots, x_n]$ *verificando:*

1. " $\geq$ " *es orden total en* M.
2. *Sean* $x^\alpha, x^\beta, x^\gamma \in M$ *tales que* $x^\alpha \geq x^\beta$, *entonces* $x^\alpha x^\gamma \geq x^\beta x^\gamma$ *(es decir, el orden es compatible con el producto de monomios).*
3. $x^\alpha \geq x^{(0,\ldots,0)} = 1$ *para todo* $x^\alpha \in M$.

Mediante la identificación $M \to \mathbb{N}^n$ *definida por* $x^\alpha \to \alpha$, *los órdenes monomiales pueden ser tratados como relaciones binarias* " $\geq$ " *en* $\mathbb{N}^n$ *verificando:*

1. " $\geq$ " *es orden total en* $\mathbb{N}^n$.
2. *Si* $\alpha, \beta, \gamma \in \mathbb{N}^n$ *tal que* $\alpha \geq \beta$ *entonces* $\alpha + \gamma \geq \beta + \gamma$ *(es decir, es compatible con la suma de* $\mathbb{N}^n$*).*
3. $\alpha \geq (0, \ldots, 0)$ *para todo* $\alpha \in \mathbb{N}^n$.

Definición 1.4. *Un orden monomial* " $\geq$ " *se dice graduado si para* $\alpha, \beta \in \mathbb{N}^n$, $\alpha > \beta$ *siempre que* $|\alpha| > |\beta|$.

Definición 1.5 (Orden lexicográfico). *Sean dos n-tuplas* $\alpha = (\alpha_1, \ldots, \alpha_n)$ *y* $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. *Diremos que* $\alpha >_{lex} \beta$ *o que* $x^\alpha >_{lex} x^\beta$ *si la primera entrada no nula de* $\alpha - \beta$ *por la izquierda es positiva.*

Si $n \geq 2$, entonces el orden lexicográfico no es graduado, ya que por ejemplo, $x_1 > x_2^2$. Veamos algunos órdenes graduados:

Definición 1.6 (Orden lexicográfico graduado). *Sean* $\alpha = (\alpha_1, \ldots, \alpha_n)$ *y* $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$, *diremos que* $\alpha >_{deglex} \beta$ *si* $|\alpha| > |\beta|$ *o si* $|\alpha| = |\beta|$ *y* $\alpha >_{lex} \beta$.

Definición 1.7 (Orden lexicográfico inverso graduado). *Consideramos* $\alpha = (\alpha_1, \ldots, \alpha_n)$ *y* $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$, *diremos que* $\alpha >_{grevlex} \beta$ *si* $|\alpha| > |\beta|$ *o si* $|\alpha| = |\beta|$ *y la primera entrada no nula por la derecha de* $\alpha - \beta$ *es negativa.*

Proposición 1.8. *Los órdenes lexicográfico, lexicográfico graduado y lexicográfico inverso graduado son órdenes monomiales.*

Ejemplo 1.9. En $\mathbb{N}^4$, tomando $\alpha = (2, 2, 0, 1)$, $\beta = (3, 0, 0, 2)$, $\gamma = (4, 0, 0, 0)$, se tiene que $\alpha >_{degrevlex} \beta >_{degrevlex} \gamma$, que $\beta >_{deglex} \alpha >_{deglex} \gamma$ y que $\gamma >_{lex} \beta >_{lex} \alpha$.

Definición 1.10. *Sea* $f = \sum a_\alpha x^\alpha$ *un polinomio no nulo en* $\mathbb{K}[x_1, \ldots, x_n]$ *y sea* " $\geq$ " *un orden monomial. Se definen:*

- *el multigrado de f como* $\text{mdeg}(f) := \text{máx}_{\geq}\{\alpha \in \mathbb{N}^n : a_\alpha \neq 0\}$,
- *el coeficiente principal de f es* $\text{LC}(f) = a_{mdeg(f)} \in \mathbb{K}$, *y*
- *el monomio principal de f es* $\text{LM}(f) = x^{mdeg(f)}$.

Ejemplo 1.11. Continuando con el ejemplo anterior. Sea $f = 3x^\alpha + 2x^\beta - x^\gamma = 3x_1^2x_2^2x_4 + 2x_1^3x_4^2 - x_1^4$: Para $>_{lex}$ se tiene que $\text{mdeg}(f) = \gamma = (4,0,0,0)$, $\text{LC}(f) = -1$ y $\text{LM}(f) = x_1^4$.
Para $>_{deglex}$ se tiene que $\text{mdeg}(f) = \beta = (3,0,0,2)$, $\text{LC}(f) = 2$ y $\text{LM}(f) = x_1^3x_4^2$.
Para $>_{degrevlex}$ se tiene que $\text{mdeg}(f) = (2,2,0,1)$, $\text{LC}(f) = 3$ y $\text{LM}(f) = \alpha = x_1^2x_2^2x_4$.

Proposición 1.12. *Sean f,g* $\in \mathbb{K}[x_1, \ldots, x_n]$ *polinomios no nulos. Entonces:*

1. $\text{mdeg}(fg) = \text{mdeg}(f) + \text{mdeg}(g)$.
2. *Si* $f + g \neq 0, \text{mdeg}(f+g) \leq \text{máx}(\text{mdeg}(f), \text{mdeg}(g))$. *En particular, si* $\text{mdeg}(f) \neq \text{mdeg}(g)$, *se da la igualdad.*

1.2. Algoritmo de la División en $\mathbb{K}[x_1, \ldots, x_n]$

En $\mathbb{K}[x_1, \ldots, x_n]$ no existe un algoritmo de división canónico, y por tanto, $\mathbb{K}[x_1, \ldots, x_n]$ no es un dominio euclídeo. En esta sección desarrollaremos un método que recibe como entrada un polinomio f y una familia de polinomios, $F = (f_1, \ldots, f_s)$, y devuelve cierto polinomio r tal que $f - r \in \langle f_1, \ldots, f_s \rangle$ donde $\langle f_1, \ldots, f_s \rangle$ denota el ideal de $\mathbb{K}[x_1, \ldots, x_n]$ generado por $f_1, \ldots, f_s$. Como consecuencia si $r = 0$, entonces $f \in \langle f_1, \ldots, f_s \rangle$. No obstante, el valor de r en el algoritmo propuesto va a depender de la elección de un orden monomial " $\geq$ " prefijado y de la ordenación de la s-tupla $(f_1, \ldots, f_s)$. En consecuencia esto nos va a proporcionar un algoritmo que resuelva el problema de pertenencia a un ideal.

Teorema 1.13. *Fijado " $\geq$ " un orden monomial en* $\mathbb{N}^n$ *y* $F = (f_1, \ldots, f_s)$ *una s-tupla de polinomios de* $\mathbb{K}[x_1, \ldots, x_n]$, *Entonces, todo* $f \in \mathbb{K}[x_1, \ldots, x_n]$ *se puede expresar como:*

$$f = q_1f_1 + \ldots + q_sf_s + r,$$

con $q_i \in \mathbb{K}[x_1, \ldots, x_n]$ *para* $1 \leq i \leq s$, $r \in \mathbb{K}[x_1, \ldots, x_n]$ *y* $r = 0$ *ó bien r es una combinación lineal de monomios de* $\mathbb{K}[x_1, \ldots, x_n]$ *y ninguno de ellos divisible por* $\text{LM}(f_i)$ *para* $1 \leq i \leq s$. *Además, si* $q_if_i \neq 0$, *entonces* $\text{mdeg}(f) \geq \text{mdeg}(q_if_i)$.

Este teorema se puede demostrar haciendo uso del Algoritmo de la división de la Figura 1.1.

Definición 1.14. *Llamaremos resto r de la división de f entre* $F = (f_1, \ldots, f_s)$ *y lo denotaremos por* $\overline{f}^F$ *al obtenido, siguiendo el Algoritmo de la División expuesto en la Figura 1.1.*

El resto de la división además del orden monomial elegido, depende del orden de $f_1, \ldots, f_s$, como muestra el siguiente ejemplo.

Ejemplo 1.15. Vamos a realizar la división de $f = x^2y^3 - xy$ entre $f_1 = x^2y^2 + 1$ y $f_2 = xy^2 - 1$ considerando como orden monomial $>_{lex}$ y siguiendo el Algoritmo 1.1.
Vamos a dividir f entre $F = (f_1, f_2)$ y se tiene que $\mathrm{LM}(f) = x^2y^3$ y que $\mathrm{LM}(f_1) = x^2y^2$ divide a $\mathrm{LM}(f)$. Así, $f^{(1)} = f - y \cdot f_1 = -xy - y$. Ahora, $\mathrm{LM}(f^{(1)}) = -xy$ y ni $\mathrm{LM}(f_1)$ ni $\mathrm{LM}(f_2)$ dividen a $\mathrm{LM}(f^{(1)})$, luego $r_1 = -xy$ pasa a ser parte del resto.
Tenemos a continuación que $f^{(2)} = f^{(1)} - r_1 = -y$ y observamos que ni $\mathrm{LM}(f_1)$ ni $\mathrm{LM}(f_2)$ dividen a $\mathrm{LM}(f^{(2)})$, luego $r_2 = -y$ pasaría al resto y tendríamos que $r = r_1 + r_2 = -xy - y$ es un resto de la división, obteniendo $f = y \cdot (x^2y^2 + 1) + 0 \cdot (xy^2 - 1) + (-xy - y)$.

Veamos a continuación qué ocurre si alteramos el orden de los polinomios f_1 y f_2. Es decir, dividiremos f entre $F' = (f_2, f_1)$, es decir, primero por f_2 y luego por f_1. Se tiene que $\mathrm{LM}(f) = x^2y^3$ y $\mathrm{LM}(f_2)$ lo divide, luego $f^{(1)} = f - xy \cdot f_2 = 0$ y habría finalizado la división. Así, $f = xy \cdot (xy^2 - 1) + 0 \cdot (x^2y^2 + 1)$.

Concluimos entonces lo dicho anteriormente, que el orden en el que se hace la división es determinante, ya que en el primer caso, $\overline{f}^F = -xy - y$ y en el segundo caso, al cambiar el orden, $\overline{f}^{F'} = 0$.

Algoritmo de la división

Entrada: $f, f_1, \ldots, f_s \in \mathbb{K}[x_1, \ldots, x_n]$ y " $\geq$ " un orden monomial.
Salida: $q_1, \ldots, q_s, r \in \mathbb{K}[x_1, \ldots, x_n]$ tales que $f = \sum q_i \cdot f_i + r$
Inicialización: $f^{(0)} := f, r^{(0)} = 0, q_1^{(0)} = \cdots = q_s^{(0)} = 0$, $i = 0$.
repeat
  if existe $j = \min\{k : \mathrm{LM}(f_k) \mid \mathrm{LM}(f^{(i)})\}$. **then**
    $f^{(i+1)} = f^{(i)} - \frac{\mathrm{LC}(f^{(i)}) \cdot \mathrm{LM}(f^{(i)})}{\mathrm{LC}(f_j) \cdot \mathrm{LM}(f_j)} f_j$
    $r^{(i+1)} = r^{(i)}$
    $q_j^{(i+1)} = q_j^{(i)} + \frac{\mathrm{LC}(f^{(i)}) \cdot \mathrm{LM}(f^{(i)})}{\mathrm{LC}(f_j) \cdot \mathrm{LM}(f_j)}$
    $q_k^{(i+1)} = q_k^{(i)}$ $\forall k \neq j$
  else
    $f^{(i+1)} = f^{(i)} - \mathrm{LC}(f^{(i)}) \cdot \mathrm{LM}(f^{(i)})$
    $r^{(i+1)} = r^{(i)} + \mathrm{LC}(f^{(i)}) \cdot \mathrm{LM}(f^{(i)})$
    $q_k^{(i+1)} = q_k^{(i)}$, $\forall k$
  end if
  $i := i + 1$
until $f^{(i)} = 0$;
return $q_1^{(i)}, \ldots, q_s^{(i)}, r^{(i)}$

Figura 1.1. Algoritmo de la división

Corolario 1.16. *Sean* $f, f_1, \ldots, f_s \in \mathbb{K}[x_1, \ldots, x_n]$ *y* $F = (f_1, \ldots, f_s)$. *Si* $\overline{f}^F = 0$, *entonces* $f \in \langle f_1, \ldots, f_s \rangle$.

Este corolario no es una condición necesaria y suficiente ya que, como hemos visto en el ejemplo anterior, dependiendo del orden escogido, el resto es distinto, y puede pasar que el resto sea cero en un caso, y en otro caso, sea distinto de cero.

1.3. Ideales monomiales

Los ideales monomiales en $\mathbb{K}[x_1, \ldots, x_n]$ forman una familia importante de ideales ya que, por un lado, varios problemas de ideales (como el de pertenencia) se pueden resolver facilmente si el ideal es monomial y, por otro lado, la idea detrás de las bases de Gröbner es que varios problemas sobre ideales generales pueden reducirse al caso de un ideal monomial. En esta sección vamos a presentar su definición, así como algunas de sus principales propiedades.

Definición 1.17. *Un ideal* J *de* $\mathbb{K}[x_1, \ldots, x_n]$ *se dice que es un ideal monomial, si existe un subconjunto* $A \subseteq \mathbb{N}^n$ *tal que* $J = \langle x^\alpha : \alpha \in A \rangle$. *Por el Teorema de la base de Hilbert 1.1 podemos suponer que* A *es finito, sin pérdida de generalidad.*

Los dos siguientes lemas justifican que en un ideal monomial, el problema de pertenencia es fácil de resolver.

Lema 1.18. *Sea* $J = \langle x^\alpha : \alpha \in A \rangle$ *un ideal monomial. Entonces* $x^\beta \in J$ *si y solo si* x^β *es divisible por* x^α *para algún* $\alpha \in A$.

Lema 1.19. *Sea* J *un ideal monomial y* $f \in \mathbb{K}[x_1, \ldots, x_n]$. *Las siguientes afirmaciones son equivalentes:*

1. $f \in J$.
2. *Todos los términos de* f *pertenecen a* J.
3. f *es* $\mathbb{K}$*-combinación lineal de monomios de* J.

