Ciencias Matemáticas, Vol. 33, No. 1, Pag. 28-35, 2019

Cálculo de bases de Gröbner completas asociadas a códigos lineales
Computing complete Gröbner bases associated with linear codes

Mijail Borges Quintana[1]*, José A. Ornella Rodriguez[1]

Resumen La Teoría de Códigos (TC) surge para resolver la corrección de errores durante la transmisión de datos, con gran aplicación en las Telecomunicaciones, la Criptografía y otras ramas vinculadas a la emisión y recepción de información. Dentro de la TC los códigos lineales constituyen una estructura muy importante. Por otra parte, las bases de Gröbner (BG) constituyen una potente herramienta de trabajo en el caso de investigaciones asociadas a un álgebra polinomial. Las BG han sido conectadas con los códigos lineales a través de la asociación de ciertos ideales con los códigos. En este trabajo se define una nueva variante de BG, denominadas BG completas y se modifica el modelo GBLA (Gröbner Bases by Linear Algebra) para el cálculo de esta BG, desarrollándose una implementación de este nuevo algoritmo. Se demuestra que ciertos tipos distintivos de éstas BG completas constituyen un invariante que puede ser utilizado para determinar la equivalencia de códigos lineales. Además, se realizan comparaciones experimentales en cuanto al tiempo de cómputo del nuevo algoritmo implementado con respecto a otras implementaciones realizadas anteriormente que calculan BG. El algoritmo implementado se basa en la asociación entre las técnicas de construcción de BG y los códigos lineales. Por otra parte, se extiende el cálculo de BG asociadas a cualquier código lineal, utilizando el modelo GBLA, lo cual solo estaba disponible para códigos binarios.

Abstract Coding Theory arises to solve the error correction during data transmission, with great application in the telecommunications, Cryptography and other tied branches related to the emission and reception of information. Inside coding theory, the linear codes constitutes a very important structure. On the other hand, the Gröbner bases (GB) constitute a powerful working tool in the case of research associated to a polynomial algebra. The GB has been connected with the linear codes through the association of certain ideals with the codes. In this work a new variant of GB is defined, called complete GB and the GBLA model (Gröbner Bases by Linear Algebra) is modified for the computation of this GB, developing an implementation of this new algorithm. It is proved that certain kinds of complete GB constitutes an invariant which can be used to determine the equivalence of linear codes. Also, experimental comparisons are made in terms of the computation time of the new algorithm implemented with respect to other implementations carried out previously for computing GB. The algorithm implemented is based on the association between GB construction techniques and linear codes. On the other hand, the GB computation associated with any linear code is extended, using the GBLA model, which was only available for binary codes.

Palabras Clave
Bases de Gröbner, Algoritmo de Möller, Códigos lineales, Equivalencia de códigos

Keywords
Gröbner bases, Möller algorithms, Linear codes, Equivalence of linear codes

[1] *Departamento de Matemática, Universidad de Oriente, Santiago de Cuba, Cuba, mijail@uo.edu.cu, jornella@uo.edu.cu*
***Autor para Correspondencia, Corresponding Author**

Introducción

La TC forma hoy un extenso y fructífero campo de interacción entre las Matemáticas y las Tecnologías de la Información, en el que conceptos y resultados matemáticos abstractos permiten dar elegantes soluciones al problema de transmitir información de forma eficiente y segura. Entre estos conceptos matemáticos juegan un papel relevante el álgebra lineal y abstracta y la definición de la estructura de códigos lineales que constituyen los códigos correctores de errores más utilizados y de mayor aplicación.

El concepto de BG se introduce en la tesis doctoral de Bruno Buchberger en 1965 con el objetivo de resolver el problema de la pertenencia de un polinomio a un ideal. Como el

aporte fundamental de Buchberger se considera la formulación y fundamentación de un algoritmo para el cálculo de las BG (Algoritmo de Buchberger).

En [**?**] se introduce el modelo GBLA (Gröbner Basis by Linear Algebra) en el cual se formaliza un contexto y algoritmo patrón general de algoritmos tipo Möller, como se reconoce mayoritariamente por la comunidad científica internacional, para la utilización de técnicas de Álgebra Lineal para el cálculo de BG y otras estructuras similares. El ideal asociado con el modelo GBLA para el caso de códigos lineales se denomina el ideal asociado al código lineal. En el 2013, en [**?**] se introduce otra forma de construir el ideal asociado al código (ideal generalizado del código).

Con el ideal generalizado, aunque se introducen más indeterminadas, se puede resolver la decodificación de cualquier código lineal mediante el cálculo de una BG reducida de manera análoga a como se hace en el caso binario para la representación estándar.

En [**?**] se trata lo referido a la equivalencia de códigos binarios, que plantea que dado un código y una BG reducida asociada a este, se puede ver que, permutando esta base, obtendremos una base reducida asociada a un código que es el resultado de aplicar la misma permutación al código dado, es decir, obtenemos una BG reducida de un código equivalente. No obstante, si se desea hallar una permutación bajo la cual dos códigos son equivalentes, teniendo dos BG de estos códigos, en general no es posible, puesto que estas bases no son necesariamente equivalentes, es decir una no es necesariamente la imagen por una permutación de la otra. Por ello, surge la necesidad de potenciar las cualidades investigadas de las BG de códigos binarios, de forma que se obtengan verdaderas estructuras invariantes para códigos equivalentes y además no solamente binarios. De esta manera surge la estructura algebraica principal de este trabajo, las BG completas asociadas a un código lineal.

