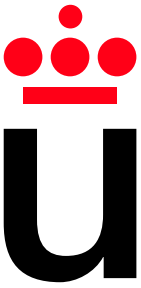

Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Matemáticas

Curso 2022-2023

Trabajo Fin de Grado

BASES DE GROBNER Y SU APLICACIÓN AL PROBLEMA DEL k - COLOREADO Y RESOLUCIÓN DE SUDOKUS

Autor: María Villalba Zapata
Tutor: Ángel Pérez del Pozo

Resumen

En este trabajo se aborda el estudio de las bases de Gröbner y el algoritmo de Buchberger para su obtención, comenzando con un análisis de casos de conjuntos de polinomios simples como son los polinomios lineales o polinomios en una sola variable. Además, se desarrolla una aplicación en Java que implementa todos los métodos y algoritmos presentados, lo que contribuye a una comprensión clara y proporciona ejemplos prácticos.

Posteriormente se analiza la aplicación de las bases de Gröbner al problema de grafos del k-coloreado y a la resolución de sudokus, así como algunas limitaciones del uso del algoritmo de Buchberger para ello.

Palabras clave:

- Bases de Gröbner
- Algoritmo de Buchberger
- Coloreado de grafos
- Sudokus
- Java

Índice de contenidos

Índice de figuras

Índice de códigos

1 Introducción

En este Trabajo de Fin de Grado, nos adentraremos en el mundo de las bases de Gröbner, un concepto fundamental en el álgebra computacional y la geometría algebraica. Estas encuentran aplicaciones en distintos ámbitos, entre ellos el problema del k-coloreado y la resolución de sudokus que estudiaremos en este trabajo.

Para comprender adecuadamente las bases de Gröbner y uno de los algoritmos que las obtienen, el algoritmo de Buchberger, nos basaremos en dos métodos clásicos en el estudio de las ecuaciones lineales y ecuaciones univariables: el método lineal de Gauss-Jordan y el algoritmo de Euler. Estos métodos proporcionarán una base para comprender la teoría detrás de las bases de Gröbner y cómo se relacionan con otros conceptos matemáticos.

A medida que avanzamos en este trabajo, exploraremos las aplicaciones prácticas de estas bases. Nos enfocaremos en dos problemas interesantes: el k-coloreado de grafos y la resolución de sudokus. Ambos problemas se pueden abordar mediante la formulación de sistemas de ecuaciones polinómicas, lo que nos permitirá aprovechar las propiedades y la eficiencia de las bases de Gröbner para resolverlos de manera precisa.

Es importante destacar que, como parte de este Trabajo de Fin de Grado, hemos desarrollado una aplicación en Java que incluye todas las implementaciones y algoritmos mencionados anteriormente. Esta aplicación será una herramienta para comprender los conceptos teóricos y visualizar su aplicación a través de la práctica. Mediante una serie de ejemplos, podremos explorar cómo las bases de Gröbner y el algoritmo de Buchberger pueden resolver problemas complejos de

manera eficiente y efectiva.

En resumen, este Trabajo de Fin de Grado se centra en las bases de Gröbner y el algoritmo de Buchberger, presentando su relación con el método lineal de Gauss-Jordan y el univariable de Euler. También exploraremos las aplicaciones de estas bases en el problema del k-coloreado de grafos y la resolución de sudokus. A través de la aplicación desarrollada en Java, podremos visualizar los resultados y comprender la importancia de estos conceptos en el campo de las matemáticas.

2

Objetivos

- Comprender los casos simples de resolución de sistemas de ecuaciones. Sistemas de polinomios lineales y sistemas de polinomios en una sola variable.
- Comprender qué son las bases de Gröbner y su importancia en las matemáticas.
- Comprender cómo se pueden obtener bases de Gröbner utilizando el algoritmo de Buchberger.
- Implementar una aplicación en Java que incluya los métodos y algoritmos que se traten durante todo el trabajo de manera que ayude a ilustrar la teoría presentada.
- Analizar el uso de bases de Gröbner para resolver el problema del k-coloreado y la resolución de sudokus.

3

Fundamentos teóricos de las bases de Gröbner y el algoritmo de Buchberger

En este capítulo comenzaremos introduciendo los conceptos básicos para conocer las bases de Gröbner. Analizaremos casos de sistemas de ecuaciones más simples para terminar construyendo un algoritmo que obtiene bases para conjuntos de ecuaciones no lineales y en varias variables, este es el algoritmo de Buchberger. Todo ello acompañado de ejemplos prácticos en la aplicación desarrollada en Java que analizaremos más en detalle en el siguiente capítulo.

La referencia principal que hemos utilizado para el desarrollo de este capítulo ha sido [1].

3.1. Introducción

Dado k un cuerpo, consideramos el conjunto de polinomios $f(x_1, ...x_n) \in k[x_1, ..., x_n]$. Para f podemos definir $V(f)$ **variedad definida por** f. Este es el conjunto de soluciones que cumplen $f = 0$.

$$V(f) = \{(a_1, ..., a_n) \in k^n | f(a_1, ..., a_n) = 0\} \subseteq k^n$$

Existen distintos algoritmos para resolver sistemas de ecuaciones no lineales, pero en general, se centran en encontrar una solución aproximada del sistema de

ecuaciones. No tienen en cuenta las propiedades geométricas del conjunto de soluciones, la variedad. Se pueden utilizar distintas descripciones de las variedades. Esto quiere decir que un conjunto de soluciones se puede definir mediante distintos sistemas de ecuaciones y resultar iguales. El proceso de llegar hasta el conjunto de soluciones de un sistema puede ser más o menos complicado dependiendo de cómo definamos el sistema.