Definición 1.20. *Sea* J *un ideal monomial. Diremos que* $\{x^{\alpha(1)}, \ldots, x^{\alpha(s)}\}$ *es una base mínima si* $J = \langle x^{\alpha(1)}, \ldots, x^{\alpha(s)} \rangle$ *y se verifica que* $x^{\alpha(j)}$ *no divide a* $x^{\alpha(i)}$ *para todo* $i \neq j$.

Proposición 1.21. *Todo ideal monomial tiene una única base mínima.*

Ejemplo 1.22. El ideal $J = \langle x_1^3 + x_1 x_2^4, x_2^3, x_1^2 x_2^2, x_1^2 x_2^2 + x_1^4 \rangle$ es un ideal monomial, su base mínima es $M = \{x_1^3, x_2^3, x_1^2 x_2^2\}$.

Llamaremos K al ideal monomial generado por M. Es sencillo ver que los generadores de J están en K, usando los lemas 1.18 y 1.19, por ejemplo, ver que está $x_1^3 + x_1 x_2^4$, sería ponerlo como combinación de $x_1^3 + (x_1 x_2 \cdot (x_2^3)$. En cambio,

para ver que $M \subseteq J$, tendremos que realizar combinaciones de los generadores de J para poder verificar que son elementos de J, por ejemplo, para conseguir x_1^3 podemos escribirlo como $x_1^3 + x_1x_2^4 - (x_1x_2) \cdot (x_2^3)$. De esta forma, podemos conseguir todos los generadores del ideal J.

Como consecuencia de la Proposición 1.21 se tiene que dos ideales monomiales son iguales si y solo si tienen la misma base mínima.

1.4. Bases de Gröbner

Para un ideal dado y un orden monomial fijado, le vamos a asociar una base de Gröbner, que es un sistema generador del ideal con ciertas propiedades adicionales. Este objeto nos va a permitir resolver, entre otros, el problema de pertenencia a un ideal. En esta sección definiremos las bases de Gröbner, veremos cómo calcularlas mediante el algoritmo de Buchberger y algunas de sus propiedades.

Definición 1.23. *Sea J un ideal no nulo de $\mathbb{K}[x_1, \ldots, x_n]$ y " $\geq$ " un orden monomial en el anillo de polinomios. El ideal monomial*

$$\langle \mathrm{LM}(J) \rangle = \langle \mathrm{LM}(f) : f \in J - \{0\} \rangle \subseteq \mathbb{K}[x_1, \ldots, x_n]$$

se denomina ideal inicial de J respecto de " $\geq$ ".

Definición 1.24 (Base de Gröbner). *Fijado un orden monomial " $\geq$ " en $\mathbb{K}[x_1, \ldots, x_n]$ y un ideal $J \subseteq \mathbb{K}[x_1, \ldots, x_n]$, $J \neq \{0\}$, un subconjunto finito $G = \{g_1, \ldots, g_t\} \subseteq J$ se denomina base de Gröbner de J respecto de " $\geq$ " si $\langle \mathrm{LM}(J) \rangle = \langle \mathrm{LM}(g_1), \ldots, \mathrm{LM}(g_t) \rangle$.*

Observación 1.25. Sea J un ideal no nulo de $\mathbb{K}[x_1, \ldots, x_n]$. Entonces, por el Teorema de la base de Hilbert, existen $g_1, \ldots, g_t \in J$ tales que

$$\langle \mathrm{LM}(J) \rangle = \langle \mathrm{LM}(g_1), \ldots, \mathrm{LM}(g_t) \rangle$$

En consecuencia, fijado " $\geq$ " un orden monomial, todo ideal admite una base de Gröbner.

Proposición 1.26. *Dado un orden monomial " $\geq$ ", y un ideal $J \subseteq \mathbb{K}[x_1, \ldots, x_n]$, toda base de Gröbner de J es un sistema generador del ideal.*

Proposición 1.27. *Sea J un ideal de $\mathbb{K}[x_1, \ldots, x_n]$. Dado $f \in \mathbb{K}[x_1, \ldots, x_n]$ y fijado un orden monomial " $\geq$ ", existe un único $r \in \mathbb{K}[x_1, \ldots, x_n]$ verificando:*

1. *Ningún término de r pertenece a $\langle \mathrm{LM}(J) \rangle$.*
2. *$f - r \in J$.*

Además, este r se obtiene al aplicar el algoritmo de la división de f entre $(g_1, \ldots, g_t)$ siendo $G = \{g_1, \ldots, g_t\}$ una base de Gröbner de f respecto de " $\geq$ ".

De la proposición se deduce que el resto de la división entre una base de Gröbner G no depende de la ordenación de los elementos de G. Como consecuencia directa de la Proposición 1.27 se tiene el siguiente resultado, que explica cómo resolver el problema de pertenencia a un ideal a partir de una base de Gröbner.

Corolario 1.28. *Sean J un ideal de $\mathbb{K}[x_1, \ldots, x_n]$, " $\geq$ " un orden monomial $G = \{g_1, \ldots, g_t\}$ una base de Gröbner de J respecto de " $\geq$ " y $f \in \mathbb{K}[x_1, \ldots, x_n]$. Entonces, $f \in J$ si y solo si el resto de la división de f entre G es cero, es decir, $\overline{f}^G = 0$.*

Veamos ahora cómo calcular una base de Gröbner a partir de un sistema generador del ideal.

Definición 1.29. *Sean $f,g \in \mathbb{K}[x_1, \ldots, x_n] - \{0\}$. Se define el S-polinomio de f y g, denotado por $S(f, g)$ como:*

$$S(f,g) := \frac{x^\gamma}{\mathrm{LC}(f)\cdot\mathrm{LM}(f)} f - \frac{x^\gamma}{\mathrm{LC}(g)\cdot\mathrm{LM}(g)} g$$

donde tenemos que $\mathrm{mdeg}(f) = \alpha$, $\mathrm{mdeg}(g) = \beta$ *y* $\gamma = (\gamma_1, \ldots, \gamma_s)$ *donde* $\gamma_i = \max(\alpha_i, \beta_i)$ *para cada i, es decir, x^γ es el mínimo común múltiplo de* $\mathrm{LM}(f)$ *y* $\mathrm{LM}(g)$.

De la definición de S-polinomio se observa que $\mathrm{mdeg}(S(f, g)) < \gamma$.

Veamos a continuación el criterio y el algoritmo de Buchberger que nos van a describir cómo construir una base de Gröbner.

Teorema 1.30. *(Criterio de Buchberger) Sea J un ideal de $\mathbb{K}[x_1, \ldots, x_n]$ y sea $G = \{g_1, \ldots, g_t\}$ un sistema generador de J. Fijado un orden monomial " $\geq$ ", G es base de Gröbner de J si y solo si para todo $i \neq j$, el resto de dividir $S(g_i, g_j)$ entre G es cero.*

Teorema 1.31. *(Algoritmo de Buchberger) Sea $J = \langle f_1, \ldots, f_s \rangle$ un ideal de $\mathbb{K}[x_1, \ldots, x_n]$. El algoritmo de la Figura 1.2 construye una base de Gröbner de J en un número finito de pasos.*

Ejemplo 1.32. Retomando el ejemplo anterior 1.15, y usando de herramienta al programa `Sagemath`, tenemos:

```
In : P.<x,y> = PolynomialRing(QQ,2,order='degrevlex')
In : I=ideal(x^2*y^2 + 1 ,x*y^2-1)
In : B=I.groebner_basis()
Out: [y^2 + 1, x + 1]
In : f=x^2*y^3-x*y
In : f.reduce(B)
Out: 0
```

Algoritmo de Buchberger

Entrada: $F = \{f_1, \ldots, f_s\}$ un sistema generador de J y " $\geq$ " un orden monomial.
Salida: $G = \{g_1, \ldots, g_t\}$ una base de Gröbner de J, con $F \subseteq G$.
$G := F$
repeat
  $\tilde{G} := G$.
  for $(p, q) \in \tilde{G}^2$ con $p \neq q$ **do**
    $r := \overline{S(p,q)}^{\tilde{G}}$
    if $r \neq 0$ **then**
      $G := G \cup \{r\}$
    end if
  end for
until $G = \tilde{G}$;
return G

Figura 1.2. Algoritmo de Buchberger

Tomando como ideal el generado por $F = (f_1, f_2)$, y como orden monomial el orden lexicográfico inverso graduado, tenemos que la base de Gröbner es $B = \{y^2 + 1, x + 1\}$ y queremos ver que $f = x^2y^3 - xy \in (f_1, f_2)$ usando únicamente la base de Gröbner. Para ello, realizamos la división de f entre B con el comando $f.reduce(B)$ y nos da resto cero. Por lo tanto, usando el Corolario 1.28, concluimos que f está en el ideal dicho.

1.4.1. Base de Gröbner reducida

Para un ideal y un orden no existe una única base de Gröbner, con el fin de conseguir unicidad surge el concepto de base de Gröbner reducida. En esta subsección definiremos este concepto y demostraremos esa unicidad buscada. Una consecuencia directa, es que nos da un criterio para decidir si dos ideales son el mismo sólo comparando sus bases de Gröbner reducidas.

Definición 1.33. *Fijado un orden monomial " $\geq$ " y un ideal I, decimos que una base de Gröbner $G = \{g_1, \ldots, g_t\}$ de I es minimal si ningún subconjunto propio de G es base de Gröbner de I. Esto es equivalente a que $\{\mathrm{LM}(g_1), \ldots, \mathrm{LM}(g_t)\}$ sea la base minimal del ideal monomial $\langle \mathrm{LM}(I) \rangle$.*

Un mismo ideal puede tener varias bases de Gröbner minimales.

Ejemplo 1.34. Fijado el orden monomial $>_{lex}$ definido anteriormente en la Definición 1.5 y sea $I = \langle x^2 + y, y^2 \rangle$, podemos encontrar dos bases de Gröbner minimales respecto del orden monomial fijado. Proponemos $G = \{x^2 + y, y^2\}$ y $G' = \{x^2 + y^2 + y, y^2\}$, demostraremos que G es base de Gröbner minimal. Como

$G' \subseteq I$ y los elementos de G' tienen los mismos términos iniciales que los de G, entonces, ésta también será base de Gröbner minimal. Denotamos $f = x^2 + y$ y $g = y^2$.

Para que G sea base de Gröbner, tendrá que cumplir que el resto de dividir $S(f, g)$ entre G es cero. Tenemos que $\mathrm{LM}(f) = x^2$ y $\mathrm{LM}(g) = y^2$, como $\mathrm{LC}(f) = \mathrm{LC}(g) = 1$, tenemos:

$$S(f, g) = y^2 \cdot f - x^2 \cdot g = x^2y^2 + y^3 - x^2y^2 = y^3$$

Por lo tanto, tomamos $h = y^3$, y siguiendo el algoritmo de la división descrito en la Figura 1.1 se observa que $\mathrm{LM}(f)$ no divide a y^3 y que $\mathrm{LM}(g)$ sí divide a y^3, por lo que $h = h - (y \cdot g) = 0$. Por lo tanto $\{f, g\}$ es base de Gröbner de I respecto de $>_{lex}$ y además es minimal ya que $\langle x^2, y^2\rangle$ es la base minimal del ideal $\langle \mathrm{LM}(I)\rangle$.

Definición 1.35. *Una base de Gröbner G de un ideal polinomial I se dice que es reducida si:*

1. $\mathrm{LC}(p) = 1$ *para todo* $p \in G$
2. *Si* $p \in G$, *ningún monomio de* p *pertenece a* $\langle \mathrm{LM}(G - \{p\})\rangle$.

De la definición se deduce que toda base de Gröbner reducida es minimal.

Proposición 1.36. *Sea $I \neq \{0\}$ un ideal polinomial. Entonces, para un orden monomial dado, tenemos una única base de Gröbner reducida.*

Demostración. Sean $g_1, \ldots, g_t \in I$ con $\mathrm{LC}(g_i) = 1$ tales que $\langle \mathrm{LM}(I)\rangle = \langle \mathrm{LM}(g_1), \ldots, \mathrm{LM}(g_t)\rangle$ y sea una base de Gröbner minimal $G = \{g_1, \ldots, g_t\}$ de I. Definimos $g' = \overline{g}^{G-\{g\}}$ y $G' = (G - \{g\}) \cup \{g'\}$, afirmamos que G' es una base minimal de Gröbner de I, para comprobarlo, vemos que $\mathrm{LM}(g') = \mathrm{LM}(g)$ porque cuando dividimos g entre $G - \{g\}$, $\mathrm{LM}(g)$ va al resto ya que no es divisible por ningún elemento de $\mathrm{LM}(G - \{g\})$, y los otros términos que aparecen en la división son menores. Entonces $\langle \mathrm{LM}(G')\rangle = \langle \mathrm{LM}(G)\rangle$, y, además, por el Teorema 1.31 ningún término de g' pertenece a $\langle LT(G' - \{g'\})\rangle$. Si repetimos este procedimiento con todos los elementos de G, y terminaremos con una base de Gröbner reducida. Para probar la unicidad, supongamos que G y $\tilde{G}$ son bases de Gröbner reducidas de un mismo ideal. Hemos visto que si G y $\tilde{G}$ son bases de Gröbner minimales, $\mathrm{LM}(G) = \mathrm{LM}(\tilde{G})$, sean $g \in G$ y $\tilde{g} \in \tilde{G}$, tales que $\mathrm{LM}(g) = \mathrm{LM}(\tilde{g})$, si demostramos que $g = \tilde{g}$, entonces $G = \tilde{G}$.
Consideramos $g - \tilde{g} \in I$ y como G es base de Gröbner, $\overline{g - \tilde{g}}^{G} = 0$ por el Teorema 1.30. Como $\mathrm{LM}(g) = \mathrm{LM}(\tilde{g})$, estos términos se cancelan en $g - \tilde{g}$ y los restantes no son divisibles por ninguno de $\mathrm{LM}(G) = \mathrm{LM}(\tilde{G})$, al ser G y $\tilde{G}$ son bases de Gröbner reducidas, esto significa que $\overline{g - \tilde{g}}^{G} = g - \tilde{g} = 0$, por lo que $g = \tilde{g}$. ■

Ejemplo 1.37. Definimos en `Sagemath`, el anillo de polinomios, en tres variables sobre $\mathbb{Q}$ y con el orden lexicográfico inverso graduado definido anteriormente

en la Definición 1.7, en el primer caso, y un orden lexicográfico definido en la Definición 1.5. A continuación, insertamos el ideal $I = (y^2 - xz, x^2 - z^3)$, y le pedimos que nos de bases de Gröbner reducidas.

```
In : P.<x,y,z> = PolynomialRing(QQ,3,order='degrevlex')
In : I=ideal(y^2-x*z ,x^2 - z^3)
In : I.groebner_basis()
Out: [z^3 - x^2, y^2 - x*z]


In : P.<x,y,z> = PolynomialRing(QQ,3,order='lex')
In : I=ideal(y^2-x*z ,x^2 - z^3)
In : I.groebner_basis()
Out: [x^2 - z^3, x*y^2 - z^4, x*z - y^2, y^4 - z^5]
```

Vemos entonces que según el orden monomial nos dan dos bases de Gröbner diferentes, las dos son reducidas ya que el programa siempre nos proporciona una base de Gröbner reducida. Destacar que $\langle \mathrm{LM}(I)\rangle = \langle z^3, y^2\rangle$ con el primer orden, y $\langle \mathrm{LM}(I)\rangle = \langle x^2, xy^2, xz, y^4\rangle$ con el segundo orden.