1. Preliminares

1.1 Códigos lineales

En esta sección se estudiarán algunas nociones sobre la teoría de los códigos lineales. Un estudio más detallado se puede encontrar en [**?**].

Sea $\mathbb{F}_q$ el campo de Galois de q elementos ($q = p^m$, donde p es un número primo) [1].

Definición 1 (Código Lineal) *Sea* $L : \mathbb{F}_q^k \to \mathbb{F}_q^n \quad (k < n)$ *una aplicación lineal.*
$C = L(\mathbb{F}_q^k) \subset \mathbb{F}_q^n$ *es el* **código lineal** *definido por L. Los elementos en C se denominan* palabras del código . *Se dice entonces que C tiene* dimensión *k y* longitud *n.*

Los códigos utilizados en la práctica (excepto algunos de pequeño tamaño) son siempre lineales. A partir de este momento al hablar de códigos nos estaremos refiriendo a códigos lineales.

[1] el número p se denomina *característica* del campo.

Asociada a toda aplicación lineal L existe una matriz G llamada matriz generatriz, que es la matriz de la aplicación lineal L, la cual genera todas las palabras del código ($C = \{xG^t : x \in \mathbb{F}_q^k\}$), sin embargo, un subespacio vectorial de $\mathbb{F}_q^n$ puede describirse no sólo mediante un sistema de generadores (lo que da lugar al concepto de generatriz), sino también mediante unas ecuaciones implícitas, entonces, existe una matriz $H_{n\times(n-k)}$ [2] tal que un vector c de $\mathbb{F}_q^n$ es una palabra del código si, y sólo si, satisface el sistema

$$cH \quad = \quad 0. \tag{1}$$

Como consecuencia el conjunto C se puede caracterizar como el conjunto de todos los elementos en $\mathbb{F}_q^n$ que satisfacen el sistema anterior. H se denomina *matriz de control (parity check matrix)* para el código.

Definición 2 (Distancia de Hamming) *Se denomina* **distancia de Hamming** *entre dos vectores a la cantidad de componentes diferentes entre ellos. O sea, dados* $x, y \in \mathbb{F}_q^n$

$$d_H(x,y) = \#\{i | 1 \le i \le n, x_i \neq y_i\}. \tag{2}$$

Definición 3 (Peso) *El* **peso** *de un vector v se define como su distancia de Hamming desde el vector nulo. El peso del vector y se representará por* $w_H(y)$.

Luego $w_H(w) = d(w,0)$. Además, $d_H(x,y) = w_H(x-y)$.

Definición 4 (Soporte) *Sea* $v \in \mathbb{F}_q^n$*, el soporte del vector v, denotado por* $supp(v)$*, se define como*

$$supp(v) = \{i : v_i \neq 0\}. \tag{3}$$

Definición 5 (Distancia Mínima) *El menor peso de las palabras del código no nulas se conoce como* **distancia mínima** *del código.*

Definición 6 (Capacidad Correctora) *Si d es la distancia mínima del código C, se conoce que pueden corregirse hasta t errores, donde* $d = 2t+1$ *o* $d = 2t+2$*, este parámetro t se conoce como la* **capacidad de corrección** *de errores de un código.*

Se dice que un código es $[n,k,t]$ si tiene longitud n, dimensión k y capacidad correctora t (también se utiliza la notación $[n,k,d]$ donde d es la distancia mínima del código), o simplemente se pondrá $[n,k]$ si no se precisa disponer de t.

De las expresiones obtenidas en la definición anterior se deduce que

$$t \quad = \quad \left[\frac{d-1}{2}\right], \tag{4}$$

donde $[\cdot]$ representa la función *parte entera*.

[2] las dimensiones pueden variar de una literatura a otra según el orden de multiplicación que se utilice.

Definición 7 (Vector Error) *Dada una palabra recibida $y \in \mathbb{F}_q^n$, el* **vector error** *e es la diferencia entre la palabra del código transmitida c e y, es decir, $e = y - c$. Cuando y contiene cuanto más t errores, e es la única solución, con peso acotado por t, del sistema:*

$$eH \quad = \quad yH. \tag{5}$$

Definición 8 (Síndrome) *El* **síndrome** *de un vector $y \in \mathbb{F}_q^n$ es el valor*

$$\xi(y) = yH \in \mathbb{F}_q^{n-k}. \tag{6}$$

Un código C define una relación de equivalencia R_C en $\mathbb{F}_q^n$ dada por

$$\begin{aligned}(y_1, y_2) \in R_C &\Leftrightarrow y_1 - y_2 \in C \\ &\Leftrightarrow \xi(y_1) = \xi(y_2); y_1, y_2 \in \mathbb{F}_q^n.\end{aligned} \tag{7}$$

Luego, R_C define q^{n-k} clases de equivalencias (dado que el espacio de los síndromes es $\mathbb{F}_q^{n-k}$) de tal forma que en la clase $y + C$ se encuentran los vectores que tienen el mismo síndrome que y.

Definición 9 (Líder de coseto) *Los vectores de menor peso en una clase de equivalencia o coseto se les denomina* ***líderes del coseto****. En general un coseto $y + C$ puede tener más de un líder. El conjunto que estos forman se denota por $CL(y)$.*

El conjunto formado por todos los líderes de cosetos de un código se denota por $CL(C)$ y cumple que $CL(y) \subset CL(C)$.

Un código t corrector de errores puede decodificar correctamente cualquier vector recibido y que se encuentre a distancia a lo sumo t de una palabra del código. El proceso de decodificación consistiría en hallar el vector error e correspondiente a y, que sería el único vector de peso a lo sumo t, que se encuentra en la clase de equivalencia (o coseto) del vector y. Luego como resultado de este proceso se obtendría la palabra del código $y - e$.