Las bases de Gröbner se utilizan para definir generadores de ideales en conjuntos de polinomios multivariables.

Def 1. Dado un anillo A, decimos que I es un ideal de A si se cumple que:

- I es subanillo de A
- I tiene la propiedad de absorción para el producto. $\forall a \in A$, $\forall r \in I$ se cumple que $ar \in I$, $ra \in I$

Dado $k[x]$ un anillo polinómico, siendo k un cuerpo, cualquier ideal I que definamos en el conjunto de polinomios lineales puede ser generado por un único elemento. Dado un conjunto generador $\{f_1, ..., f_s\} \subseteq k[x]$ de I, se puede calcular un solo polinomio q tal que $I = \langle f_1, ...f_s\rangle = \langle q\rangle$.

Además, este polinomio q se obtiene al realizar el máximo común divisor de los polinomios del conjunto generador.

$$g = MCD(f_1, ...f_s)$$

Por tanto, podemos añadir que un polinomio $f \in k[x]$ está en el ideal I si y solo si el resto de dividir f entre q es 0.

Las bases de Gröbner son un análogo a esto que acabamos de presentar. Se trata de encontrar una mejor representación del ideal dado buscando el máximo común divisor de un conjunto de polinomios multivariables y no lineales, de manera que obtengamos así una mejor representación de la variedad.

Para comprender de dónde nace la idea de estas bases que introdujo por primera vez Bruno Buchberger en 1965, primero debemos dar un paso atrás y analizar polinomios en espacios lineales.

3.2. Caso lineal

En primer lugar, vamos a estudiar el conjunto de polinomios de $k[x_1, ..., x_n]$, donde k es un cuerpo y todos los polinomios del conjunto son lineales.

Consideramos el sistema $f_1 = 0, f_2 = 0, ...f_n = 0$, con f_i lineal.

Para estos casos, el método que se utiliza para encontrar un conjunto de mejores generadores del ideal dado por los polinomios que componen el sistema es el de Gauss-Jordan. Consiste en ir modificando la matriz asociada al sistema hasta conseguir que todos los elementos bajo la diagonal principal sean 0.

Por ejemplo, dado el sistema formado por f_1, f_2 y f_3

$$\begin{aligned} f_1 &= x + y + 2z = 0 \\ f_2 &= 2x + 3y - z = 0 \\ f_3 &= 3x + 4y + z = 0 \end{aligned}$$

Tomamos su matriz asociada y vamos operando sobre ella para conseguir que los elementos bajo la diagonal principal se hagan 0.

$$\begin{bmatrix} 1 & 1 & 2 \\ 2 & 3 & -1 \\ 3 & 4 & 1 \end{bmatrix} \tag{3.1}$$

$\xrightarrow{f_2 - 2f_1}$

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & -5 \\ 3 & 4 & 1 \end{bmatrix} \tag{3.2}$$

$\xrightarrow{f_3 - 3f_1}$

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & -5 \\ 0 & 1 & -5 \end{bmatrix} \tag{3.3}$$

$\xrightarrow{f_3 - f_2}$

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & -5 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

El sistema obtenido mediante la aplicaciónd de Gauss-Jordan es el siguiente:

$$f_1 = x + y + 2z = 0$$
$$f_4 = y - 5z = 0$$

Habiendo llegado a esta reducción, podemos afirmar que $\langle f_1, f_2, f_3\rangle = \langle f_1, f_4\rangle$. Mediante el Método de Gauss-Jordan hemos encontrado una mejor representación del ideal que contiene a los tres polinomios.

3.3. Caso univariable

Vamos a tener en cuenta ahora solo el conjunto de polinomios en una variable, $k[x]$. Sea f un polinomio de $k[x]$, se definen los siguientes conceptos:

- Se define el **grado** $deg(f)$ como el mayor exponente al que x aparece elevado en f.
- Se define el **término director** $lt(f)$ como el término de f con mayor grado.
- Se define el **coeficiente del término director** $lc(f)$ como el coeficiente del término director de f.
- Se define el **producto del término director** $lp(f)$ como las variables del término director de f.

De manera genérica, dado $f = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 + a_0$, con $a_0, ..., a_n \in k$ y $a_n \neq 0$, donde $x_n > x_{n-1} > ... > x_1$ $deg(f) = n$, $lt(f) = a_n x^n$, $lc(f) = a_n$ y $lo(f) = x_n$

Para este conjunto de polinomios, la manera de responder a la cuestión que se planteaba en la introducción sobre encontrar mejores representaciones del ideal dado, es a través del Algoritmo Euclidiano.

La base de este algoritmo es el uso de la división polinómica que vamos a ver a través del siguiente ejemplo.