1.4.2. Bases de Gröbner de ideales binomiales

En el próximo capítulo le asignaremos a todo código binario un ideal binomial y estudiaremos el código por medio del ideal asociado. Por este motivo esta subsección la dedicaremos al estudio de estos ideales.

Definición 1.38. *Un binomio en $\mathbb{K}[x_1, \ldots, x_n]$ es toda expresión de la forma $x^\alpha - x^\beta$, donde x^α y x^β son monomios. A todo ideal generado por binomios se le denomina ideal binomial.*

Lema 1.39. *Si la entrada del algoritmo de Buchberger descrito en el Teorema 1.31 es un conjunto de binomios, la salida también es un conjunto de binomios.*

Demostración. Sea " $\geq$ " orden monomial, al aplicar el Algoritmo de Buchberger se hacen S-polinomios y restos de divisiones. Para demostrar el resultado veamos primero que si tenemos dos binomios, su S-polinomio es un binomio o cero.

$$S(x^\alpha - x^\beta, x^\gamma - x^\delta) = \frac{x^\nu}{x^\alpha}(x^\alpha - x^\beta) - \frac{x^\nu}{x^\gamma}(x^\gamma - x^\delta) = x^{\nu-\gamma+\delta} - x^{\nu-\alpha+\beta} \quad (1.1)$$

donde $x^\nu = \mathrm{mcm}(x^\alpha, x^\gamma), x^\alpha \geq x^\beta, x^\gamma \geq x^\delta$. Estudiamos ahora el resto de la división de un binomio $x^\alpha - x^\beta$ por un conjunto de binomios $F = (f_1, \ldots, f_s)$. Supondremos también que $x^\alpha \geq x^\beta$. Si $\mathrm{LM}(f_i)$ no divide a x^α para ningún i, tenemos que el resto $r = x^\alpha - \overline{x^\beta}^F$, x^α es un monomio y veamos que $\overline{x^\beta}^F$ es un monomio:

1. Si para todo $i \in 1 \leq i \leq n$, $\mathrm{LM}(f_i)$ no divide a x^β, entonces el resto es $x^\alpha - x^\beta$, un binomio.
2. Si existe $i \in 1 \leq i \leq n$ tal que $\mathrm{LM}(f_i)$ divide a x^β, entonces: $f_i = x^\gamma - x^\nu$ con $\mathrm{LM}(f_i) = x^\gamma$
$$x^\beta - x^{\beta-\gamma}(x^\gamma - x^\nu) = x^{\beta-\gamma+\nu} \tag{1.2}$$
Por lo que $\overline{x^\beta}^F = \overline{x^{\beta-\gamma+\nu}}^F$. Repetimos este proceso hasta obtener el resto, que será un monomio.

En el caso de que $\mathrm{LM}(f_i)$ divida a x^α, tenemos que $f_i = x^\gamma - x^\delta$, y por lo tanto:

$$x^\alpha - x^\beta - x^{\alpha-\gamma}(x^\gamma - x^\delta) = -x^\beta + x^{\alpha-\gamma+\delta} \tag{1.3}$$

por lo tanto, $\overline{f_i}^F = \overline{-x^\beta + x^{\alpha-\gamma+\delta}}^F$ que también es un binomio o es cero. Repitiendo este proceso se deduce que el resto es o bien un binomio o cero. ■

De la prueba de este resultado se extrae también el siguiente Corolario, que utilizaremos en capítulos posteriores.

Corolario 1.40. *Sea $I \subseteq \mathbb{K}[x_1, \ldots, x_n]$ un ideal binomial " $\geq$ " un orden monomial y G una base de Gröbner de I. El resto de la división de un monomio entre I es también un monomio.*

Teorema 1.41. *Sea I un ideal en $\mathbb{K}[x_1, \ldots, x_n]$ y sea " $\geq$ " un orden monomial cualquiera. Si I es un ideal binomial, entonces la base de Gröbner de I reducida respecto de " $\geq$ " está formada por binomios.*

Demostración. Sea I un ideal binomial, entonces $\exists g_1, \ldots, g_t \in \mathbb{K}[x_1, \ldots, x_n]$ binomios, tales que $I = (g_1, \ldots, g_t)$. Por el Lema 1.39 y la construcción de base de Gröbner reducida de la Proposición 1.36, podemos concluir que la base de Gröbner reducida de I está formada por binomios. ■

Ejemplo 1.42. Definimos en `Sagemath`, el anillo de polinomios, en tres variables sobre $\mathbb{Q}$ y consideramos el orden lexicográfico inverso graduado definido anteriormente en 1.7, a continuación, insertamos el ideal $I = (zy^2 - x, x - xyz)$, y le pedimos que nos de una base de Gröbner reducida.

```
In : P.<x,y,z> = PolynomialRing(QQ,3,order='degrevlex')
In : I=ideal(z*y^2-x,x-y*x*z)
In : I.groebner_basis()
Out: [x*y*z - x, y^2*z - x, x^2 - x*y]
```

Como se puede observar, la base de Gröbner reducida de I es $\{xyz - x, y^2z - x, x^2 - xy\}$ y está formada por binomios.

2

Teoría de Códigos

La Teoría de Códigos se encarga de diseñar códigos para garantizar una transmisión de información eficiente y precisa. Se consideran diferentes criterios al diseñar códigos, como la posibilidad de la corrección de errores, la eficiencia de la transmisión y la seguridad de la información.

Un código es un subconjunto $\mathcal{C} \subseteq \mathbb{F}_q^n$. Si $\mathcal{C}$ tiene M elementos, decimos que $\mathcal{C}$ es un $[n, M]$ código. Cada mensaje $m \in \mathbb{F}_q^k$, con $n > k$ lo transformamos en una palabra del código $\mathcal{C}$, mediante una aplicación biyectiva:

$$\begin{aligned} \text{Enc} : \mathbb{F}_q^k &\longrightarrow \mathcal{C} \subseteq \mathbb{F}_q^n \\ m &\longmapsto \text{Enc}(m) = c \end{aligned}$$

A este proceso lo llamamos codificación. La elección del tipo de codificación adecuada depende de los requisitos específicos de la transmisión y almacenamiento de la información. El proceso de decodificación, consiste en tener una palabra $x \in \mathbb{F}_q^n$ y buscar la palabra c del código $\mathcal{C}$ que sea más cercana a ese c dado. Este proceso es más complejo ya que el canal por el que se envía el mensaje puede producir fallos en él, es por ello, que nuestro x es susceptible de venir con fallos. Hay diferentes técnicas de decodificación utilizadas en la Teoría de Códigos, dependiendo del tipo de codificación utilizada y del canal por el que se envía. En este trabajo, vamos a considerar canales sin memoria, que no tengan pérdidas y que sean simétricos y vamos a utilizar la decodificación por máxima verosimilitud, que consiste en comparar la distancia de la palabra dada, con todas las del código según la llamada métrica de Hamming, para así, devolver la palabra más próxima del código.

La información de este capítulo la podemos encontrar en [12].

2.1. Distancia y peso de Hamming

Consideramos el alfabeto $\mathcal{A} = \mathbb{F}_q$, siendo $\mathbb{F}_q$ el cuerpo finito con q elementos (y, por tanto, q es potencia de un número primo).

Definición 2.1. *Para dos vectores $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ se llama distacia de Hamming entre x e y, y se denota por $d_H(x, y)$, al número de posiciones en las que el vector x e y difieren, es decir,*

$$d_H(x, y) = |\{i \in \{1, \ldots, n\} | x_i \neq y_i\}|$$

Proposición 2.2. *La distancia de Hamming es una métrica, es decir, verifica:*

1. $d_H(x, y) \geq 0$ *y* $d_H(x, y) = 0$ *si y solo si* $x = y$.
2. $d_H(x, y) = d_H(y, x)$.
3. $d_H(x, y) \leq d_H(x, z) + d_H(z, y)$ *(desigualdad triangular)*

Demostración. La primera y segunda propiedad son evidentes. Para demostrar la tercera propiedad, definimos:

$$d_H(x, y) = |A_{xy}|,\ d_H(x, z) = |A_{xz}|,\ d_H(z, y) = |A_{zy}| \text{ donde}$$
$$A_{ab} = \{i \in \{1, \ldots, n\} | a_i \neq b_i\}, \forall a, b \in \mathbb{F}_q^n$$

Como conjuntos, $A_{xy} \subseteq A_{xz} \cup A_{zy}$ por lo que, $|A_{xy}| \leq |A_{xz} \cup A_{zy}| = |A_{xz}| + |A_{zy}| - |A_{xz} \cap A_{zy}| \leq |A_{xz}| + |A_{zy}|$ y volviendo a nuestra notación, $d_H(x, y) \leq d_H(x, z) + d_H(z, y)$. ■

Definición 2.3. *Se define el soporte de $x = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$, denotado por* $\text{supp}(x)$ *al conjunto de posiciones de $x \in \mathbb{F}_q^n$ donde hay elementos distintos de cero. Es decir,* $\text{supp}(x) = \{i | x_i \neq 0\}$. *El peso de Hamming de un vector $x \in \mathbb{F}_q^n$ es el número de elementos de su soporte, y se denota por $w_H(x) = |\text{supp}(x)|$.*

Definición 2.4. *La distancia mínima de un código $\mathcal{C}$ se define como:*

$$d_H(\mathcal{C}) = \min_{x,y \in \mathcal{C}, x \neq y} d_H(x, y).$$

El peso mínimo de un código $\mathcal{C}$ se define como:

$$w_H(\mathcal{C}) = \min_{x \in \mathcal{C}, x \neq 0} w_H(x).$$

Como $w_H(x) = d_H(x, 0)$ para todo $x \in \mathbb{F}_q^n$, se tiene que $d_H(\mathcal{C}) \leq w_H(\mathcal{C})$ siempre que $0 \in \mathcal{C}$.

2.2. Códigos lineales

Cuando el alfabeto $\mathcal{A} = \mathbb{F}_q$ es un cuerpo finito, entonces $\mathbb{F}_q^n$ tiene estructura de $\mathbb{F}_q$-espacio vectorial. Por ello, podemos considerar aquellos códigos $\mathcal{C} \subseteq \mathbb{F}_q^n$ que sean subespacios vectoriales y aprovechar esta estructura para facilitar el estudio de sus propiedades así como los procesos de codificación y decodificación.

Definición 2.5. *Un código lineal $\mathcal{C}$ es un subespacio vectorial de $\mathbb{F}_q^n$. La dimensión de $\mathcal{C}$, denotada $k(\mathcal{C})$, es su dimensión como $\mathbb{F}_q$-espacio vectorial. Dado un código lineal $\mathcal{C} \subseteq \mathbb{F}_q^n$ de dimensión $\dim(\mathcal{C}) = k$, decimos que es un $[n, k]$ código. Si además $d_H(\mathcal{C}) = d$, entonces diremos que $\mathcal{C}$ es un $[n, k, d]$ código.*

Lema 2.6. *La distancia mínima de un código lineal coincide con su peso mínimo.*

Demostración. Sea $\mathcal{C}$ un código lineal, como $0 \in \mathcal{C}$ se tiene que $d_H(\mathcal{C}) \leq w_H(\mathcal{C})$. Para demostrar la igualdad basta con observar que si $x \neq y, x, y \in \mathcal{C}$, entonces $x - y \in \mathcal{C} - \{0\}$ y que $d_H(x, y) = w_H(x - y)$. ■

Ejemplo 2.7. Se define el Código de Hamming, $\mathcal{C} \subseteq \mathbb{F}_2^9$ de longitud $n(\mathcal{C}) = 9$ y dimensión $k(\mathcal{C}) = 4$ como la imagen de la aplicación lineal

$$\begin{aligned} \text{Enc} : \mathbb{F}_2^4 &\longrightarrow \mathbb{F}_2^9 \\ m &\longmapsto \text{Enc}(m) = (m, r) \end{aligned}$$

donde a un $m = (m_1, m_2, m_3, m_4) \in \mathbb{F}_2^4$ le añadimos 5 bits redundantes, obteniendo $(m, r) \in \mathbb{F}_2^9$, siendo $r = (r_1, r_2, r_3, r_4, r_5)$ los bits redundantes de información referidos al mensaje m enviado, esos bits se calculan mediante las siguientes ecuaciones: $r_1 = m_1 + m_2, r_2 = m_3 + m_4, r_3 = m_1 + m_4, r_4 = m_2 + m_4, r_5 = m_1 + m_2 + m_3 + m_4$. En la Figura 2.1, podemos observar un diagrama en el que los bits redundantes se pueden ver como bits de paridad en cada una de las filas y columnas de la tabla. Esta redundancia de información tiene, como veremos posteriormente, el propósito de detectar errores y corregirlos.

m_1	m_2	r_1
m_3	m_4	r_2
r_3	r_4	r_5

Figura 2.1. Tabla de paridad del código de Hamming

Calculamos la distancia mínima del código, gracias a la Proposición 2.6 podemos decir que si encontramos el peso mínimo, éste será igual a la distancia mínima. Vemos que

$$\begin{aligned} \mathcal{C} = \{ &(000000000), (000101011), (001100110), (011110101), \\ &(111100000), (001001101), (010010011), (100010101), \\ &(011011110), (100111110), (101011000), (010111000), \\ &(111001011), (101110011), (110101101), (110000110) \ \} \end{aligned}$$

son todos los elementos del código, y podemos observar que el peso mínimo es 4, por lo tanto, la distancia mínima es 4.