Lema 10 ([?], Teorema 1.12.6.v) *Sea y un líder de coseto de C. Si $y' \in \mathbb{F}_q^n$ y $y'_i = y_i$ para todo $i \in supp(y')$, entonces y' también es un líder de coseto de C.*

Definición 11 (Códigos equivalentes) *Si se define S como el conjunto de las permutaciones de orden n y sea $\sigma \in S_n$, tal que*

$$\sigma(C) = \{(y_{\sigma^{-1}(i)})_{i=1}^n | (y_i)_{i=1}^n \in C\}. \tag{8}$$

Se dice que C y $\sigma(C)$ son ***códigos equivalentes*** *y se denota por $C \sim \sigma(C)$.*

En [**?**] se asocia un cierto monoide con F_q^n y por lo tanto con el código, de manera que cada monomio está asociado con un vector de F_q^n y a cada vector de F_q^n le corresponde un conjunto de monomios de los cuales existe uno solo estandarizado. Siendo X el conjunto de indeterminadas del monoide $[X]$ asociado al código, por x^a se denota el monomio asociado con el vector $a \in F_q^n$ (ver [**?**],[**?**]).

Entonces se denomina ***Ideal asociado al código*** C al siguiente ideal:

$$I(C) = \left\langle \{x^a - x^b | (a,b) \in R_C\} \right\rangle \subseteq K[X]. \tag{9}$$

Es decir, el ideal binomial generado por las diferencias de elementos en $[X]$ cuyos exponentes están relacionados por R_C definida en [**?**].

Existen dos maneras distintas de introducir el conjunto de las indeterminadas X. En [**?**] se introduce X considerando el campo F_q como una extensión simple del campo F_p siendo p la característica de F_q. Para la otra representación se parte de que el grupo $(F_q^*, \cdot)$ es cíclico, por lo que los elementos diferentes de cero son de la forma α^j con $j = 1, ..., q-1$, donde α se denomina elemento primitivo.

Se representa por $w_1, ..., w_k$ las filas de la matriz generatriz G del código y se define el siguiente ideal:

$$\Delta I = \left\langle \left\{ x^{(\alpha^j w_i)} - 1 \right\} \bigcup \{R_{x_i}(T_+)\} \right\rangle; \tag{10}$$

donde está formado por todos los binomios en la variable asociados a las relaciones de dependencia entre los vectores asociados a esas variables dadas por la tabla aditiva del campo $F_q = \langle \alpha \rangle \cup \{0\}$.

$$R_{x_i}(T_+) = \left\{ \begin{array}{c} \{x_{iu}x_{iv} - x_{iw} : \alpha^u + \alpha^v = \alpha^w\} \\ \cup \\ \{x_{iu}x_{iv} - 1 : \alpha^u + \alpha^v = 0\} \end{array} \right. \tag{11}$$

con $i = \{1, ..., n\}$.

El ideal ΔI es el ***Ideal generalizado*** asociado al código C. Con esta representación del ideal todo ideal del código se comporta como en el caso binario. Los ordenamientos compatibles con el grado total son compatibles con el peso de los vectores asociados a los monomios. De esta forma, los monomios de menor grado están asociados con los vectores de menor peso.

1.2 Bases de Gröbner

Definición 12 (Ordenamiento de términos) *Por ordenamiento de términos en $[X]$ se define a un orden total $\prec$ en $[X]$ que satisface las siguientes condiciones:*

1. *$1 \prec x^b$ para todo $x^b \in [X], x^b \neq 1$;*
2. *si $x^a \prec x^b$, entonces $x^a x^c \prec x^b x^c, \forall x^c \in [X]$.*

Además, dado un ordenamiento de términos auxiliar $<$, se dice que un ordenamiento $\prec_T$ es de grado total si

$$x^a \prec_T x^b \Leftrightarrow \left\{ \begin{array}{ll} deg(x^a) < deg(x^b) & \text{o} \\ deg(x^a) = deg(x^b) & \text{y } x^a < x^b \end{array} \right.$$

donde $deg(x^a)$ denota el grado de x^a. Un ordenamiento de términos de grado total es el ordenamiento Graduado

Lexicográfico (denotado por $<_{deglex}$) el cual emplea como ordenamiento auxiliar el ordenamiento Lexicográfico. Otro ordenamiento de grado total muy utilizado es el Graduado Reverso Lexicográfico ([**?**]).

Definición 13 *Para $f = \sum_{i=1}^{m} c_i s_i$, donde $c_i \in K \setminus \{0\}$, $s_i \in [X]$, y $s_m \prec s_{m-1} \prec \ldots \prec s_1$:*

- *El término máximo de f con respecto a $\prec$ es $T_\prec(f) = s_1$ (se referirá también a este término como la* cabeza *de f).*
- *El coeficiente principal de f con respecto a $\prec$ es: $LC_\prec(f) = c_1$.*
- *El resto de f con respecto a $\prec$ es: $Rest_\prec(f) = f - LC_\prec(f)T_\prec(f)$.*
- *El conjunto de términos máximos de F con respecto a $\prec$ es:*
$$T_\prec\{F\} = \{T_\prec(f) \mid f \in F \setminus \{0\}\}.$$
- *El conjunto de todos los términos máximos de un ideal I será denotado por $T(I)$.*
- *El conjunto de los múltiplos de los terminos máximos de F con respecto a $\prec$ es:*
$$T_\prec(F) = T(< F >).$$

Cuando no exista riesgo de confusión, el símbolo $\prec$ se omitirá de las notaciones. Esta convención será también utilizada en las restantes notaciones y definiciones de este trabajo.