Ejemplo 1. Dados $f(x) = x^4 + 4x^3 - 5x^2 - 22x - 24$ y $g(x) = x^2 + 3x + 2$, donde $f, g \in \mathbb{Q}[x]$. Dividimos f entre g y vamos a obtener con ello el cociente $x^2 + x - 10$ y el resto $6x - 4$. Para obtener estos resultados los pasos que se han seguido son los siguientes:

- **Paso 0.** Para poder realizar la división, tenemos que comprobar que el grado de $f(x)$ sea superior o igual al de $g(x)$. Como $lt(f) = (x^4) > (x^2) = lt(g)$, se puede comenzar la división.

- **Paso 1.** Se divide $lt(f)$, entre $lt(g)$. $\frac{x^4}{x^2} = x^2$
- **Paso 2.** Se multiplica el divisor obtenido por $g(x)$.

$$(x^2)g(x) = x^2(x^2 + 3x + 2) = x^4 + 3x^3 + 2x^2$$

- **Paso 3.** Se resta el resultado obtenido en el paso 2 a f.

$$h(x) = f(x) - (x^2)(x^2 + 3x + 2) = x^3 - 7x^2 - 22x - 24$$

- Se comprueba el paso 0 de nuevo y se repiten los pasos 1 y 2.

$$Paso\ 0: \quad \frac{x^3}{x^2} = x$$

$$Paso\ 1: \quad x(x^2 + 3x + 2) = x^3 + 3x^2 + 2x$$

$$\begin{aligned} Paso\ 2: \quad & h(x) = f(x) - (x^3 + 3x^2 + 2x) = (x^3 - 7x^2 - 22x - 24) \\ & - (x^3 + 3x^2 + 2x) = -10x^2 - 24x - 24 \end{aligned}$$

- Como en este caso el grado de $lt(f) = lt(g)$, se puede seguir. Se repiten de nuevo los pasos 1 y 2.

$$Paso\ 0: \quad \frac{-10x^2}{x^2} = -10$$

$$Paso\ 1: \quad -10(x^2 + 3x + 2) = -10x^2 - 30x - 20$$

$$\begin{aligned} Paso\ 2: \quad & h(x) = f(x) + 10(x^2 + 3x + 2) = (-10x^2 - 24x - 24) \\ & + (10x^2 + 30x + 20) = 6x - 4 \end{aligned}$$

- En este caso, el grado de $lt(f)$ es 1, mientras que el de $lt(g)$ es 2, por tanto, ha terminado la división. El resto obtenido es $6x - 4$ y el cociente de la división $x^2 + x - 10$, por tanto, tenemos que $f = (x^2 + x - 10)$ g $+ 6x - 4$.

Introduciendo los polinomios del Ejemplo 1 en la aplicación de Java se obtiene el siguiente resultado:

```
Division entre x_1^4 + 4x_1^3 -5x_1^2 -22x_1 -24
y x_1^2 + 3x_1 + 2

PASO 1
q = x_1^2
r =  -24 -7x_1^2 -22x_1 + x_1^3

PASO 2
q = x_1 + x_1^2
r =  -10x_1^2 -24x_1 -24

PASO 3
q = x_1 + x_1^2 -10
r =  -4 + 6x_1


COCIENTE: x_1 + x_1^2 -10
RESTO:  -4 + 6x_1
```

Código 3.1: Ejemplo division univariable Java

Como se había calculado en el propio ejemplo, el cociente obtenido es $x^2 + x - 10$ con resto $6x - 4$.

Analizando los pasos que se han dado hasta llegar al resultado, lo que se ha ido haciendo en cada iteración ha sido ir buscando un término que multiplicado por g, hiciera que $lt(g)$ por este término cancelara $lt(f)$.

El término que hemos ido calculando para cancelarlos ha sido $\frac{lt(f)}{lt(g)}$, por tanto, la operación que se realizaba en cada iteración para obtener el nuevo polinomio, $h(x)$, ha sido $h = f - \frac{lt(f)}{lt(g)}g$

A h le llamaremos a partir de ahora reducción de f por g, denotándolo

$$f \xrightarrow{g} h$$

En el ejemplo, esta reducción se ha repetido dos veces. Primero reduciendo f a otro polinomio, h, y finalmente al resto r.

$$f \xrightarrow{g} h \xrightarrow{g} r$$

Esto se puede simplificar sin importar el número de pasos intermedios y la notación qudaría así:

Algoritmo 1 *Algoritmo división univariable*

ENTRADA: $f, g \in k[x]$ con $g \neq 0$

SALIDA: q, r tal que $f = qg + r$ y $r = 0$ o $deg(r) < deg(g)$

INICIALIZACIÓN: $q := 0; r := f$

1: **while** $r \neq 0$ & $deg(g) \leq deg(r)$ **do**
2: $\quad q := q + \frac{lt(r)}{lt(g)}$
3: $\quad r := r - \frac{lt(r)}{lt(g)} g$
4: **end while**

$$f \xrightarrow{g}_{+} r$$

Teorema 1. *Sea g un polinomio distinto de 0 en $k[x]$. Entonces para cualquier $f \in k[x]$, existen q y r en $k[x]$ tal que $f = qg + r$, con $r = 0$ o $deg(r) < deg(g)$. Además r y q son únicos.*

Para encontrar los polinomios q y r se presenta el siguiente algoritmo de división para conjuntos de polinomios univariables que puede encontrarse en el Algoritmo 1. Está basado en las iteraciones del Ejemplo 1 y el Teorema 1 . El resultado se habrá encontrado cuando se cumpla una de las dos condiciones, o bien que $r = 0$ o que el grado de r sea menor que el de g. Además dentro de cada iteración se va sumando al cociente q el término obtenido de $\frac{lt(r)}{lt(g)}$ y el resto se va calculando restando g por ese término a r de manera que se vaya reduciendo el grado de r en cada paso eliminando su término director.