2.2.1. Matriz generatriz y de paridad de códigos lineales

Dado un $[n, k]$-código lineal $\mathcal{C}$ y una base de $\mathcal{C}$, se define matriz generatriz o generadora del código, a la matriz que incluye en sus filas los elementos de la base del código.

Definición 2.8. *Sea $\mathcal{C}$ un $[n, k]$-código lineal y sea $B = \{g_1, \ldots, g_k\}$ con $g_i = (g_{i1}, \ldots, g_{in}) \in \mathbb{F}_q^n$, una base de $\mathcal{C}$ como $(\mathbb{F}_q)$-espacio vectorial, se denomina matriz generatriz G de $\mathcal{C}$ a la matriz $k \times n$ cuya fila i-ésima está compuesta por g_i, es decir, $G = (g_{ij}) \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$.*

Observación 2.9. Como en general $\mathcal{C}$ no tiene una única base, tampoco tiene una única matriz generatriz.

Gracias a la matriz generatriz, podemos definir el proceso de codificación de un código lineal como:

$$\begin{aligned} \text{Enc} : \mathbb{F}_q^k &\longrightarrow \mathbb{F}_q^n \\ m &\longmapsto mG = m_1 g_1 + \ldots + m_k g_k \end{aligned}$$

donde $G = \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix} \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$.

Observación 2.10. Enc es una aplicación lineal inyectiva y $\text{Im}(\text{Enc}) = \mathcal{C}$. Por lo que $|\mathcal{C}| = q^k$.

Ejemplo 2.11. Siguiendo el Ejemplo 2.7, del código de Hamming, se observa que la aplicación Enc allí descrita envía:

$$\begin{aligned} 1000 &\longrightarrow 100010101 \\ 0100 &\longrightarrow 010010011 \\ 0010 &\longrightarrow 001001101 \\ 0001 &\longrightarrow 000101011 \end{aligned}$$

Por tanto, una matriz generatriz del código es:

$$G = \left(\begin{array}{cccc|ccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array}\right) \in \mathcal{M}_{4\times 9}(\mathbb{F}_2)$$

Como además, todo subespacio vectorial, se caracteriza mediante sus ecuaciones implícitas. Llamamos matriz de paridad a la matriz de los coeficientes de dichas ecuaciones, esta matriz sirve para indicar si una palabra está o no en el código. Más precisamente:

Definición 2.12. *Definimos matriz de paridad de un $[n,k]$-código $\mathcal{C}$ a una matriz $(n-k)\times n$ de rango $n-k$, tal que $\mathcal{C}$ es el espacio anulador de esa matriz. Es decir, de forma que:*

$$\mathcal{C} = \{x \in \mathbb{F}_q^n : Hx^T = 0\}.$$

Ejemplo 2.13. Siguiendo el Ejemplo 2.7, del código de Hamming, daremos un ejemplo de matriz de paridad del código. Sea $x = (x_1, \ldots, x_9) \in \mathcal{C}$, la tabla de paridad de la Figura 2.1, indica que x debe satisfacer:

$$\begin{cases} x_1 + x_2 + x_5 = 0 \\ x_3 + x_4 + x_6 = 0 \\ x_1 + x_3 + x_7 = 0 \\ x_2 + x_4 + x_8 = 0 \\ x_7 + x_8 + x_9 = 0 \end{cases}$$

Los coeficientes de estas ecuaciones describen una matriz de paridad del código, de tamaño 5×9 y rango 5:

$$H = \begin{pmatrix} 1&1&0&0&1&0&0&0&0 \\ 0&0&1&1&0&1&0&0&0 \\ 1&0&1&0&0&0&1&0&0 \\ 0&1&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&1&1&1 \end{pmatrix}$$

Proposición 2.14. *Sea $\mathcal{C}$ un $[n,k]$ código, con matriz generatriz G de $\mathcal{C}$ y sea H una matriz $(n-k)\times n$ de rango $(n-k)$. Entonces, H es matriz de paridad de $\mathcal{C}$ si y solo si $GH^T = 0$.*

Demostración. " $\Rightarrow$ " Sabemos que si una palabra c está en el código, cumple que $Hc^T = 0$, donde $c = mG$. Si tomamos $m = e_i \in \mathbb{F}_q^k$, donde e_i vector de ceros con un 1 en la posición i, tenemos que $H(mG)^T = 0$, por lo tanto, podemos concluir entonces que $GH^T = 0$.
" $\Leftarrow$ " Tenemos ahora que G es la matriz generatriz de $\mathcal{C}$ y que $GH^T = 0$, veamos que H es la matriz de paridad.
Sea $\mathcal{C}'$ el código con matriz de paridad H. Veamos que $\mathcal{C} \subseteq \mathcal{C}'$. Sea $c \in \mathcal{C}$, tenemos que $c = mG$, con $m \in \mathbb{F}_q^k$, entonces, $Hc^T = HG^T m^T = (mGH^T)^T = 0$, ya que por hipótesis $GH^T = 0$. Por lo tanto, $c \in \mathcal{C}'$. Como $\mathcal{C}$ y $\mathcal{C}'$ tienen la misma dimensión k, concluimos que $\mathcal{C} = \mathcal{C}'$, y que H es matriz de paridad de $\mathcal{C}$. ■

Lo deseable para un código lineal es tener una dimensión k lo mayor posible, para poder mandar un mensaje lo más grande posible, una distancia mínima también grande, y así ser más resistente a errores y por lo tanto más confiable para la transmisión de información, y una n, lo más pequeña posible, para minimizar la longitud de las palabras a enviar por el canal. El siguiente resultado establece una cota que limita los posibles valores que pueden tomar n, k y d.

Corolario 2.15 (Cota de Singleton). *Si $\mathcal{C}$ es un $[n, k, d]$-código lineal, entonces, $d \leq n - k + 1$.*

Demostración. Sea H la matriz de paridad de $\mathcal{C}$ y veamos que cualesquiera $d-1$ columnas de H son independientes. Supongamos por reducción al absurdo que existen $d-1$ columnas dependientes en H, entonces, existirá $c \in \mathbb{F}_q^n$ con $0 < |\text{supp}(c)| \leq d-1$ tal que $Hc^T = 0$. Significa entonces que $c \in \mathcal{C}$ y que $w_H(c) = d - 1 < d$. Esto es absurdo porque d es la distancia mínima. Por lo tanto, el rango de H es $\geq d-1$ y, por la definición de matriz de paridad se tiene que rango(H)=$n-k$, por lo que $d - 1 \leq n - k$, entonces, despejando, $d \leq n - k + 1$. ■

En el Ejemplo 2.7, destacar que la dimensión del código de Hamming $k(\mathcal{C})$ es 4, la longitud $n(\mathcal{C})$ es 9 y la distancia mínima $d_H(\mathcal{C})$ es 4, se observa que, en efecto, se cumple la cota de Singleton, es decir, $4 \leq 9 - 4 + 1 = 6$.

2.3. Decodificación y capacidad correctora de un código

La decodificación de un código $\mathcal{C}$ trata de recuperar el mensaje dado con errores, debido a que éste ha pasado por un canal con perturbaciones, obteniendo finalmente una palabra del código a la menor distancia posible de la palabra recibida. Este proceso es bastante complejo en general. La capacidad correctora, por otro lado, se refiere a la capacidad de un código para corregir errores, es decir, que si una palabra del código está perturbada en no más posiciones de lo que indica la capacidad correctora, entonces habrá una única palabra del código más próxima a la recibida.

Definición 2.16. *Un decodificador completo de un código $\mathcal{C}$ por mínima distancia es una aplicación:*

$$\text{Dec} : \mathbb{F}_q^n \longrightarrow \mathcal{C}$$

que cumple que si $Dec(x) = c$, entonces $d_H(x, c') \geq d_H(x, c)$ para todo $c' \in \mathcal{C}$.

Observación 2.17. Se observa que si $c_1, \ldots, c_s \in \mathcal{C}$ son todas las palabras que minimizan la distancia a un $x \in \mathbb{F}_q^n$, entonces $\text{Dec}(x) \in \{c_1, \ldots, c_s\}$. Veremos que cuando hay una palabra $c \in \mathcal{C}$ lo suficientemente cercana a x, esta será la única que minimice la distancia entre x y todas las palabras de $\mathcal{C}$.

Definición 2.18. *Diremos que un código $\mathcal{C}$ es t-corrector si para todo $c_1, c_2 \in \mathcal{C}$ dos palabras distintas de $\mathcal{C}$ y para todo $e_1, e_2 \in \mathbb{F}_q^n$, con $w_H(e_i) \leq t$ se tiene que $c_1 + e_1 \neq c_2 + e_2$. Esto quiere decir que si modificamos una palabra del código $\mathcal{C}$ en a lo sumo t posiciones, entonces c es la única palabra del código a distancia menor o igual a t de la palabra modificada.*

Se define la capacidad correctora de un código $\mathcal{C}$ como el mayor valor t tal que $\mathcal{C}$ es t-corrector, y se denota por $t_{\mathcal{C}}$.

Proposición 2.19. *Sea un $[n, k, d]$-código $\mathcal{C}$. Entonces la capacidad correctora de $\mathcal{C}$ es $t_{\mathcal{C}} = \lfloor \frac{d-1}{2} \rfloor$.*

Demostración. Veamos primero que $t_{\mathcal{C}} \geq \lfloor \frac{d-1}{2} \rfloor$ o, equivalentemente, que $\mathcal{C}$ es $\lfloor \frac{d-1}{2} \rfloor$-corrector. Sean $c_1, c_2 \in \mathcal{C}$ y sean $e_1, e_2 \in \mathbb{F}_q^n$ con $w_H(e_i) \leq \lfloor \frac{d-1}{2} \rfloor$ tales que $c_1 + e_1 = c_2 + e_2$. Veamos que $c_1 = c_2$. En efecto, $c_1 - c_2 \in \mathcal{C}$ y $d_H(c_1, c_2) = w_H(c_1 - c_2) = w_H(e_1 - e_2) \leq w_H(e_1) + w_H(e_2) \leq 2 \cdot \lfloor \frac{d-1}{2} \rfloor \leq d - 1 < d$ y, de la definición de distancia mínima se sigue que $c_1 = c_2$.

Veamos ahora que $\mathcal{C}$ no es $\lfloor \frac{d-1}{2} \rfloor + 1$ corrector. Sea $c \in \mathcal{C}, c \neq 0$ con $w_H(c) = d$. Entonces podemos escribir $c = e_1 + e_2$ con $w_H(e_1) = \lfloor \frac{d}{2} \rfloor$ y $w_H(e_2) = \lceil \frac{d}{2} \rceil$ se observa pues que $0, c \in \mathcal{C}$ y que $0 + e_1 = c - e_2$ y $w_H(e_i) \leq \lceil \frac{d}{2} \rceil = \lfloor \frac{d-1}{2} \rfloor + 1$. Por tanto, $\mathcal{C}$ no es $\lfloor \frac{d-1}{2} \rfloor + 1$ corrector. ■

En la Definición 2.16 se observa que dado $x \in \mathbb{F}_q^n$, si existe $c \in \mathcal{C}$ con $d_H(x, c) \leq \lfloor \frac{d-1}{2} \rfloor$ esta será la única palabra del código a distancia $\leq \lfloor \frac{d-1}{2} \rfloor$ de x y, por tanto, $\mathrm{Dec}(x) = c$ en todo decodificador total.

2.3.1. Decodificación por fuerza bruta

La decodificación por fuerza bruta es un algoritmo de decodificación que podemos usar para todo código, sea lineal o no, y en el que se recibe $x \in \mathbb{F}_q^n$ y se comprueba de forma exhaustiva cuál es la palabra en $\mathcal{C}$ más próxima a x.