Teorema 14 *Sea $Span_K(N(I))$ el K-espacio vectorial cuya base es $N(I) = [X] \setminus T(I)$. Entonces:*

i. $K[X] = I \oplus Span_K(N(I))$.

ii. Para cada $f \in K[X]$ existe un único polinomio de $Span_K(N(I))$, denotado por $Can(f,I)$, tal que $f - Can(f,I) \in I$; más aún:
- *$Can(f,I) = Can(g,I)$ si, y sólo si, $f - g \in I$.*
- *$Can(f,I) = 0$ si, y sólo si, $f \in I$.*

iii. Para cada $f \in K[X]$, $T(Can(f,I)) \preceq T(f)$.

iv. Existe un isomorfismo de K-espacios vectoriales entre $Span_K(N(I))$ y $K[X]/I$ (el isomorfismo asocia $Can(f,I)$ con la clase de f módulo I).

$Can(f,I)$ se denomina la forma normal o forma canónica del elemento f con respecto a un ordenamiento dado. Note que este elemento es el representante de f en el espacio vectorial $Span_K(N(I))$ de las clases residuales de $K[X]$ módulo el ideal I. Es decir, el conjunto $N(I)$ es el conjunto de representantes de las distintas clases residuales, de manera que, cada uno de ellos es el menor polinomio respecto al ordenamiento que se encuentra en su clase.

Definición 15 (Base de Gröbner) *$G \subset I \setminus \{0\}$ recibe el nombre de* **base de Gröbner** *de I si $T\{G\}$ genera $T(I)$, es decir, si todo $t \in T(I)$ es un múltiplo de algún $s \in T\{G\}$.*

Definición 16 (Base de Gröbner reducida) *Una BG G de I se denomina* **reducida** *si para todo $g \in G$*

i. $T(g)$ *no es múltiplo de ningún $s \in T(G) \setminus \{T(g)\}$,*

ii. $LC(g) = 1$,

iii. $g = T(g) - Can(T(g), I)$.

Las BG reducidas constituyen un ente único para un ideal y un ordenamiento dados, no así cualquier otra BG. Además, son mínimas respecto a la cantidad de polinomios requeridos para que un conjunto de polinomios sea una BG para un ordenamiento. Por estas razones, es de interés en algunos casos trabajar con una BG reducida y no con una arbitraria.

Diversos autores coinciden en que la teoría de las BG se ha convertido en una de las principales áreas de investigación del Álgebra Computacional (AC). Los mayores resultados de la teoría de las BG están asociados con los anillos conmutativos de polinomios. Verdaderamente, esta teoría está generando un interés creciente, al suministrar herramientas computacionales que son aplicables a un amplio rango de problemas en la Matemática, la Ciencia de la Computación y otras ramas del conocimiento.

Para el trabajo específicamente con los códigos resultan de particular importancia los ordenamientos de términos compatibles con el grado, debido a que las formas canónicas con respecto a este tipo de orden están directamente asociadas a los vectores de menor peso, es decir, a los vectores errores (líderes de los cosetos correspondientes) [**?**].

Sean C un código binario de longitud n,

$$\sigma \in S_n, C^* = \sigma(C).; \tag{12}$$

Teorema 17 (Teorema 6 de [?]) *Sea G una BG de un código binario C, $x_1 \prec \ldots, \prec x_n$ donde $\prec$ es un ordenamiento de términos compatible con el grado total y $\sigma \in S_n$. Entonces $G^* = \sigma(G)$ es una BG reducida para C^* y el orden $\prec$ para el orden de las indeterminadas dado por $x_{\sigma(1)} \prec \ldots \prec x_{\sigma(n)}$.*

Es decir, si las BG G^* y G son equivalentes ($G^* = \sigma(G)$), entonces los códigos C^* y C son equivalentes.

Se puede notar que cuando σ no es conocido, G^* tampoco lo es, pero puede calcularse la BG reducida para $C^*(G')$ considerando un orden para las variables. La relación existente entre estas dos bases es que ambas son BG reducidas de C^*, pero con respecto a diferente orden de las variables; y todo binomio de G^* se reduce a cero módulo G' y viceversa.

Notar además que si se conocen las bases y se desea hallar una permutación que satisface (12), no siempre es posible, pues las BG de códigos equivalentes no son necesariamente equivalentes. Es necesario entonces encontrar algún invariante que permita resolver este problema.

En esta investigación se ha desarrollado una implementación del modelo GBLA, para el cálculo de BG reducidas en códigos no binarios, siguiendo [**?**] para el trabajo con el ideal generalizado. Se ha podido comprobar experimentalmente con varios códigos que la implementación realizada para el cálculo de BG de códigos resulta más eficiente, comparando los tiempos de cómputo, que las funciones que están implementadas en los sistemas computacionales GAP y MAPLE.

1.3 El modelo GBLA

En [**?**] se presenta el modelo GBLA (Gröbner Bases by Linear Algebra) para el cálculo de BG y estructuras similares de ideales cero-dimensionales, el cual se apoya en las técnicas del álgebra lineal como su principal herramienta. En el último capítulo de esta tesis doctoral se asocian las BG con los códigos de manera que el problema de la decodicación en códigos se transforma en el problema de la reducción a su forma canónica, mediante ciertas BG o estructuras similares ([**?**]).