Teorema 2. *Todo ideal I de $k[x]$ está generado por un elemento (a un ideal generado por un solo elemento se le llama **ideal principal**).*

Dem. Vamos a demostrar por reducción al absurdo el teorema anterior.

Dado I ideal de $k[x]$, tomamos $g \in I$ tal que $g \neq 0$ y $n = deg(g)$, siendo n el menor grado posible. Usando el teorema 1, para cualquier f del ideal, existen r y q tal que $f = qg + r$, con $r = 0$ o con el grado de r estrictamente menor que el de g. Si suponemos que $r \neq 0$, entonces $r = f - qg$, $r \in I$ porque f y g también lo están. Pero como g tiene el menor grado posible entre todos los polinomios del ideal, r debe ser 0. Por tanto, $f = qg$ y $I \subseteq \langle g \rangle$. Además, como $g \in I$, por absorción, todos sus múltiplos también estarán en I, por lo que $\langle g \rangle \subseteq I$ ■

A continuación, vamos a estudiar cómo encontrar un elemento generador de un ideal principal. Para ello suponemos primero un ideal $I \subseteq k[x]$ generado por

Algoritmo 2 *Algoritmo Euclidiano*

ENTRADA: $f_1, f_2 \in k[x]$ con $f_1, f_2 \neq 0$

SALIDA: $f = MCD(f_1, f_2)$

INICIALIZACIÓN: $f = f_1, g = f_2$

1: **while** $g \neq 0$ **do**
2: $f \xrightarrow{g}_+ r$, siendo r el resto de la división de f entre g
3: $f := g$
4: $g := r$
5: **end while**
6: $f := \frac{1}{lc(f)} f$

dos polinomios donde alguno de los dos es distinto de 0. El máximo común divisor de f_1 y f_2, $MCD(f_1, f_2)$ es un polinomio g que cumple lo siguiente:

- g divide a f_1 y f_2.
- si $h \in k[x]$ divide a f_1 y f_2, entonces h también divide a g.
- $lc(g) = 1$, es decir, el coeficiente de su término director es 1.

Prop 1. Dados $f_1, f_2 \in k[x]$ donde al menos uno de los dos es distinto de 0. Entonces existe $MCD(f_1, f_2)$ y $\langle f_1, f_2 \rangle = \langle MCD(f_1, f_2) \rangle$

Por tanto, para encontrar el polinomio generador de un ideal principal de $k[x]$ debemos encontrar el máximo común divisor de sus polinomios generadores. Para ello se va a utilizar el Algoritmo de Euclides que se presenta en el Algoritmo A.2. En él se introducen dos polinomios $f_1(x), f_2(x) \in k[x]$ univariables obteniendo su máximo común divisor. Sobre los dos polinomios, de manera iterativa, se va calculando su división mediante el Algoritmo 1, y se reemplazan los polinomios de manera que f_2 pasa a ser el nuevo dividendo y el resto obtenido r será el nuevo divisor. Una vez que se haya obtenido resto 0 al realizar la división, habremos obtenido el $MCD(f_1, f_2)$, a falta de dividir por su coeficiente director para hacerlo mónico.

Aplicando ahora el Algoritmo de Euler en Java sobre el Ejemplo 1, obtenemos que, según el resultado 3.3, el máximo común divisor de $f(x) = x^4 + 4x^3 - 5x^2 - 22x - 24$ y $g(x) = x^2 + 3x + 2$ es 1.

```
El maximo comun divisor entre
```

```
x_1^4 + 4x_1^3 -5x_1^2 -22x_1 -24
y x_1^2 + 3x_1 + 2

Paso 1
Cociente: x_1 + x_1^2 -10
Resto:  -4 + 6x_1
Nueva f: x_1^2 + 2 + 3x_1
Nueva g:  -4 + 6x_1

Paso 2
Cociente: 1/6x_1 + 11/18
Resto: 40/9
Nueva f:  -4 + 6x_1
Nueva g: 40/9

Paso 3
Cociente:  -9/10 + 27/20x_1
Resto:
Nueva f: 40/9
Nueva g:


MCD: 1
```

Código 3.2: Ejemplo Algoritmo Euler Java 1

Ejemplo 2. Veamos ahora otro ejemplo donde el máximo común divisor de dos polinomios sea distinto de 1. Sea $f(x) = 3x^2 - 6x$ y sea $g(x) = 4x^3 - 8x^2$, calculamos su MCD. En este caso el resultado al introducir estos polinomios en la aplicación de Java es el que se presenta en 3.3.

```
El maximo comun divisor entre 3x_1^2 -6x_1 y 4x_1^3 -8x_1^2

Paso 1
Cociente:
Resto: 3x_1^2 -6x_1
Nueva f: 4x_1^3 -8x_1^2
Nueva g: 3x_1^2 -6x_1

Paso 2
Cociente: 4/3x_1
Resto:
Nueva f: 3x_1^2 -6x_1
Nueva g:
```

```
MCD:  x_1^2 -2x_1
```

Código 3.3: Ejemplo Algoritmo Euler Java 2

3.4. Caso multivariable

Para estudiar ahora los conjuntos de polinomios multivariables vamos a basarnos en los términos y algoritmos que hemos visto en las secciones anteriores.