Proposición 2.20. *Todo código tiene un decodificador por mínima distancia.*

Demostración. Para esta demostración, justificaremos el algoritmo descrito en la Figura 2.2, en el que como entrada, recibe una palabra $x \in \mathbb{F}_q^n$ y la matriz de paridad H del código. Haciendo uso de H, se comprueba si $Hx^T = 0$ o, equivalentemente, si $x \in \mathcal{C}$. Si $x \in \mathcal{C}$ se devuelve x. En caso contrario, se procede a verificar si hay una palabra $c \in \mathcal{C}$ con $d_H(x, c) = 1$. Para ello, se construyen todos los vectores $e \in \mathbb{F}_q^n$ con $|\mathrm{supp}(e)| = 1$ y se comprueba si $x - e \in \mathcal{C}$ para cada uno de ellos. Esto es equivalente a comprobar que $H(x - e)^T = 0$ o, lo que es lo mismo, que $Hx^T = He^T$. Si existe un $e \in \mathbb{F}_q^n$ con $|\mathrm{supp}(e)| = 1$ tal que $x - e \in \mathcal{C}$, entonces se devuelve $x - e$. En caso contrario se prosigue con los vectores $e \in \mathbb{F}_q^n$ con $|\mathrm{supp}(e)| = 2$ y así sucesivamente. Este algoritmo devuelve siempre una de las palabras más próximas a x. Esto se consigue gracias a que se toman los valores de $|\mathrm{supp}(e)|$ en orden creciente. ■

Algoritmo de decodificación por fuerza bruta

Entrada: $x \in \mathbb{F}_q^k$, H matriz de Paridad de $\mathcal{C}$
Salida: $c \in \mathcal{C}$ tal que $d_H(x,c) \leq d_H(x,c')$ para cualquier $c' \in \mathcal{C}$
for $i = 0$ to n **do**
  for all $e \in \mathbb{F}_q^n$, con $|w_H(e)| = i$ **do**
    if $Hx^T = He^T$ **then**
      return $x - e$
    end if
  end for
end for

Figura 2.2. Algoritmo de decodificación por fuerza bruta

Observación 2.21. Este algoritmo, en general no es eficiente, ya que para un valor i fijado, se tiene que el conjunto $\{e \in \mathbb{F}_q^n / w_H(e) = i\}$ tiene $\binom{n}{i}(q-1)^i$ elementos. Por esto, el número de pasos del algoritmo será como cota superior $\sum_{i=0}^{n} \binom{n}{i}(q-1)^i = q^n$, que es de una complejidad de orden $O(q^n)$ y es inviable en la práctica para valores de q y n grandes.

2.3.2. Decodificación de códigos binarios por descenso de gradiente

Nos centraremos ahora en un algoritmo de decodificacion, llamado decodificación por descenso de gradiente, que sirve para códigos binarios. Este algoritmo requiere del conocimiento previo de todas las palabras minimales del codigo. Remitimos al lector a las referencias [1] y [2] que hablan de forma más desarrollada de la decodificación por descenso de gradiente.

Definición 2.22. *Sea $\mathcal{C} \subseteq \mathbb{F}_2^n$ un código binario. Se dice que una palabra $c \in \mathcal{C}$, $c \neq 0$ es minimal si no existe $c' \in \mathcal{C}$ distinta a cero, tal que* $\mathrm{supp}(c') \subseteq \mathrm{supp}(c)$.

Se define $M \subseteq \mathcal{C}$ como el conjunto de palabras minimales de $\mathcal{C}$.

En este trabajo, no abordaremos el problema de calcular todas las palabras minimales del codigo, pues esto se puede hacer mediante el procedimiento descrito en [9].

Lema 2.23. *Sea $\mathcal{C} \subseteq \mathbb{F}_2^n$ un código binario y sea $x \in \mathcal{C}$, con $x \neq 0$, entonces $\exists m_1, \ldots, m_k \in M$ de soporte disjunto tales que $x = m_1 + \ldots + m_k$.*

Demostración. Procederemos por inducción sobre los elementos del soporte $|\mathrm{supp}(x)|$, cuándo vale uno, es fácil ver que $x \in M$, supongamos cierto para $|\mathrm{supp}(x)| \leq r$ y vamos a demostrarlo para $|\mathrm{supp}(x)| \leq r+1$. Si $x \in M$, lo tendríamos directamente. En el caso de que x no esté en M, tendrá que existir un $m \in M$, tal que el soporte de m esté estrictamente contenido en el de x, y

por lo tanto, tomando $x' = x+m$ se satisface que $\text{supp}(x') = \text{supp}(x) - \text{supp}(m)$ (por ser $\mathcal{C}$ binario), y por tanto $|\text{supp}(x')| < |\text{supp}(x)|$. Por inducción se sigue el resultado. ■

Proposición 2.24. *Sean $\mathcal{C} \subseteq \mathbb{F}_2^n$ un código binario y $x \in \mathbb{F}_2^n$. Entonces, se cumple una de estas dos condiciones:*

1. *Existe $m \in M$ tal que $w_H(x+m) < w_H(x)$, o*
2. *$d(x,0) \leq d(x,c)$ para todo $c \in \mathcal{C}$.*

Demostración. Supongamos que (1) no se cumple, entonces, $w_H(x+m) \geq w_H(x)$ para cualquier $m \in M$. Sea ahora c un palabra del código y sea $c = m_1 + \ldots + m_l$ con $m_1 + \ldots + m_l \in M$ de soporte disjunto.

Veamos que $w_H(x) \leq w_H(x+c)$:
Seguiremos la demostración apoyándonos en la Figura 2.3. Definimos dos conjuntos $I = \text{supp}(x)$ y $J_i = \text{supp}(m_i)$ $\forall i \in \{1, \ldots, l\}$ con $J_{i1} = J_i \cap I$ y $J_{i2} = J_i - J_{i1}$. Como $|\text{supp}(x+m_i)| \geq |\text{supp}(x)|$, entonces $|J_{i2}| \geq |J_{i1}|$. De aquí se deduce que

$$\begin{aligned} |\text{supp}(x+c)| = w_H(x+c) = \\ = |I| - |J_{11}| - \ldots - |J_{k1}| + |J_{12}| + \ldots + |J_{k2}| \geq \\ \geq |I| = |\text{supp}(x)| = w_H(x). \end{aligned}$$

Y, por tanto, $d_H(x,0) \leq d(x,c)$ $\forall c \in \mathcal{C}$. Así quedaría demostrado que si (1) no se cumple, estamos en la condición (2) y se sigue el resultado. ■

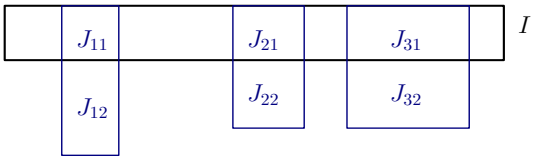


Figura 2.3. Ilustración de los soportes de los vectores de la Proposición 2.24

Esta proposición sugiere un decodificador, llamado decodificador por descenso de gradiente, de un código binario que utiliza el conjunto de palabras minimales de $\mathcal{C}$. Este algoritmo esté descrito en la Figura 2.4.

Teniendo en cuenta que si $x, y \in \mathbb{F}_2^n$, entonces

$$\text{supp}(x+y) = \text{supp}(x) \triangle \text{supp}(y)$$

donde $\triangle$ denota la diferencia simétrica, este algoritmo se puede reescribir en términos del soporte de las palabras de M y de x como indica la Figura 2.5.

Algoritmo de decodificación por descenso de gradiente

Entrada: $M \subseteq \mathbb{F}_2^n$ conjunto de palabras minimales de $\mathcal{C}$ y $x \in \mathbb{F}_2^n$
Salida: una de las palabras $c \in \mathcal{C}$ más próximas a x.
$c = 0$
while $\exists\ m \in M$, tal que $w_H(x+m) < w_H(x)$ **do**
  $c = c + m$
  $x = x + m$
end while
return c

Figura 2.4. Algoritmo de decodificación por descenso de gradiente

Algoritmo de decodificación por descenso de gradiente

Entrada: $M \subseteq \mathbb{F}_2^n$ conjunto de palabras minimales de $\mathcal{C}$ y $x \in \mathbb{F}_2^n$
Salida: $K = \mathrm{supp}(c)$, siendo $c \in \mathcal{C}$ una de las palabras más próximas a x.
$I = \mathrm{supp}(x)$
$K = \emptyset$
while $\exists\ m \in M$, tal que $|I \triangle \mathrm{supp}(m)| < |I|$ **do**
  $K = K \cup \mathrm{supp}(m)$
  $I = I \triangle \mathrm{supp}(m)$
end while
return K

Figura 2.5. Algoritmo de decodificación por descenso de gradiente

Ejemplo 2.25. Sea $\mathcal{C}$ el código binario $[6,3]$-código, donde

$$\mathcal{C} = \{(000000), (000111), (011010), (011101),$$
$$(100001), (100110), (111011), (111100), (111111)\}$$

son todos los elementos del código, su matriz generatriz es

$$G = \begin{pmatrix} 1\,0\,0\,0\,0\,1 \\ 0\,1\,1\,0\,1\,0 \\ 0\,0\,0\,1\,1\,1 \end{pmatrix}$$

las palabras de soporte minimal son

$$M = \left\{ \begin{array}{l} m_1 = (100001), m_2 = (100110), m_3 = (011010), \\ m_4 = (000111), m_5 = (111100),\ m_6 = (011101) \end{array} \right\}$$

viendo en el primer algoritmo descrito en Figura 2.4, dada una palabra $x = (100101) \in \mathbb{F}_2^6$ se tiene que $w_H(x) = 3$. Buscamos en el conjunto de palabras

minimales un $m \in M$ tal que $w_H(x+m) < 3$. Si tomamos m_1 se tiene que $w_H(x+m_1) = 1$. Como $w_H(x+m_1+m_i) > 1 \ \forall i \in \{1,\ldots,6\}$, acabaríamos, devolviendo $c = m_1$. Sin embargo, si cogemos primero m_2 se tiene que $w_H(x+m_2) = 2$, y tendríamos $x + m_2 = (000011)$. Intentamos encontrar otra palabra minimal que sumada a $x + m_2$, nos dé peso menor que 2. Ahora encontramos que $x + m_2 + m_4 = (000100)$ tiene peso $w_H(x + m_2 + m_4) = 1$. Por lo que, el algoritmo devuelve $c = m_2 + m_4 = m_1$. Se observa que en este ejemplo, en ambos casos se devuelve m_1. Esto se debe a que m_1 es la única palabra en $\mathcal{C}$ a distancia 1 de x (aunque la $d_H(\mathcal{C}) = 2$ y la capacidad correctora es 0).

Para el segundo algoritmo, con la misma palabra, tenemos $I = \text{supp}(x) = \{1,4,6\}$ y además, que $|I| = 3$. Hallemos los soportes de las palabras minimales, donde $\text{supp}(m_i) = J_i$, serán:

$$J_1 = \{1,6\}, \quad J_2 = \{1,4,5\}, \quad J_3 = \{2,3,5\},$$
$$J_4 = \{4,5,6\}, J_5 = \{1,2,3,4\}, J_6 = \{2,3,4,6\}.$$

Comprobamos ahora los soportes de las palabras minimales que cumplen, $|I \triangle J_i| < |I|$. En este caso, tomando J_1 como primera opción, obtenemos $I \triangle J_1 = \{4\}$ donde $|I \triangle J_1| = 1$ y, siendo menor que $|I|$. Ahora observamos que $|(I \triangle J_1)\ \textit{triangle}\ J_i| > |I \triangle J_1|$, para todo $i \in \{1,\ldots,6\}$. Por lo tanto, el algoritmo devuelve $K = J_1 = \{1,6\}$. Como en el caso del algoritmo anterior, tendremos la segunda opción de coger primero $(I \triangle J_2) = \{5,6\}$, con $|I \triangle J_2| = 2$, siendo menor que $|I|$. Intentamos encontrar otro soporte de las palabras minimales donde podamos encontrar que el número de elementos obtenidos sea menor que 2. Encontramos que $(I \triangle J_2) \triangle J_4 = \{4\}$, siendo así $|(I \triangle J_2) \triangle J_4| = 1$ y por lo tanto, el número de elementos es menor que 2. Por lo que, el algoritmo devolverá $K = \{1,6\}$. Dando así, el mismo resultado en cada caso.

3

Bases de Gröbner y Códigos Binarios

A todo código binario $\mathcal{C} \subseteq \mathbb{F}_2^n$, le vamos a asociar un ideal $I(\mathcal{C})$ del anillo de polinomios en n variables y aplicaremos técnicas basadas en bases de Gröbner (desarrolladas en el Capítulo 1) para dar las soluciones algorítmicas a los dos grandes problemas: el problema de calcular la distancia mínima (Teorema 3.7) y el problema de la decodificación (Algoritmos 3.1 y 3.2). En particular, propondremos dos algoritmos de decodificación basados en bases de Gröbner y compararemos estos algoritmos con los propuestos en los capítulos anteriores. Este capítulo se divide en dos secciones. En la primera definiremos el ideal asociado a un código binario y estudiaremos la estructura de la base de Gröbner reducida de este ideal respecto de un orden graduado. En la segunda sección discutiremos cómo usar esta base de Gröbner para el cálculo de la distancia mínima y para la decodificación.

3.1. El ideal asociado a un código binario

Definición 3.1. *Sea $\mathcal{C} \subseteq \mathbb{F}_2^n$ un código binario se define el ideal del código $\mathcal{C}$ como el ideal binomial:*

$$I(\mathcal{C}) = \langle \{x^a - x^b \mid a, b \in \mathbb{F}_2^n, a + b \in \mathcal{C}\} \rangle + \langle \{x_i^2 - 1 \mid 1 \leq i \leq n\} \rangle.$$

Donde para $a = (a_1, \ldots, a_n) \in \mathbb{F}_2^n$ denotamos $x^a := \prod_{a_i=1} x_i$.

Lema 3.2. *Sean $a, b \in \mathbb{F}_2^n$ y $c = a + b \in \mathbb{F}_2^n$. Sea $J \subseteq \mathbb{K}[x_1, \ldots, x_n]$ un ideal tal que $\langle x_1^2 - 1, \ldots, x_n^2 - 1 \rangle \subseteq J$. Entonces,*

$$[x^a] \cdot [x^b] = [x^c],$$

donde $[g]$ denota la clase de $g \in \mathbb{K}[x_1, \ldots, x_n]$ en $\mathbb{K}[x_1, \ldots, x_n]/J$.

Demostración. Si $a = (a_1, \ldots, a_n) \in \mathbb{F}_2^n$, $b = (b_1, \ldots, b_n) \in \mathbb{F}_2^n$ y definimos $a + b = c = (c_1, \ldots, c_n) \in \mathbb{F}_2^n$. Para demostrar el resultado veamos que $[x_i^{a_i}] \cdot [x_i^{b_i}] = [x_i^{c_i}]$ $\forall i \in \{1, \ldots, n\}$. Si $a_i = 1, b_i = 1$, entonces $c_i = 0$ y $x_i^{a_i} \cdot x_i^{b_i} = x_i^2$.