Precisamente esta relación entre la pertenencia a un ideal y la dependencia lineal en el espacio vectorial cociente permite la utilización del Álgebra Lineal para el cálculo de BG y otras estructuras relacionadas. Como elemento común en todos estos casos resulta esencial la construcción del espacio vectorial de las formas canónicas isomorfo al espacio vectorial cociente que determina el ideal. A los objetos requeridos junto con el algoritmo de cálculo mediante álgebra lineal se denomina modelo **GBLA**. Este modelo puede ser especificado a muchas situaciones concretas donde se logre realizar la conexión entre sus objetos y estructuras con un álgebra específica en cuestión, donde la naturaleza del cálculo depende de esta álgebra concreta.

Este algoritmo se basa en la generación de los monomios con respecto al orden y decidir mediante dependencia lineal los monomios que corresponden a términos máximos de la BG. Por otra parte, aquellos elementos que sean linealmente independientes irán construyendo una base del espacio cociente de las clases de equivalencia que determina el ideal (en este caso se corresponde con las clases de equivalencia que determina el código).

2. Base de Gröbner completa

En esta sección se mostrará un tipo de BG que constituye un invariante para cualquier código.

Definición 18 *Llamamos **BG completa** (GC) con respecto al orden $\prec$, a una BG asociada al código C, donde al cambiar el orden de las indeterminadas, continua siendo una BG para dicho código con respecto al orden $\prec$.*

Una BG completa distintiva es la siguiente:

Definición 19 *Llamamos **BG completa reducida** a la unión de todas las BG reducidas con respecto a todos los posibles órdenes de las indeterminadas y la denotaremos como GrC.*

Como consecuencia directa de la definición 19 se puede observar la unicidad de esta BG y el cumplimiento del siguinte lema.

Lema 20 *Un binomio está en GrC si y solo si pertenece a una BG reducida para algún orden de las indeterminadas.*

Se cumple que al permutar la *GrC* de un código *C* se obtiene la *GrC* del código equivalente a *C* correspondiente a la misma permutación, debido a la unicidad de la misma y esto constituye una generalización del teorema 17.

Una BG completa reducida está compuesta por todos los binomios de la forma $w - w'$ tales que w' sea la forma canónica de w, pero lograr esta construcción computacionalmente es difícil porque dentro de los líderes de cosetos no todos tienen que constituir forma canónica para un orden de las indeterminadas. Por tanto mostraremos otra BG completa, que tendrá características similares a la *GrC*, en la cual se sustituye el papel de las formas canónicas en *GrC* por el de los líderes de cosetos.

Definición 21 *Sea $w = x^a \in [X]$, decimos que esta palabra es un término irredundante si para todo $i \in supp(a)$ que cumple que $w = x^b x_i$ entonces $b \in CL(C)$. Al conjunto formado por todos estos términos irredundantes lo denotaremos por $TI(C)$.*

La definición 14 se puede expresar de la siguiente manera:

$$x^a \in TI(C) \Leftrightarrow \begin{cases} x^a = x^b x_i \Rightarrow b \in CL(C) \\ \forall x_i \in supp(x^a) \end{cases}. \qquad (13)$$

Note que cada elemento de este conjunto de términos irreduntes, o bien es un término correspondiente a un coseto con un único líder o resultará ser término máximo con respecto a $\prec$ para órdenes específicos de las indeterminadas.

Definición 22 *Llamamos BG completa reducida extendida (GC_e) a una BG completa definida de la siguiente forma:*
$GC_e = \{x^a - x^b : x^a \in TI(C), b \in CL(a), a \neq b\}$.

Definición 23 *Sea Q un conjunto de elementos de $[X]$. Llamaremos nivel de grado k al conjunto Q_k tal que:*

$$Q_k = \{v \in Q \mid deg(v) = k\} \qquad (14)$$

El subconjunto de elementos de Q hasta el nivel k se denotará por $Q_{[k]}$.

A continuación veremos el algoritmo que permite calular una GC_e.

Nos referiremos primero a algunas subrutinas del algoritmo.

InsertNext[w,List] inserta apropiadamente los productos xw (para $x \in X$) en *List*, que se mantiene ordenada en orden creciente con respecto al ordenamiento de términos $\prec$.

Member$[v', \{v_1, \ldots, v_r\}]$ devuelve una primera componente binaria que toma valor TRUE si v' está en la lista de los síndromes ($\{v_1, \ldots, v_r\}$) o FALSE, en caso contrario. Y como

segunda componente devuelve la posición del síndrome de ese elemento en caso de que esté en esa lista.

El objeto **N** es una lista donde se va constuyendo el conjunto de los líderes de cosetos del código, cada componente de **N** corresponde a un coseto.

El objeto **List** es una lista donde se van insertando en orden creciente según el ordenamiento de términos $\prec$ los múltiplos de líderes de cosetos del código. Es decir, este objeto que se va generando contiene al conjunto de términos irredudantes.

Algoritmo 24 (Algoritmo GBLA para el cálculo de GC_e)

Entrada: *p, n, m, H los parámetros que definen un código lineal.*

Salida: $GC_e(I,\prec)$.