3.4.1. Orden terminológico

Hasta el momento no se ha presentado problema a la hora de ordenar los monomios de un polinomio. En el caso de tomar polinomios de un espacio univariable, se pueden ordenar según el grado de éste.

$$1 < x < x^2 < x^3 < ...$$

En el caso lineal, como máximo cada monomio puede tener solo una variable con grado 1, por tanto, para definir un orden basta con fijar el orden de las variables que existan en el espacio de polinomios sobre el que trabajemos.

$$\begin{aligned} Fijando\ x > y > z \\ 1 < z < y < x \end{aligned}$$

En el caso de espacios de polinomios multivariables no lineales, se pueden definir distintas formas de establecer un orden entre sus monomios.

- **Orden lexicográfico por grado: *deglex***

  Para comparar dos monomios de esta manera, primero se debe fijar el orden de las variables implicadas. Dado el conjunto de variables $\{x_1, x_2, ..., x_n\}$, se debe definir un orden entre ellas, por ejemplo, $x_1 > x_2 > ... > x_n$

  Una vez que se ha fijado el orden entre las variables, se revisa el grado total de ambos monomios, es decir, la suma de los exponentes de las variables que aparecen en él. El orden de los monomios vendrá marcado por su grado total en caso de ser de grados distintos.

  En caso de coincidir en el grado, se va analizando las variables que aparecen de mayor a menor (primero x_1, luego x_2, etc.) y el exponente que tienen en

cada uno de los monomios. Cuando se encuentra que una de ellas tiene un exponente superior en uno de los monomios, este será el mayor.

Se muestra un ejemplo del orden *deglex* sobre el conjunto $\{x, y\}$ con $x > y$.

$$1 < y < x < y^2 < xy < x^2 < y^3 < y^2x < yx^2 < x^3 < ...$$

- **Orden lexicográfico: *lex***

  En esta segunda forma de ordenar monomios se vuelve a fijar el orden de las variables. Tomaremos el mismo ejemplo que antes con $x_1 > x_2 > ... > x_n$.

  Ahora no hay que esudiar el grado en conjunto de cada monomio, si no que hay que ir comparando el exponente de cada variable por separado. Por ejemplo, si tenemos los dos monomios $x_1x_2^2$ y $x_1x_2x_3^2$, el segundo tiene mayor grado en su totalidad. Si lo estudiaramos a través del orden *deglex*, diríamos que el segundo monomio es mayor. En cambio, con el orden lexicográfico se van estudiando los exponentes de cada variable una a una.

  Como x_1 tiene exponente 1 en ambos, pasamos a analizar x_2. Esta variable tiene exponente 2 en el primer monomio, mientras que en el primero es 1. Por tanto, ya podemos decir que mediante el orden *lex*, el primer monomio es mayor.

  Se muestra ahora un ejemplo del orden *lex* sobre el conjunto $\{x, y\}$ con $x > y$.

$$1 < y < y^2 < ... < x < yx < y^2x < y^3x < ... < x^2 < yx^2 < y^2x^2 < ...$$

Siempre que se realicemos operaciones sobre un conjunto no lineal y con varias variables, será primero necesario fijar el orden de las variables implicadas y qué tipo de orden se va a utilizar. Los resultados obtenidos diferirán según el que se escoja tal y como hemos visto en en ejemplo de los monomios $x_1x_2^2$ y $x_1x_2x_3^2$.

De forma análoga a como hemos definido en el caso anterior vamos a presentar los siguientes términos aplicados a polinomios multivariables y no lineales.

- Se define el **término director** $lt(f)$ como el término de f con mayor grado según el orden elegido.
- Se define el **grado** $deg(f)$ como el grado total del término director.
- Se define el **coeficiente del término director** $lc(f)$ como el coeficiente del término director de f.
- Se define el **producto del término director** $lp(f)$ como las variables del término director de f.

Ejemplo 3. Veamos un ejemplo donde se apliquemos todos estos conceptos. Dado el polinomio $f(x) = 7x^2y + 3xy^3 - y^2$, donde $x > y$ con el orden terminológico lex. Tendríamos, $lt(f) = 7x^2y$, $deg(f) = 3$, $lc(f) = 7$ y $lp(f) = x^2y$.

En caso de haber tomado el orden $deglex$, tendríamos $lt(f) = 3xy^3$, $deg(f) = 4$, $lc(f) = 3$ y $lp(f) = xy^3$.