Como $x_i^2 - 1 \in J$, tomando clases de equivalencia en el anillo cociente, tenemos $[x_i^{a_i} \cdot x_i^{b_i}] = [x_i^2] = [1] = [x_i^{c_i}]$.
Para los demás casos, es decir, si $a_i \neq 1$ o $b_i \neq 1$, lo tenemos directamente ya que $x_i^{a_i} \cdot x_i^{b_i} = x_i^{c_i}$. ■

Lema 3.3. *Sea $J \subseteq \mathbb{K}[x_1, \ldots, x_n]$ un ideal tal que $\langle x_1^2 - 1, \ldots, x_n^2 - 1\rangle \subseteq J$. Sea $a, b \in \mathbb{F}_2^n$ y $a + b \in \mathcal{C}$, $x^a - x^b \in J$ si y solo si $x^{a+b} - 1 \in J$.*

Demostración. " $\Rightarrow$ " Vemos que si $x^a - x^b \in J$, considerando el anillo cociente $\mathbb{K}[x_1, \ldots, x_n]/J$ como en el Lema 3.2, tenemos que:
Como $[0] = [x^a - x^b]$, entonces $[0] = [x^b][0] = [x^b(x^a - x^b)] = [x^{a+b} - x^{2b}] = [x^{a+b} - 1]$.
" $\Leftarrow$ " Si $[x^{a+b} - 1] = [1]$, entonces $[x^b][x^{a+b}] = [x^b]$ y se tiene el resultado $[x^{a+2b}] = [x^a]$, concluimos que $[x^a] = [x^b]$. ■

De la Definición 3.1 se podría interpretar que hace falta conocer todas las palabras de $\mathcal{C}$ para definir $I(\mathcal{C})$. Sin embargo, como veremos en el siguiente resultado, basta con conocer una base del subespacio $\mathcal{C}$ o, en otras palabras, una matriz generatriz de $\mathcal{C}$.

Proposición 3.4. *Si $\mathcal{C}$ es un código binario con matriz generatriz $G = (g_{ij}) \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$, entonces $I(\mathcal{C}) = \langle\{x^{g_j} - 1 \mid 1 \leq j \leq k\}\rangle + \langle\{x_i^2 - 1 \mid 1 \leq i \leq n\}\rangle$.*

Demostración. A lo largo de la demostración, denotamos por $J = \langle\{x^{g_j} - 1 \mid 1 \leq j \leq k\}\rangle + \langle\{x_i^2 - 1 \mid 1 \leq i \leq n\}\rangle$ para todo $j \in \{1, \ldots, k\}$.
" $\supseteq$ " Como $g_j \in \mathcal{C}$, entonces los $x^{g_j} - x^0 = x^{g_j} - 1 \in I(\mathcal{C})$.
" $\subseteq$ " Sean $a, b \in \mathbb{F}_2^n$ tales que $a + b \in \mathcal{C}$ están en el código. Veamos que $x^a - x^b \in J$. Por el Lema 3.3 tenemos que $x^{a+b} - 1 \in J$.
Como $a + b \in \mathcal{C}$ y G es la matriz generatriz, entonces $\exists\, i_1, \ldots, i_r \in \{1, \ldots, k\}$ tales que $a + b = g_{i_1} + \ldots + g_{i_r}$ y suponemos sin pérdida de generalidad que $a + b = g_{i_1} + \ldots + g_{i_r}$. Procedemos a demostrarlo por inducción sobre r. Para $r = 1$, $x^{a+b} - 1 = x^{g_1} - 1 \in J$, por lo tanto, aceptamos que $a + b = g_1 + \ldots + g_r$. Demostramos entonces, cierto para $r + 1$. Tenemos $a + b = g_{i_1} + \ldots + g_{r+1}$, y sea $[x^{a+b} - 1] = [x^{a+b} - x^{g_1+\ldots+g_r} - 1] + [x^{g_1+\ldots+g_r} - 1]$ donde, por hipótesis, sabemos que $[x^{g_1+\ldots+g_r} - 1] \in J$, por lo tanto, $[x^{g_1+\ldots+g_r}] \cdot [x^{g_{r+1}} - 1] = [0]$, concluyendo así que $x^{a+b} - 1 \in J$. ■

Consideremos " $\geq$ " un orden monomial graduado y sea $G = \{g_1, \ldots, g_t\}$ la base de Gröbner reducida de $I(\mathcal{C})$ respecto de " $\geq$ ". En el Teorema 1.41, vimos que si I es un ideal binomial, entonces su base de Gröbner reducida respecto de " $\geq$ " está formada por binomios. Por lo tanto, $g_1, \ldots, g_t$ son binomios.

Vamos a estudiar la estructura de la base de Gröbner del ideal $I(\mathcal{C})$. A partir de ahora denotaremos por G a la base de Gröbner reducida de $I(\mathcal{C})$ respecto de un orden monomial graduado prefijado.

Lema 3.5. *Si $g \in G$, entonces:*

- $g = x_i^2 - 1$, *o*
- $g = x_1^{\alpha_1} \cdots x_n^{\alpha_n} - x_1^{\beta_1} \cdots x_n^{\beta_n}$ *con* $\alpha_i, \beta_i \in \{0,1\}$ *y* $\alpha_i \cdot \beta_i = 0$ $\forall i \in \{1, \ldots, n\}$

Demostración. Sea $g = x_1^{\alpha_1} \cdots x_n^{\alpha_n} - x_1^{\beta_1} \cdots x_n^{\beta_n} \in G$. Veamos primero que $\beta_i \leq 1$ $\forall i \in \{1, \ldots, n\}$.
Por reducción al absurdo, suponemos que $\exists i \in \{1, \ldots, n\}$ con $\beta_i \geq 2$. Como G es base de Gröbner reducida y $x_i^2 - 1 \in I(\mathcal{C})$, entonces $x_i^2 \in \langle \mathrm{LM}(I(\mathcal{C}))\rangle$ y por lo tanto, $x_1^{\beta_1} \cdots x_n^{\beta_n} \in \langle \mathrm{LM}(I(\mathcal{C}))\rangle$. Esto es absurdo, porque contradice que G es la base de Gröbner reducida.
Supongamos ahora que $\exists i \in \{1, \ldots, n\}$ tal que $\alpha_i \geq 2$. Como G es reducida y $x_i^2 - 1 \in I(\mathcal{C})$,entonces $x_i^2 \in \langle \mathrm{LM}(I(\mathcal{C}))\rangle$ como en el caso anterior. Esto implica que $g = x_i^2 - x_1^{\beta_1} \cdots x_n^{\beta_n}$ lo que significa que $g_j - (x_i^2 - 1) \in I(\mathcal{C})$. Si $x_1^{\beta_1} \cdots x_n^{\beta_n} - 1 \neq 0$, entonces $x_1^{\beta_1} \cdots x_n^{\beta_n} - 1 \in I(\mathcal{C})$ y, en consecuencia $x_1^{\beta_1}, \ldots, x_n^{\beta_n} \in \langle \mathrm{LM}(I(\mathcal{C}))\rangle$. De nuevo, esto es imposible porque G es la base de Gröbner reducida. Por tanto, $x_1^{\beta_1} \cdots x_n^{\beta_n} = 1$ y $g = x_i^2 - 1$. Ya solo falta considerar el caso en que $\alpha_i, \beta_i \leq 1$ $\forall i \in \{1, \ldots, n\}$. Supongamos ahora que $\exists j \in \{1, \ldots, n\}$ tal que $\alpha_j \cdot \beta_j \neq 0$ es decir, $\alpha_j, \beta_j = 1$, podemos escribir $g = x_j \cdot h$, veamos que $h \in I(\mathcal{C})$. Como $x_j \cdot g = x_j^2 \cdot h \in I(\mathcal{C})$ y $(x_j^2 - 1) \cdot h \in I(\mathcal{C})$, entonces decimos que $h = x_j^2 \cdot h - (x_j^2 - 1)h \in I(\mathcal{C})$.
Además, $\mathrm{LM}(h) = \frac{\mathrm{LM}(g)}{x_j}$, lo cual es absurdo porque G es base de Gröbner reducida y en particular minimal. Por lo tanto, $\alpha_j \cdot \beta_j = 0$. ■

3.2. Distancia mínima y decodificación de códigos binarios mediante bases de Gröbner

En el Teorema 3.7, vamos a ver cómo resolver uno de los grandes problemas que abordamos en este trabajo, calcular la distancia mínima de un código binario haciendo uso de las bases de Gröbner. En su demostración usaremos el siguiente lema.

Lema 3.6. *Si $a, b \in \mathbb{F}_2^n$, $a + b \in \mathcal{C}$ si y solo si $x^a - x^b \in I(\mathcal{C})$.*

Demostración. " $\Rightarrow$ " Por definición.
" $\Leftarrow$ " Por la Proposición 3.4, si $G = (g_{ij}) \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$ es una matriz generatriz de $I(\mathcal{C})$, entonces

$$I(\mathcal{C}) = \langle \{x^{g_j} - 1 \mid 1 \leq j \leq k\}\rangle + \langle \{x_i^2 - 1 \mid 1 \leq i \leq n\}\rangle.$$

Sea $x^c - x^d \in I(\mathcal{C})$, con $c = (c_1, \ldots, c_n), d = (d_1, \ldots, d_n) \in \mathbb{N}^n$, veamos que $(c_1 + d_1 \bmod 2, \ldots, c_n + d_n \bmod 2) \in \mathcal{C}$. Para ello basta observar que, como $g_i \in \mathcal{C}$, se tiene que para todo $\gamma = (\gamma_1, \ldots, \gamma_n) \in \mathbb{N}^n$ se tiene que $x^\gamma(x^{g_i} - 1) = x^{\gamma + g_i} - x^\gamma$ y $(2\gamma_1 + g_{i_1} \bmod 2, \ldots, 2\gamma_n + g_{i_n} \bmod 2) = (g_{i_1}, \ldots, g_{i_n}) \in \mathcal{C}$. ■

Teorema 3.7. *Sea* $G = \{x^{a_i} - x^{b_i} \mid 1 \leq i \leq s\} \cup \{x_i^2 - 1 \mid j \in J\}$ *la base de Gröbner reducida de* $I(\mathcal{C})$ *respecto de un orden graduado* " $\geq$ " *donde* $a_i, b_i \in \mathbb{F}_2^n$, *para todo* $i \in \{1, \ldots, s\}$ *y* $J \subseteq \{1, \ldots, n\}$. *Entonces:*

$$d_H(\mathcal{C}) = \min\{w_H(a_i + b_i) \mid 1 \leq i \leq s\}.$$

Demostración. " $\leq$ " Se tiene por el Lema 3.6 ya que $a_i + b_i \in \mathcal{C}\ \forall i \in \{1, \ldots, s\}$. " $\geq$ " Sea $c \in \mathcal{C} - \{0\}$ tal que $x^c \leq x^d$ para todo $d \in \mathcal{C} - \{0\}$. Como " $\geq$ " es graduado se tiene en particular que $w_H(c) = d_H(\mathcal{C})$. Sean $a, b \in \mathbb{F}_2^n$ cumpliendo:

- $w_H(a) - 1 \leq w_H(b) \leq w_H(a)$
- $x^a \geq x^b$
- $a_i \cdot b_i = 0, \forall i \in \{1, \ldots n\}$
- $a + b = c$

Vamos a demostrar que $x^a - x^b \in G$ y, por consiguiente $d_H(\mathcal{C}) = w_H(c) = w_H(a + b)$. Para ello veamos que (a) x^a es generador minimal de $\langle \mathrm{LM}(I(\mathcal{C}))\rangle$, y (b) $x^b \notin \langle \mathrm{LM}(I(\mathcal{C}))\rangle$.

Si no se cumple (a), entonces $\exists x^{a'} - x^{b'} \in G$ tal que $x^{a'} \geq x^{b'}, x^{a'} \neq x^a$ y $x^{a'}$ divide a x^a. En este caso, $w_H(b') \leq w_H(a') < w_H(a)$ ya que " $\geq$ " es graduado. Por esto, se tiene que:

$$\begin{aligned} w_H(a' + b') &\leq w_H(a') + w_H(b') \leq \\ &\leq 2 \cdot w_H(a') \leq \\ &\leq 2 \cdot w_H(a) - 2 < w_H(a) + w_H(b) = \\ &= w_H(a + b) = d_H(\mathcal{C}) \end{aligned}$$

Lo cual es un absurdo ya que por el Lema 3.6, $a' + b' \in \mathcal{C}$ y $w_H(a' + b') < d_H(\mathcal{C})$. Si (b) no se cumple, tenemos que $x^b \in \langle \mathrm{LM}(I(\mathcal{C}))\rangle$, por lo que $\exists\ x^{b'} - x^{a'} \in G$ tal que $x^{b'} \geq x^{a'}$ y $x^{b'}$ divide a x^b. Consideramos: $x^a - x^{b-b'+a'} \in I(\mathcal{C})$, entonces:

$$\begin{aligned} w_H(a + b - b' + a') &\leq w_H(a) + w_H(b - b' + a') \leq \\ &\leq w_H(a) + w_H(b) - w_H(b') + w_H(a') \leq \\ &\leq w_H(a) + w_H(b) = \\ &= w_H(a + b) = d_H(\mathcal{C}) \end{aligned}$$

Como $d_H(\mathcal{C})$ es la distancia mínima, $w_H(a + b - b' + a') = d_H(\mathcal{C})$. Veamos que $x^a \cdot x^{b-b'+a'} < x^a \cdot x^b$; en efecto, $x^{b'} > x^{a'}$ entonces, $x^{a+b+b'} > x^{a+b+a'}$ y por lo tanto, $x^{a+b} > x^{a+b-b'+a'}$. No obstante, esto es un absurdo porque $a + b = c_1$ y $x^{c_1} < x^d\ \forall d \in \mathcal{C} - \{0\}$. ■

El siguiente Teorema resuelve el problema de decodificación.