1: $G := \emptyset; List := \{1\}; N := \emptyset; r := 0;$
2: ***while*** $List \neq \emptyset$ ***do***
3: $w :=$ ***NextTerm***$[List];$
4: ***if*** $w \in TI(C)$ ***then***
5: $c :=$*False;*
6: $v' := \xi(w);$
7: $(\Lambda, j) :=$ ***Member***$[v', \{v_1, \ldots, v_r\}];$
8: ***if*** $\Lambda =$ **True** ***then***
9: ***if*** ***(deg***$(w_{ji}) \neq 0$***) and (deg***(w)***=deg***$(w_{ji}))$ ***then***
10: $List :=$ ***InsertNext****$[w, List];$
11: $N_j := N_j \cup \{w\};$
12: $c :=$*True;*
13: ***end if***
14: ***for*** $i = 1$ ***to Length*** $[N_j]$ ***do***
15: $G := G \cup \{w - w_{ji}\};$
16: ***if*** $c =$*True* ***then***
17: $G := G \cup \{w_{ji} - w\};$
18: ***end if***
19: ***end for***
20: ***else***
21: $r := r + 1;$
22: $v_r := v';$
23: $w_{r1} := w; N := N \cup \{w_{r1}\};$
24: $List :=$ ***InsertNext***$[w_{r1}, List];$
25: ***end if***
26: ***end if***
27: ***end while***
28: *return G;*

Para demostrar el funcionamiento del algoritmo nos apoyaremos de los siguientes teoremas.

Teorema 25 *Sea $x^a \in [X]$. Entonces $a \in CL(C)$ si y solo si $x^a \in List$ y $x^a \in N$.*

Demostración.

Para la siguiente demostración seguiremos el principio de inducción sobre $[X]$ con respecto a los grados de los monomios.

Sea $x^a \in [X]$ tal que $a \in CL(C)$. Demostraremos que $x^a \in List$ y $x^a \in N$, mediante el funcionamiento del algoritmo.

Asumiremos como hipótesis de inducción que para todas las palabras $x^a \in [X]$ hasta grado k, $a \in CL(C)$ si y solo si $x^a \in List$ y $x^a \in N$.

Demostraremos que si el grado de x^a es $deg(x^a) = k+1$ entonces se cumple que $x^a \in List$ y $x^a \in N$.

Primeramente podemos expresar x^a como $x^a = x_i x^b$, con $x_i \in X$, donde $deg(x^b) = k$.

Al tener que $a \in CL(C)$ entonces $b \in CL(C)$ por Lema 10 y $x^b \in List$ por hipótesis de inducción, pues se cumple que $deg(x^b) = k$.

Por tanto, el algoritmo insertará en *List* a los multiplos de x^b en el paso 10 o en el paso 24, en particular a $x^a = x_i x^b$ con $x_i \in X$ demostrando que efectivamente $x^a \in List$.

Cumpliendose que $x^a \in List$ el algoritmo garantizará en el paso 3 tomar ese elemento y como $a \in CL(C)$, por el Lema 10, $x^a \in TI(C)$, y por ende se añadirá x^a a la lista N en los pasos 11 ó 23.

Ahora pasaremos a la segunda parte de la demostración. Demostraremos que una palabra x^a de grado $k+1$ que se añada a *List* y a N cumple que $a \in CL(C)$.

Sea la palabra x^a, tal que $deg(x^a) = k+1$. Si $x^a \in List$ significa que su forma canónica también pertenece a *List*, es decir, $Can(x^a) \in List$ pues $deg(Can(x^a)) \leq k+1$ y por hipótesis este elemento estará en *List* y por ende estará en N. Note que $Can(x^a)$ es el menor elemento, según el orden $\prec$ que utiliza el algoritmo, cuyo coseto es el mismo de x^a, por tanto, $Can(x^a) \in N$ por el paso 23. Note además que $Can(x^a)$ es una palabra que corresponde a un líder de coseto, debido a la compatibilidad entre el orden $\prec$ para los monomios y el peso de los vectores correspondientes a los monomios. Como $x^a \in N$ y $Can(x^a) \in N$ pueden ocurrir dos cosas:

1. $x^a = Can(x^a)$ lo que significaría que x^a entró en el paso 23 y por tanto $a \in CL(C)$; ó

2. $x^a \neq Can(x^a)$, en este caso se satisface condición del paso 9, añadiéndose a N en el paso 11, lo que quiere decir que $a \in CL(C)$.

De esta forma queda demostrado que bajo cualquier circunstancia $a \in CL(C)$, lo que da por concluida la prueba del teorema. ■

Teorema 26 *El Algoritmo 24 calcula la GC_e del código C con respecto al orden $\prec$, es decir, $G = GC_e$.*

Demostración.

Seguiremos la misma idea que la demostración del teorema 25.

Asumiremos como hipótesis de inducción que G y GC_e son iguales restringiendo los conjuntos a aquellos binomios que tienen términos máximos hasta grado k, es decir $G_{[k]} = GC_{e[k]}$. Demostraremos que ocurre lo mismo si tomamos los términos máximos hasta grado $k+1$ ($G_{[k+1]} = GC_{e[k+1]}$).

Si no existe ningún término máximo en GC_e de grado $k+1$, significa que todos los elementos de $TI(C)$ de grado $k+1$ corresponden a líderes de cosetos que constituyen únicos líderes en sus cosetos. Por ende, es claro por el funcionamiento del algoritmo que $\Lambda = False$ para todos los términos de grado $k+1$. Viceversa, si no hay términos máximos en G de grado $k+1$ y haberse incluido en *List* todos los elementos que constituyen términos irredundantes de grado $k+1$ (por Teorema 25) entonces se obtiene la misma conclusión con respecto a los elementos de $TI(C)$ de grado $k+1$. Por tanto, no existiría ningún término máximo en GC_e de grado $k+1$. En este caso entonces $G_{[k]} = G_{[k+1]}$ y $GC_{e[k+1]} = GC_{e[k]}$, por lo que por hipótesis de inducción $G_{[k+1]} = GC_{e[k+1]}$.