3.4.2. Algoritmo de división

En primer lugar, se va a definir un algoritmo de división para conjuntos de polinomios multivariables. Para ello, se va a tomar la idea que utilizan los algoritmos que se han visto en secciones anteriores. Dado un polinomio f, para dividirlo por $f_1, ..., f_s$ se van a ir cancelando términos de f usando los términos directores de $f_1, ..., f_s$ y el proceso continuará hasta que no se puedan reducir más. Se presenta en Algoritmo 3.

Def 2. Dadas $f, g, h \in k[x_1, ...x_n]$ con $g \neq 0$, decimos que f es reducido a h módulo g en un solo paso,

$$f \xrightarrow{g} h$$

si y solo si $lp(g)$ divide al término $X \neq 0$ que aparece en f y

$$h = f - \frac{X}{lt(g)}g$$

Def 3. Dados $f, h, f_1, ..., f_s$ polinomios de $k[x_1, ..., x_n]$, con $f_i \neq 0$ $(1 \leq i \leq s)$, y $F = \{f_1, ..., f_s\}$. Decimos que f reduce a h módulo F, denominado

$$f \xrightarrow{F}_+ h$$

si y solo si existe una secuencia de índices $i_1, ..., i_t \in \{1, ..., s\}$ y una secuencia de polinomios $h_1, ..., h_{t-1} \in k[x_1, ..., x_n]$ tal que

$$f \xrightarrow{f_{i_1}} h_1 \xrightarrow{f_{i_2}} h_2 \xrightarrow{f_{i_3}} ... \xrightarrow{f_{i_{t-1}}} h_{t-1} \xrightarrow{f_{i_t}} h$$

Def 4. Decimos que un polinomio r es irreducible con respecto a un conjunto de polinomios distintos de 0, $F = \{f_1, ..., f_s\}$ si se cumple una de las dos:

- $r = 0$

- no existe ningún término en r divisible por ningún $lp(f_i)$, $i = \{1, ..., s\}$. Es decir, r no es reducible módulo F.

Además, cuando r es irreducible módulo F y $f \xrightarrow{F}_{+} r$, llamamos a r resto de f con respecto a F.

Ejemplo 4. Vemos a continuación un ejemplo donde aplicamos el Algoritmo 3. Sean $f(x, y) = x^3y^3 + 2y^2$, $f_1(x, y) = 2xy^2 + 3x + 4y^2$, $f_2(x, y) = y^2 - 2y - 2 \in \mathbb{Q}$, calculamos la división de f sobre el conjunto $F = \{f_1, f_2\}$.

```
Division de: x_1^3x_2^3 + 2x_2^2
entre: {2x_1x_2^2 + 3x_1 + 4x_2^2, x_2^2 -2x_2 -2}

PASO 1
Existe lp(f1)= x_1x_2^2 que divide a lp(h)= x_1^3x_2^3
h = 2x_2^2 -2x_1^2x_2^3

PASO 2
No existe lp(f_i) que divida a lp(h)
r =
h = 2x_2^2 -2x_1^2x_2^3

PASO 3
Existe lp(f1)= x_1x_2^2 que divide a lp(h)= x_1^2x_2^3
h = 3x_1^2x_2 + 4x_1x_2^3 + 2x_2^2

PASO 4
No existe lp(f_i) que divida a lp(h)
r = 3x_1^2x_2
h = 4x_1x_2^3 + 2x_2^2

PASO 5
Existe lp(f1)= x_1x_2^2 que divide a lp(h)= x_1x_2^3
h =  -6x_1x_2 -8x_2^3 + 2x_2^2

PASO 6
No existe lp(f_i) que divida a lp(h)
r = 3x_1^2x_2 -6x_1x_2
h =  -8x_2^3 + 2x_2^2

PASO 7
Existe lp(f2)= x_2^2 que divide a lp(h)= x_2^3
h =  -14x_2^2 -16x_2
```

Algoritmo 3 *Algoritmo de división multivariable*

INPUT: $f, f_1, ..., f_s \in k[x_1, ...x_n]$ *con* $f_i \neq 0$ $(1 \leq i \leq s)$

OUTPUT: $u_1, ..., u_s, r$ tal que $f = u_1 f_1 + ... + u_s f_s + r$ y r es irreducible con respecto a $f_1, ...f_s$ y $max(lp(u_1)lp(f_1), ..., lp(u_s)lp(f_s), lp(r)) = lp(f)$

INICIALIZACIÓN: $u_1 := 0, ..., u_s := 0, r := 0, h := f$

1: **while** $h \neq 0$ **do**
2: **if** existe i tal que $lp(f_i)$ divide a $lp(h)$ **then**
3: tomamos $i \mid lp(f_i)$ divide a $lp(h)$
4: $u_i := u_i + \frac{lt(h)}{lt(f_i)}$
5: $h := h - \frac{lt(h)}{lt(f_i)} f_i$
6: **else**
7: $r := r + lt(h)$
8: $h := h - lt(h)$
9: **end if**
10: **end while**

```
PASO 8
Existe lp(f2)= x_2^2 que divide a lp(h)= x_2^2
h =  -44x_2 -28

PASO 9
No existe lp(f_i) que divida a lp(h)
r = 3x_1^2x_2 -6x_1x_2 -44x_2
h =  -28

PASO 10
No existe lp(f_i) que divida a lp(h)
r = 3x_1^2x_2 -6x_1x_2 -44x_2 -28
h =

Obtenemos:
u_1 = 1/2x_1^2x_2 + 2x_2 -x_1x_2
u_2 = -8x_2 -14
r = 3x_1^2x_2 -6x_1x_2 -44x_2 -28
```