Teorema 3.8. *Sean* $w \in \mathbb{F}_2^n$, G *la base de Gröbner reducida de* $I(\mathcal{C})$ *respecto de un orden graduado* " $\geq$ " *y* x^e *el resto de la división de* x^w *entre* G. *Entonces:*

1. $w + e \in \mathcal{C}$

2. Si $c \in \mathcal{C}$, entonces $d_H(w,c) \geq d_H(w, w+e)$

Es decir, $w+e$ es una de las palabras de $\mathcal{C}$ más próximas a w.

Demostración. Primero observamos que, por el Corolario 1.40, el resto de la división de x^w entre G es, en efecto, un monomio. Demostramos (1) a partir de que $\overline{x^w}^G = x^e$, entonces $x^w - x^e \in I(\mathcal{C})$ y utilizando el Lema 3.6 concluimos que $w+e \in \mathcal{C}$.
Para demostrar (2), tenemos c en el código, esto implica que $x^w - x^{w+c} \in I(\mathcal{C})$ y que $x^w - x^e \in I(\mathcal{C})$. Restando los dos anteriores tenemos que $h = x^{w+c} - x^e \in I(\mathcal{C})$. Si $h = 0$, entonces $w + c = e$. En caso contrario, $x^{w+c} > x^e$ ya que x^e no pertenece a $\langle \mathrm{LM}(I(\mathcal{C}))\rangle$ al ser el resto de una división (Proposición 1.27). Tendremos entonces que $x^{w+c} \geq x^e$ y como " $\geq$ " es graduado, $d_H(w,c) = w_H(w+c) \geq w_H(e) = d_H(w, w+e)$. ■

Este teorema sugiere y justifica el algoritmo de decodificacion descrito en la Figura 3.1.

Algoritmo de decodificación por bases de Gröbner I

Entrada: $G' \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$ matriz generatriz de $\mathcal{C}$ e $w \in \mathbb{F}_2^n$
Salida: $c \in \mathcal{C}$ una de las palabras más próximas a w.
$I = \langle x_1^{g_{i1}} \cdots x_n^{g_{in}} - 1 | 1 \leq i \leq k\rangle + \langle x_i^2 - 1 | 1 \leq i \leq n\rangle$
Elegir " $\geq$ " orden monomial graduado.
Calcular G la base de Gröbner reducida de I respecto de " $\geq$ "
$\overline{x^w}^G = x^e$
return $w+e$

Figura 3.1. GB Decoder 1

Como se puede ver en la figura, la decodificación se consigue calculando la base de Gröbner y el resto de la división de un monomio entre su base de Gröbner.

Ejemplo 3.9. Retomamos el código $\mathcal{C}$ del Ejemplo 2.25, donde a partir de la matriz generatriz G del código, tenemos que $I(\mathcal{C}) = \langle x_1x_6 - 1, x_2x_3x_5 - 1, x_4x_5x_6 - 1\rangle + \langle x_i^2 - 1 | 1 \leq i \leq n\rangle$, hagamos su base de Gröbner reducida respecto del orden lexicográfico graduado con ayuda de `Sagemath`:

```
In : P.<x1,x2,x3,x4,x5,x6>=PolynomialRing(GF(2),6,order='deglex')
In : I=ideal(x1*x6 -1,x2*x3*x5 -1,x4*x5*x6 -1,x1^2-1,x2^2-1,
    x3^2-1,x4^2-1,x5^2-1,x6^2-1)
In : B=I.groebner_basis()
In : B
```

```
Out: [x2^2 + 1, x2*x3 + x5, x2*x4 + x3*x6, x2*x5 + x3,
    x2*x6 + x3*x4,x3^2 + 1, x3*x5 + x2, x4^2 + 1, x4*x5 + x6,
    x4*x6 + x5, x5^2 + 1, x5*x6 + x4, x6^2 + 1, x1 + x6]
```

Aplicando el Teorema 3.7 vemos que la distancia mínima del código es 2, ya que el valor mínimo $w_H(a_i + b_i)$ con $x^{a_i} - x^{b_i} \in G$ se alcanza para $x_1 - x_6$ y, por tanto $d_H(\mathcal{C}) = w_H(100001) = 2$. Para calcular una de las palabras más cercanas a $w = (100101)$, vamos a hacer la división de $x^w = x_1x_4x_6$ respecto a la base de Gröbner calculada anteriormente y obtenemos el resto:

```
In : f=x1*x4*x6
In : f.reduce(B)
Out: x4
```

El resto de la división es $x^e = x_4$, por lo tanto, $e = (000100)$ y el algoritmo de la Figura 3.1 devolverá $w + e = (100001)$.

Veamos ahora un contraejemplo que muestra que el Teorema 3.8 no es cierto cuando el orden monomial, no es graduado, por ejemplo el orden lexicográfico:

```
In : P.<x1,x2,x3,x4,x5,x6> = PolynomialRing(GF(2),6,order='lex')
In : I=ideal((x1*x6 -1),(x2*x3*x5 -1),(x4*x5*x6 -1),(x1^2-1),
    (x2^2-1),(x3^2-1),(x4^2-1),(x5^2-1),(x6^2-1))
In : B=I.groebner_basis()
In : B
Out: [x1 + x6, x2 + x3*x5, x3^2 + 1, x4 + x5*x6, x5^2 + 1,
    x6^2 + 1]
In : f=x1*x4*x6
In : f.reduce(B)
Out: x5*x6
```

Observamos, que al cambiar a un orden que no es graduado, el resultado es completamente erróneo, dando $e = (000011)$ y por lo tanto devolviendo el algoritmo $w + e = (1000110)$, que sí es una palabra del código, pero no es la más próxima en distancia de Hamming a w.

El algoritmo de la Figura 3.1 tiene algunas fortalezas respecto a los del capítulo anterior. Por ejemplo, no tener que saber el conjunto de palabras minimales del código, si no que nos basta con utilizar una base de Gröbner del ideal. En cambio tiene una debilidad, y es que, en general, esta base de Gröbner suele tener muchos elementos asociados a la misma palabra del código en la base de Gröbner. Una muestra de ello es el Ejemplo 3.9 donde hay 3 binomios:

$x_4x_5 - x_6, x_4x_6 - x_5$ y $x_4x_5 - x_6$ que se corresponden con la palabra (000111) del código. Para mejorar este algoritmo, presentamos una nueva propuesta en la que combinamos este método con el algoritmo visto en el capítulo dos, el algoritmo de decodificación por descenso de gradiente 2.5, para así obtener una versión mejorada.

Sea $G = (g_1, \ldots, g_k)^T \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$ la matriz generatriz de un código binario $\mathcal{C}$ y sea $w \in \mathbb{F}_2^n$ un mensaje recibido. Veamos cómo decodificar este mensaje: En primer lugar vamos a definir el ideal del código

$$I = \langle x^{g_1} - 1, \ldots, x^{g_k} - 1, x_1^2 - 1, \ldots, x_n^2 - 1\rangle$$

y elegimos un " $\geq$ " orden monomial graduado. A continuación, calculamos G la Base de Gröbner reducida de I respecto de " $\geq$ ", que será

$$G = \{x_{I_1} - x_{J_1}, \ldots, x_{I_l} - x_{J_l}, x_{i_1}^2 - 1, \ldots, x_{i_s}^2 - 1\},$$

para ciertos $I_1, \ldots, I_l, J_1, \ldots J_l \subseteq \{1, \ldots, n\}, i_1, \ldots, i_s \in \{1, \ldots, n\}$. Donde, si $I \subseteq \{1, \ldots, n\}$, entonces denotamos $x_I = \prod_{i\in I} x_i$. Definimos el test-set asociado a $\mathcal{C}$ respecto de " $\geq$ " como,

$$\mathcal{T} := \{I_1 \cup J_1, \ldots, I_l \cup J_l\} \subseteq P(\{1, \ldots, n\}).$$

Ahora, consideramos el conjunto de binomios

$$\mathcal{H} = \{x_K - x_L | K \cap L = \emptyset, K \cup L \in \mathcal{T}, x_K \geq x_L\} \cup \{x_1^2 - 1, \ldots, x_n^2 - 1\}.$$

Se observa que $G \subseteq \mathcal{H}$ y, por tanto, $\mathcal{H}$ es una base de Gröbner de I respecto de " $\geq$ ".

Veamos cómo obtener el resto de la división de x^w respecto de $\mathcal{H}$ solo usando el test-set. Consideramos $x^w = x_W$ con $W = \text{supp}(w)$. Si $\exists i \in \{1, \ldots, l\}$ tal que $x_W > x_{W\triangle T_i}$, tomamos:

$$K = W \cap T_i \ y \ L = T_i - W.$$

Y se tiene que

$$x_W = x_K \cdot x_{W-K} \ y \ x_{W\triangle T_i} = x_L \cdot x_{W-K}$$

Entonces, como $x_W > x_{W\triangle T_i}$ podemos concluir que $x_K > x_L$ con $K \cup L = T_i$ y $K \cap L = \emptyset$, y entonces $x_K - x_L \in \mathcal{H}$ y dividiendo x_W entre $x_K - x_L$, se tiene que $x_W - x_{W-K} \cdot (x_K - x_L) = x_{W\triangle T_i}$ y, por tanto $\overline{x_W}^{\mathcal{H}} = \overline{x_{W\triangle T_i}}^{\mathcal{H}}$. Si $\forall\, i \in \{1, \ldots, l\}$ se tiene que $x_W < x_{W\triangle T_i}$, esto significa que el resto de la división de x_W entre $\mathcal{H}$ es x_W. En resumen, se puede simular la división de x^W entre $\mathcal{H}$ solo haciendo comprobaciones entre W y los subconjuntos de $\{1, \ldots, n\}$ que están en el test-set.

Siguiendo lo descrito, proponemos en la Figura 3.2 una versión mejorada del algoritmo de decodificación por bases de Gröbner.

Algoritmo de decodificación por bases de Gröbner II

Entrada: $G' \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$ matriz generatriz de $\mathcal{C}$ e $w \in \mathbb{F}_2^n$
Salida: $c \in \mathcal{C}$ una de las palabras más próximas a w.
$I = \langle x_1^{g_{i1}} \cdots x_n^{g_{in}} - 1 | 1 \leq i \leq k\rangle + \langle x_i^2 - 1 | 1 \leq i \leq n\rangle$
Elegir " $\geq$ " orden monomial graduado.
Calcular G la base de Gröbner reducida de I respecto de " $\geq$ "
Tomar $I_1, \ldots, I_l, J_1, \ldots, J_l \subseteq \{1, \ldots, n\}$
tales que $\mathcal{H} = \{x_{I_i} - x_{J_i} | 1 \leq i \leq l\} \cup \{x_{i_1}^2 - 1, \ldots, x_{i_s}^2 - 1\}$
$\mathcal{T} = \{T_1, \ldots, T_l\}$ con $T_i = I_i \cup J_i, i = 1, \ldots, l$
Tomar $W = \text{supp}(w)$, $C = \emptyset$
while $\exists\ i \in \{1, \ldots, l\}$, tal que $x_{W \triangle T_i} < x_W$ **do**
  $C = C \triangle T_i$
  $W = W \triangle T_i$
end while
return $c \in \mathcal{C}$ tal que $\text{supp}(c) = C$

Figura 3.2. GB Decoder 2

Ejemplo 3.10. Recuperando el Ejemplo anterior 2.25, y la segunda parte del Ejemplo 3.9, sigamos el algoritmo descrito en la Figura 3.2 donde recordemos que G matriz generatriz de $\mathcal{C}$ es

$$G = \begin{pmatrix} 1\,0\,0\,0\,0\,1 \\ 0\,1\,1\,0\,1\,0 \\ 0\,0\,0\,1\,1\,1 \end{pmatrix}$$

Además, el ideal obtenido es

$$I(\mathcal{C}) = \langle x_1x_6 - 1, x_2x_3x_5 - 1, x_4x_5x_6 - 1\rangle + \langle x_i^2 - 1 | 1 \leq i \leq n\rangle$$

Eligiendo el orden monomial lexicográfico graduado, obtenemos la base de Gröbner

$$\{x_2^2 - 1, x_2x_3 - x_5, x_2x_4 - x_3x_6, x_2x_5 - x_3, x_2x_6 - x_3x_4, x_3^2 - 1, x_3x_5 - x_2,$$

$$x_4^2 - 1, x_4x_5 - x_6, x_4x_6 - x_5, x_5^2 - 1, x_5x_6 - x_4, x_6^2 - 1, x_1 - x_6\}.$$

Calculemos ahora el test-set: $\mathcal{T} = \{T_1, \ldots, T_4\}$ donde $T_1 = \{2, 3, 5\}$ está asociado a los binomios $x_2x_3 - x_5, x_2x_5 - x_3, x_3x_5 - x_2$, $T_2 = \{2, 3, 4, 6\}$ está asociado a los binomios $x_2x_4 - x_3x_6, x_2x_6 - x_3x_4$, $T_3 = \{4, 5, 6\}$ está asociado a los binomios $x_4x_5 - x_6, x_4x_6 - x_5, x_5x_6 - x_4$ y $T_4 = \{1, 6\}$ está asociado a los binomios $x_1 - x_6$. Se observa que $\mathcal{T}$ es un subconjunto de las palabras minimales, donde $\mathcal{T}$ solo tiene 4 elementos y M tiene 6.
Ahora procedemos a decodificar $w = (101001)$, tomamos $W = \text{supp}(w) = \{1, 4, 6\}$. Hagamos la diferencia simétrica de I con cada elemento del test-set:

$$W \triangle T_1 = \{1, 2, 3, 4, 5, 6\}$$
$$W \triangle T_2 = \{1, 2, 3\}$$
$$W \triangle T_3 = \{1, 5\}$$
$$W \triangle T_4 = \{4\}$$

La forma más eficiente de terminar este algoritmo en el menor número de pasos posibles es tomar como la primera opción $W \triangle T_4 = \{4\}$ donde el peso 1 sería el mínimo peso y daría $C = \{1, 6\}$ y acabaríamos el algoritmo devolviendo $c = (100001) \in \mathcal{C}$.