Analicemos ahora el caso donde si existan términos máximos de grado $k+1$. Sea $x^a - x^b \in GC_{ek+1}$, con $deg(x^a) = k+1$. Entonces por definición de GC_e se cumple:

$$x^a \in TI(C), b \in CL(a), a \neq b.$$

Si $x^a \in TI(C)$, $x^a \in List$ por el Teorema 25 y x^a satisface paso 4. Igualmente, por el Teorema 25, $b \in CL(a)$ implica que $x^b \in List$ y $x^b \in N$. Por tanto, $x^a - x^b \in G$ por los pasos 15 o 17 del algoritmo. Por lo que, $GC_{ek+1} \subset G_{k+1}$.

Veamos ahora la otra inclusión. Sea $x^a - x^b \in G_{k+1}$, con $deg(x^a) = k+1$. Entonces, teniendo en cuenta los pasos 4, 15 y 17 del algoritmo se tiene que $x^a \in TI(C)$, $x^b \in TI(C)$, y si la inclusión en G se realiza en paso 15, $x^b \in N$ y $b \in CL(a)$ (por el Teorema 25), de donde $x^a - x^b \in GC_{ek+1}$.
Si la inclusión en G se realiza en paso 17, $x^a \in N$ y $x^b \in N$, y $b \in CL(a)$, por tanto, $x^a - x^b \in GC_{ek+1}$.

Con esto queda demostrado que $G_{k+1} \subset GC_{ek+1}$, por lo que $G_{k+1} = GC_{ek+1}$ y utilizando la hipótesis de inducción se obtiene que $G_{[k+1]} = GC_{e[k+1]}$.

■

Note que la finitud del algoritmo es clara debido a la cantidad finita de cosetos que tiene el código C, y la acotación de la cantidad de elementos de *List* en función de ello.

Teorema 27 *Sea G una GC_e de un código C, con respecto a un ordenamiento de términos $\prec$ compatible con el grado total y $\sigma \in S_n$. Entonces $G^* = \sigma(G)$ es la GC_e para $C^* = \sigma(C)$ y el orden $\prec$.*

Demostración.

Teniendo en cuenta el cálculo de la GC_e para el C^* en el Algoritmo 24, para el orden $\prec$, tomando $x_{\sigma(1)} \prec x_{\sigma(2)} \prec ... \prec x_{\sigma(n)}$. Es claro que $List* = \sigma(List)$. Como una consecuencia, $N^* = \sigma(N)$, donde *List* y N se obtendrían para C por el Algoritmo 24, para el orden $\prec$, con $x_1 \prec x_2 \prec ... \prec x_n$. Por ende, la relación que se obtiene entre las GC_e calculadas por el algoritmo para C y C^* será $G^* = \sigma(G)$. ■

Note que si tenemos las GC_e G^* para C^* y G para C respectivamente, de manera que $G^* = \sigma(G)$; entonces, puede demostrarse que $C^* = \sigma(C)$ siguiendo una idea muy similar a la desmostración de la suficiencia del Teorema 3 de [**?**].

El Algoritmo 24 calcula la GC_e por niveles, por lo que es posible detener el mismo una vez calculada esta base hasta el nivel deseado. Este elemento puede ser importante en la determinación de la equivalencia de códigos, el hecho es que si la GC_e es un invariante, también lo es ésta base hasta un nivel dado, la cual constituye una estructura más pequeña.

3. Experimentación

La Tabla 1 muestra una comparación de códigos sobre F_3 y F_4 en cuanto al tiempo que demoran estos códigos al calcular su BG reducida por medio de la implementación del algoritmo GBLA realizada en GAP [8], con el tiempo que demora en calcular la misma base mediante el Sistema de Cálculo Simbólico Maple que tiene incorporado algoritmos eficiente para el cálculo de BG.

Como se puede observar el tiempo que demora en calcular una BG por la implementación realizada es mucho menor que si se calcula la misma base en el MAPLE. Esta comparación se puede apreciar de forma gráfica en la Figura 1. En la celda donde se observa “ERROR", significa el mensaje que devuelve el MAPLE al no poder calcular la BG de dicho código.

				t(Segundos)	t(Segundos)
Cód	**n**	**k**	**Cosetos**	**GAP/GBLA**	**MAPLE**
CF4-1	5	2	64	0.4	1.8
CF3-2	7	3	81	0.7	1.0
CF3-3	7	2	243	1.7	0.7
CF4-4	8	4	256	4.4	338.3
CF3-5	10	4	729	11.4	39.1
CF3-6	11	5	729	13.4	586
CF4-7	8	3	1024	53.2	ERROR

Tabla 1. Tiempo de ejecución de BG en GAP/GBLA y MAPLE.

Estas comparaciones se pueden observar gráficamente en la Figura 1.

Los códigos se organizan según el tamaño de la cantidad de cosetos que determina el código q^{n-k}, lo cual constituye uno de los factores que más influye en la complejidad con el cálculo algorítmico sobre éstos, en particular incide directamente en el tiempo de ejecución del Algoritmo GBLA.

Estos experimentos fueron realizados en una computadora con microprocesador Intel(R) Core(TM) i3-3120M CPU con núcleos a 2.50GHz cada uno, memoria RAM de 6 GB y sistema operativo de 64 bits.

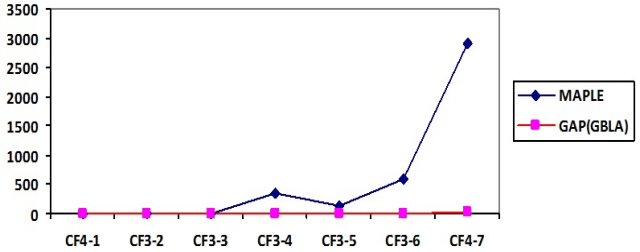

Figura 1. Gráfico con los tiempos de ejecución de BG en GAP y MAPLE.