Código 3.4: Ejemplo Algoritmo Division Multivariable

Teorema 3. *Dado un conjunto de polinomios distintos de vacío,* $F = \{f_1, ...f_s\}$ *y* $f \in k[x_1, ...x_n]$*, tras aplicar el algoritmo de división multivariable visto en el Algoritmo 3, se obtienen polinomios* $u_1, ..., u_s, r \in k[x_1, ..., x_n]$ *tales que*

$$f = u_1 f_1 + ... + u_s f_s + r,$$

con r irreducible con respecto a F y

$$lp(f) = max(max_{i \leq i \leq s}(lp(u_i)lp(f_i)), lp(r))$$

Al igual que en la sección anterior, vamos a tratar ahora de encontrar el mejor conjunto generador de un ideal, pero ahora para polinomios multivariables.

3.4.3. Bases de Gröbner

Def 5. Dado un conjunto de polinomios distintos de vacío $G = \{g_1, ..., g_t\}$ contenidos en un ideal I,

G es una **Base de Gröbner** del ideal $I \longleftrightarrow \forall f \in I$ tal que $f \neq 0$, $\exists\, i \in \{1, ..., t\}$ tal que $lp(g_i)$ divide a $lp(f)$.

Dicho de otra forma, si G es una Base de Gröbner, entonces no existen polinomios distintos de vacío en I irreducibles con respecto de G.

Vamos a construir ahora el algoritmo que las obtiene.

En la Definición 5 se plantea que G es Base de Gröbner si y solo si para todo $f \in I$, existe $i \in \{i, ..., s\}$ tal que $lp(f_i)$ divide a $lp(f)$. Por tanto, G no será Base de Gröbner si encontramos $f \in I$ donde para ningún $i \in \{i, ..., s\}$ su $lp(f_i)$ divida al de f.

Def 6. Dadas $f, g \neq 0$, $f, g \in k[x_1, ..., x_n]$. Sea $L = mcm(lp(f), lp(g))$. Se denomina S-Polinomio de f y g al siguiente polinomio:

$$S(f, g) = \frac{L}{lt(f)} f - \frac{L}{lt(g)} g$$

Ejemplo 5. Sean $f = y^2x + y$, $g = y - x \in Q[x, y]$ con $y > x$, donde $lt(f) = y^2x, lt(g) = y$. En este caso, $L = mcm(lp(f), lp(g)) = y^2x$ y $S = \frac{L}{lt(f)} f - \frac{L}{lt(g)} g = y + yx^2$

Introduciendo este mismo ejemplo en la aplicación de Java, se obtiene el siguiente output.

```
El mcm de: x_1^2x_2 + x_1 y x_1^2x_2 + x_1 es:

x_1^2x_2
```

Algoritmo 4 *Algoritmo de Buchberger*

INPUT: $F = \{f_1, ..., f_s\} \subseteq\ k[x_1, ...x_n]$ con $f_i \neq\ 0, (1 \leq\ i \leq\ s)$

OUTPUT: $G = \{g_1, ..., g_t\}$, Base de Gröbner para $\langle f_1, ..., f_s \rangle$

INICIALIZACIÓN: $G := F, \Phi := \{\{f_i, f_j\} \mid f_i \neq\ f_j \in\ G\}$

1: **while** $\Phi \neq \emptyset$ **do**
2: elegir cualquier subconjunto $\{f, g\} \in \Phi$
3: $\Phi := \Phi - \{f, g\}$
4: $S(f, g) \xrightarrow{G}_+ h$, donde h es irreducible con respecto a G

5: **if** $h \neq 0$ **then**
6: $\Phi := \Phi \cup \{\{u, h\} \mid \forall u \in G\}$
7: $G := G \cup \{h\}$
8: **end if**

9: **end while**

```
El S-Polinomio de: x_1^2x_2 + x_1
y x_1^2x_2 + x_1 es:

x_1 + x_1x_2^2
```

Código 3.5: Ejemplo cálculo mcm y S-Polinomio en Java

Teorema 4. ***Teorema de Buchberger***. *Sea $G = \{g_1, ..., g_t\}$ un conjunto de polinomios distintos de vacío de $k[x_1, ..., x_n]$. G es una base de Gröbner para el ideal $I = \langle g_1, ..., g_t \rangle$ si y solo si para todo $i \neq j$,*

$$S(g_i, g_j) \xrightarrow{G}_+ 0$$

Este teorema proporciona una estrategia para obtener bases de Gröbner que se presenta en el Algoritmo 4. Sobre el conjunto de polinomios dado, se realizan iteraciones donde se toma uno de los pares de G del cual se calcula su S-polinomio. Este se reduce sobre el conjunto G y si el resto obtenido es distinto de 0, se añade al conjunto de polinomios de la base resultante. Esto se repite hasta que se hayan procesado todos los pares existentes en G. El conjunto resultante formado por los restos de las divisiones distintos de 0, será la base de Gröbner obtenida.

A continuación, se muestra un ejemplo de aplicación de este algoritmo.