Sin embargo, veremos qué pasa si tomamos como primera opoción T_1. Tomando $W \triangle T_1 = \{1, 2, 3, 4, 5, 6\}$, sería mayor que el soporte de W, por lo tanto, lo descartamos. Tomando $W \triangle T_2 = \{1, 2, 3\}$ también es mayor que W, y por último, tomando $W \triangle T_3 = \{1, 5\}$, se observa que $x_{W \triangle T_3} < x_W$ es menor que W, por lo tanto podemos cogerlo. Ahora, repetimos el proceso con $W \triangle T_3$, encontramos que $(W \triangle T_3) \triangle T_4 = \{5, 6\}$ y $x_5x_6 < x_1x_5$ por lo que hacemos lo mismo y encontramos ahora que $((W \triangle T_3) \triangle T_4) \triangle T_3 = \{4\}$, donde el soporte es de peso mínimo.
También en este caso el algoritmo obtiene $C = W \triangle \{4\} = \{1, 6\}$ y devuelve $c = (100001)$.

Para finalizar, citar que en el artículo [9] se encuentra un ejemplo a mayor escala. Este ejemplo es un $[14, 9]$-código donde su conjunto de palabras minimales está formado por 147 monomios. En cambio, cuándo calculamos el test-set, contenido en ese conjunto de palabras minimales, obtenemos 24 elementos. Es un gran avance ya que desciende notablemente ese conjunto.

En un algoritmo por descenso de gradiente se necesitaría calcular las 147 palabras minimales que sería muy costoso y luego hacer el descenso de gradiente, que es mas rápido. Lo que realmente hace que el algoritmo sea costoso es el cálculo del conjunto de palabras minimales. En cambio, para el primer algoritmo con bases de Gröbner, el cálculo es mas rápido que en el de descenso de gradiente, pero se necesitaría calcular los 182 elementos de la base de Gröbner y realizar el algoritmo de la división del monomio entre ellos.
Sin embargo, en el último algoritmo propuesto, se construye el test-set, dando así un conjunto de 24 elementos, donde la reducción se hará sólo respecto a este conjunto, lo que hará a este método el más eficiente.

Conclusiones

En este trabajo se han abordado dos problemas difíciles en Teoría de Códigos: el cálculo de la distancia mínima y el problema de decodificación de códigos binarios usando herramientas del Álgebra Computacional y del Álgebra Conmutativa. En el primer caso, hemos resuelto el problema del cálculo de la distancia mínima, mediante el cálculo de una base de Gröbner en el Teorema 3.7 y, en el segundo caso, el problema de decodificación hemos dado algunos algoritmos válidos, tales como el algoritmo de la decodificación por fuerza bruta en la Figura 2.2, el algoritmo de la decodificación por descenso de gradiente en las Figuras 2.4 y 2.5 y los algoritmos de decodificación por bases de Gröbner de las Figuras 3.1 y 3.2. Si bien ninguno de estos algoritmos son polinomiales, sí que suponen un avance respecto al algoritmo de fuerza bruta.

Los principales resultados en este trabajo están pensados para códigos binarios y una posible continuacion, podría ser estudiarlos para códigos q-arios, como por ejemplo se habla en el artículo [11]. Por otro lado, también, sería interesante estudiar familias de códigos específicos para los que sí se conocen algoritmos de decodificación eficientes. Un primer ejemplo podría ser el estudio de decodificación eficiente de códigos Reed-Solomon descrito en [4].

También considero interesante estudiar cómo adaptar los algoritmos de decodificación aportados para hacer decodificación por listas, donde la diferencia es que en vez de devolver una de las palabras mas cercanas del código, devuelve todas las palabras que estén a la misma distancia. El algoritmo de fuerza bruta se puede adaptar fácilmente para esta decodificación por listas, sin embargo, no está tan claro cómo adaptar los algoritmos de descenso de gradiente y de decodificación por bases de Gröbner a la decodificación por listas.

Un primer paso para atacar estos problemas podría ser el estudio de los algoritmos de decodificación por listas expuestos en [14]. Otro problema que podríamos abordar, sería el cálculo del conjunto de todas las palabras minimales, el cual se puede llevar a cabo mediante el cálculo de las denominadas bases de Graver (ver [10]). En cuanto a éstas, decir que una base de Graver suele ser notablemente mayor que una base de Gröbner.

Bibliografía

[1] A. Ashikhmin, ;A. Barg, Minimal vectors in linear codes. IEEE Trans. Inform. Theory 44 (1998), no. 5, 2010-2017.

[2] A. Barg, Complexity issues in coding theory. Handbook of coding theory, Vol. I, II, 649-754, North-Holland, Amsterdam, 1998

[3] E. Berlekamp, R. McEliece and H. van Tilborg. On the inherent intractability of certain coding problems. IEEE Transactions on Information, Theory IT:24 (1978), 384–386.

[4] E. Berlekamp, L. Welch. Error Correction of Algebraic Block Codes. US Patent 4.633.470 (1968).

[5] M. Borges-Quintana, M.A. Borges-Trenard, P. Fitzpatrick, E. Martínez-Moro. Gröbner bases and combinatorics for binary codes. AAECC 19, 393-411 (2008).

[6] B. Buchberger, An algorithm for Finding a Basis for the Residue Class Ring of a ZeroDimensional Ideal. Ph. D. Thesis, University of Innsbruck, Math. Inst., 1965.

[7] D. Cox, J. Little, y D. O'Shea, Ideals, Varieties, and Algorithms, Springer-Verlag, 1997.

[8] D. Eisenbud, B. Sturmfels, Binomial ideals, Duke Math. J. 84 (1996), 1-45.

[9] I. García-Marco, I. Márquez-Corbella, E. Martínez-Moro, Y. Pitones. Free resolutions and Generalized Hamming Weights of binary linear codes. Mathematics 2022, 10(12), 2079.

[10] I. Márquez-Corbella, E. Martínez-Moro, Algebraic structure of the minimal support codewords set of some linear codes, Adv. Math. Commun., 5 (2011), 233-244.

[11] I. Márquez-Corbella, E. Martínez-Moro, E. Suárez-Canedo. On the ideal associated to a linear code. Adv. Math. Commun. 10 (2016), no. 2, 229-254.

[12] R. Pellikaan, Xin-Wen Wu, S. Islav Bulyg In, R. Jurrius, Codes, Cryptology and Curves with Computer Algebra, 2018.

[13] `SageMath`, the Sage Mathematics Software System (Version 9.8), The Sage Developers, 2023, https://www.sagemath.org

[14] M. Sudan. List decoding: Algorithms and applications. SIGACT News, 31:16-27, 2000.

Study of binary codes through Gröbner Bases

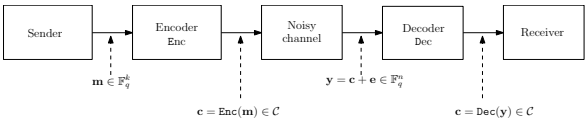
Karen Rodríguez Torres
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
Alu0101124797@ull.edu.es

Abstract

Coding Theory is a branch of Mathematics and Computer Science that aims to efficiently send a message through a channel affected by noise, so that it can be recovered by the receiver without errors. With linear codes, the goal is to introduce tools from Linear Algebra to efficiently solve problems related to codes, such as the decoding problem. The main parameters of a linear code are length, dimension, and minimum distance. The first two are related to efficiency and the third to the correcting capacity of the code.
Given a binary code, we will study two problems. The first is to calculate its minimum distance, and the second is decoding. To do this, we will associate an ideal of the polynomial ring to every binary code and solve the above problems using Gröbner bases.

1. Linear Codes

A message is transmitted through a noisy channel, as depicted in the following figure:



An objective of Coding Theory is to design encoding and decoding protocols to efficiently recover the transmitted codeword c form the perturbed message $c + e = y$.

Definition 1. A linear code $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$. Its parameters are:

- Length: $n(\mathcal{C}) = n$.
- Dimension: $k(\mathcal{C}) = \dim(\mathcal{C})$.
- Minimum distance: $d_H(\mathcal{C}) = \min\{w_H(c) \,|\, c \in \mathcal{C}, c \neq 0\}$, where $w_H(c)$ is the Hamming weight of c, i.e., its number of nonzero entries.

Definition 2. Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear code of dimension k. A $k \times n$ matrix $G = (g_{ij}) \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$ is a generator matrix of $\mathcal{C}$ if $B = \{g_1, \ldots, g_k\}$ with $g_i = (g_{i1}, \ldots, g_{in}) \in \mathbb{F}_q^n$, is a basis of $\mathcal{C}$ as $\mathbb{F}_q$-vector space.

Definition 3. A complete minimum distance decoder for a code $\mathcal{C}$ is a function:

$$\mathrm{Dec} : \mathbb{F}_q^n \longrightarrow \mathcal{C}$$

such that $\mathrm{Dec}(x) = c$, then $d_H(x, c') \geq d_H(x, c)$ for every $c' \in \mathcal{C}$.

The two main **goals of this work** are:

(1) the computation of the minimum distance, and
(2) the derivation of decoding algorithms

for binary codes (i.e., linear codes $\mathcal{C} \subseteq \mathbb{F}_2^n$).

2. Binary Codes and Grobner bases

To overcome these challenges, we associate binomial ideal to each binary code and we propose algorithms based on Gröbner bases.

Definition 4. Let $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a binary code, the ideal of $\mathcal{C}$ is the ideal:

$$I(\mathcal{C}) = \langle \{x^a - x^b \mid a, b \in \mathbb{F}_2^n, a + b \in \mathcal{C}\} \rangle + \langle \{x_i^2 - 1 \mid 1 \leq i \leq n\} \rangle.$$

Where, for $a = (a_1, \ldots, a_n) \in \mathbb{F}_2^n$ we denote $x^a := \prod_{a_i=1} x_i$.

Proposition 5. If $\mathcal{C}$ is a binary code with generator matrix $G = (g_{ij})$, then: $I(\mathcal{C}) = \langle \{x^{g_j} - 1 \mid 1 \leq j \leq k\} \rangle + \langle \{x_i^2 - 1 \mid 1 \leq i \leq n\} \rangle$.

Theorem 6. Let $B = \{x^{a_i} - x^{b_i} \mid 1 \leq i \leq s\} \cup \{x_i^2 - 1 \mid j \in J\}$ be the Reduced Gröbner basis of $I(\mathcal{C})$ with respect to a graded monomial order " $\geq$ " with $a_i, b_i \in \mathbb{F}_2^n, i \in \{1, \ldots, s\}$ and $J \subseteq \{1, \ldots, n\}$.

Then, $d(\mathcal{C}) = min\{w_H(a_i + b_i) \mid 1 \leq i \leq s\}$.

Theorem 7. Let $w \in \mathbb{F}_2^n$ and let x^e be the remainder of the division of x^w by B. Then,

1. $w + e \in \mathcal{C}$, and
2. If $c \in \mathcal{C}$, then $d_H(w, c) \geq d_H(w, w + e)$.

From this result, one derives the following decoding algorithm.

Gröbner bases decoder

Input: $G \in \mathcal{M}_{k\times n}(\mathbb{F}_2)$ generator matrix of $\mathcal{C}$ and $w \in \mathbb{F}_2^n$
Output: $c \in \mathcal{C}$ one of the closests codewords to w.
$I = \langle x_1^{q_1} \cdots x_n^{q_n} - 1 | 1 \leq i \leq k \rangle + \langle x_i^2 - 1 | 1 \leq i \leq n \rangle$
Choose " $\geq$ " a graded monomial order.
Compute B the reduced Gröbner basis of I with respect to " $\geq$ ".
Compute x^e, the remainded of the division of x^w by B.
return $w + e$

Example 8. Let $\mathcal{C}$ be a [6,3]-binary code, with generator matrix:

$$G = \begin{pmatrix} 1\,0\,0\,0\,0\,1 \\ 0\,1\,1\,0\,1\,0 \\ 0\,0\,0\,1\,1\,1 \end{pmatrix}$$

Following **Proposition 5** We consider the ideal

$$I(\mathcal{C}) = \langle x_1x_6 - 1, x_2x_3x_5 - 1, x_4x_5x_6 - 1 \rangle + \langle x_i^2 - 1 | 1 \leq i \leq n \rangle,$$

and compute with `Sagemath` its reduced Gröbner basis with respect to the graded lexicographic order:

```
In : P.<x1,x2,x3,x4,x5,x6> =
    = PolynomialRing(GF(2),6,order='deglex')
In : I = ideal(x1*x6 -1,x2*x3*x5 -1, x4*x5*x6 -1,x1^2-1,x2^2-1,
    x3^2-1,x4^2-1,x5^2-1,x6^2-1)
In : B=I.groebner_basis()
In : B
Out: [x2^2 + 1, x2*x3 + x5, x2*x4 + x3*x6, x2*x5 + x3,
    x2*x6 + x3*x4,x3^2 + 1, x3*x5 + x2, x4^2 + 1,
    x4*x5 + x6, x4*x6 + x5, x5^2 + 1, x5*x6 + x4,
    x6^2 + 1, x1 + x6]
```

By **Theorem 6**, we have that the minimum distance of the code is 2, as for $x^{a_i} - x^{b_i} \in G$, the value $w_H(a_i + b_i)$ attains the smallest value for $x_1 - x_6$, thus resulting in $d_H(\mathcal{C}) = w_H(100001) = 2$.

Consider now $w = (100101)$, to determine one of the closest words in $\mathcal{C}$ to w. We follow **Theorem 7** and compute the remainder of $x^w = x_1x_4x_6$ with respect to B, and obtain the following:

```
In : f=x1*x4*x6
In : f.reduce(B)
Out: x4
```

Since remainder of the division is $x^e = x_4$; therefore $e = (000100)$, and one of the the closest words in $\mathcal{C}$ to w is $w + e = (100001)$.

References

[1] M. Borges-Quintana, M.A. Borges-Trenard, P. Fitzpatrick, E. Martínez-Moro. Gröbner bases and combinatorics for binary codes. AAECC 19, 393-411 (2008).