La Tabla 2 recoge el tiempo que demora en calcular la BG completa hasta el nivel 2 y una BG específica hasta el final para los distintos códigos analizados, para el orden natural de las indeterminadas.

Se puede observar que en los primeros códigos el cálculo de la GC_e hasta el segundo nivel demora más que la BG específica, pues la diferencia de estos tiempos es proporcional a la diferencia entre la cantidad total de líderes de cosetos hasta el nivel dado para la GC_e y la cantidad de cosetos que tenga el código para el cálculo de la BG específica, lo cual no es una cantidad constante. Se observa claramente que al aumentar la longitud de los códigos esta diferencia cambia y corre en menos tiempo la GC_e hasta el nivel fijado. Por lo que en general la GC_e hasta el segundo nivel sería una estructura invariante que se logra calcular en menos tiempo que una BG específica.

Cód	**n**	**k**	*GrC*(**t en seg**)	*G* (**t en seg**)
CF4-1	5	2	1.1	0.5
CF3-2	7	3	1.4	0.5
CF3-3	7	2	1.6	1.8
CF3-4	10	4	7.9	11
CF4-5	8	3	95	30
CF2-6	12	4	0.8	1.2
CF2-7	15	5	11	11.8
CF2-8	18	6	52	60
CF2-9	19	6	9	540
CF2-10	21	6	2023	2138

Tabla 2. Tiempo de ejecución de BG completas y específicas.

Estas comparaciones se pueden observar gráficamente en la Figura 2.

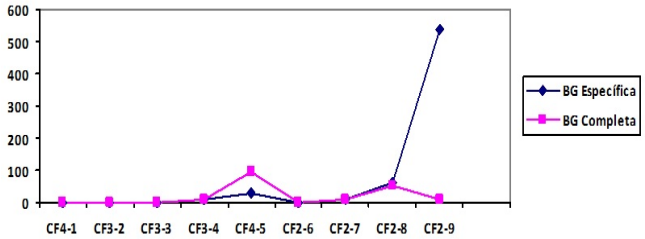

Figura 2. Gráfico con los tiempos de ejecución de BG completas y específicas.

La similitud de tiempo de la última línea en la tabla se debe a que es un código de pocos elementos con respecto a su longitud, lo cual lo determina la dimensión del código sobre la longitud (k/n). En este caso, un algoritmo de cálculo de BG al estilo del Algoritmo de Buchberger analiza menos binomios que pertenecen al ideal y puede correr más rápido que cuando la dimensión es mayor. Sin embargo, la cantidad de cosetos aumenta en la medida que k es más pequeño y hace que un algoritmo GBLA se demore más tiempo. Aun así, la implementación en GAP estuvo ligeramente por debajo de la función del MAPLE.

Las fluctuaciones que se observan en la tabla y en la gráfica, se debe precisamente a la relación entre k y $n-k$, es decir, dimensión y cantidad de cosetos del código. Ahora bien, en los códigos más interesantes y complejos existe un compromiso entre estos dos valores, ocasionando que ambos sean altos y en estos casos una implementación tipo GBLA tiene mayores posibilidades de ser más rápida.

4. Conclusiones

En este trabajo se brinda un nuevo tipo de BG llamada completa y se implementó computacionalmente teniendo en cuenta dos formas: calcular la BG completa hasta el final y calcularla solo hasta un nivel dado. Esta segunda implementación (hasta el segundo nivel) se comparó en cuanto al tiempo de cómputo con una implementación realizada en el trabajo de diploma que antecedió esta investigación que calcula una BG reducida, resultando que el tiempo de cómputo es menor a medida que los códigos aumentan su complejidad.

Se ha demostrado que las BG completa reducida extendida constituyen un invariante para un código, y que éstas BG se pueden obtener por medio de un algoritmo. Esto permitirá que en trabajos futuros se disponga de este invariante para determinar la equivalencia de códigos.

Además, se muestra como las implementaciones realizadas para el cálculo de BG específicas de cualquier código, son más eficientes que la función de cálculo de BG del Sistema de Cálculo Simbólico MAPLE, en la cual se encuentran incorporados algoritmos eficientes para su cálculo.

Referencias

[1] Borges Quintana, M., M. A. Borges Trenard, P. Fitz-Patric y E. Martinez Moro: *On a Gröbner bases and combinatorics for binary codes.* Applicable Algebra in Engineering, Communication and Computing, 19:393–411, 2008.

[2] Borges Quintana, M., M. A. Borges Trenard y E. Martinez Moro: *On a Gröbner bases structure associated to linear codes.* Journal of Discrete Mathematical Sciences and Cryptography, 10:151–191, 2007.

[3] Corbella, I. Marquez: *A Combinatorial Commutative Algebra Approach to Complete Decoding*. Tesis de Doctorado, Universidad de Valladolid, 2013.

[4] Mora, T.: *Solving Polynomial Equation Systems II: Macaulay's Paradigm and Gröbner Technology*. Cambridge University Press, 2005.

[5] Pless, V. y C. Huffman: *Fundamentals of error-correcting codes*. Cambridge University Press, 2003.

[6] Quintana, M. Borges: *Sobre Algunas Técnicas de Bases de Gröbner y sus Aplicaciones*. Tesis de Doctorado, Universidad de Oriente, 2002.