Ejemplo 6. Sean $f_1 = x_1^2 + x_1x_2 + x_2^2$, $f_2 = x_1 + x_2$ y $f_3 = x_1 \in \mathbb{Q}[x_1, x_2]$ con

$x_1 > x_2$ tomando el orden lexicográfico. Sea $I = \langle f_1, f_2, f_3 \rangle$ vamos a calcular la base de Gröbner sobre I.

Llamamos G al espacio formado por f_1 y f_2. Φ será el conjunto que contiene a los subconjuntos de pares formados por los tres polinomios, $\phi = \{\{f_1, f_2\}, \{f_1, f_3\}, \{f_2, f_3\}\}$.

PASO 1. En el primer paso eliminamos el par $\{f_1, f_2\}$ de Φ.

Calculando su S-Polinomio, se obtiene que $S = x_2^2$. Reduciendo este sobre el conjunto G se obtiene el resto $x_2^2 = f_4$. Por tanto, ahora $G = \{f_1, f_2, f_3, f_4\}$, mientras que $\phi = \{\{f_1, f_3\}, \{f_2, f_3\}, \{f_1, f_4\}, \{f_2, f_4\}, \{f_3, f_4\}\}$

PASO 2. Eliminando ahora el par $\{f_1, f_3\}$, se obtiene el S-Polinomio $x_2^2 + x_1 x_2$ que al reducirlo sobre G obtenemos de resto 0, por tanto, no se añade nada a la base.

PASO 3. En este tercer paso se elige el siguiente par a eliminar de ϕ, $\{f_2, f_3\}$. En este caso al reducir el S-Polinomio $S = x_2$ sobre G se obtiene el resto $x_2 = f_5$ que se añade a la base y los respectivos pares se añaden al conjunto Φ.

Si se continúa ejecutando este mismo proceso hasta el paso 10º, se obtendrá el mismo resultado, pero siempre el resto será 0. Al no obtener ningún nuevo polinomio en el resto de pasos, la base de Gröbner obtenida para este ejemplo será el conjunto formado por $\{f_1, f_2, f_3, f_4, f_5\}$.

Se muestra a continuación el Ejemplo 6 introducido en la aplicación de Java.

```
Vamos a calcular la base de Grobner
del siguiente conjunto de polinomios:

x_1^2 + x_1x_2 + x_2^2
x_1 + x_2
x_1

PASO 1
Eliminamos el par {x_1^2 + x_1x_2 + x_2^2, x_1 + x_2}
El S-Polinomio del par elegido es x_2^2

El resto de la division es x_2^2
Los polinomios que estan en G son:
x_1^2 + x_2^2 + x_1x_2
x_1 + x_2
x_1
x_2^2

PASO 2
Eliminamos el par {x_1^2 + x_2^2 + x_1x_2 , x_1}
```

```
El S-Polinomio del par elegido es x_2^2 + x_1x_2
El resto de la division es 0

PASO 3
Eliminamos el par {x_1 + x_2 , x_1}
El S-Polinomio del par elegido es x_2
El resto de la division es x_2
Los polinomios que estan en G son:
x_1^2 + x_2^2 + x_1x_2
x_1 + x_2
x_1
x_2^2
x_2

PASO 4
Eliminamos el par {x_1^2 + x_2^2 + x_1x_2 , x_2^2}
El S-Polinomio del par elegido es x_1x_2^3 + x_2^4
El resto de la division es 0

PASO 5
Eliminamos el par {x_1 + x_2 , x_2^2}
El S-Polinomio del par elegido es x_2^3
El resto de la division es 0

PASO 6
5
Eliminamos el par {x_1 , x_2^2}

PASO 7
Eliminamos el par {x_1^2 + x_2^2 + x_1x_2 , x_2}
El S-Polinomio del par elegido es x_2^3 + x_1x_2^2
El resto de la division es 0

PASO 8
Eliminamos el par {x_1 + x_2 , x_2}
El S-Polinomio del par elegido es x_2^2
El resto de la division es 0

PASO 9
Eliminamos el par {x_1 , x_2}
El S-Polinomio del par elegido es 0

PASO 10
Eliminamos el par {x_2^2, x_2}
```

```
El S–Polinomio del par elegido es 0

La base obtenida G es:
x_1^2 + x_2^2 + x_1x_2
x_1 + x_2
x_1
x_2^2
x_2
```

Código 3.6: Ejemplo Algoritmo de Buchberger en Java

Bases de Gröbner reducidas

Con el algoritmo de Buchberger proporcionado en 4 es posible obtener distintas bases de Gröbner para un mismo conjunto de polinomios. Por una parte, el orden de los polinomios en G influirá en el algoitmo de división a la hora de encontrar un polinomio cuyo lp divida al lp del polinomio principal.

Además, el orden de eliminación de los pares de polinomios también puede modificar la base resultante.

Por ejemplo, tomando el Ejemplo 6, si siempre seguimos la estructura de eliminación $\{1,2\},\{1,3\},\{2,3\}$ pero llamamos ahora a cada polinomio de manera distinta: $f_1 = x_1$, $f_2 = x_1 + x_2$ y $f_3 = x_1^2 + x_1x_2 + x_2^2$, obtendríamos la base siguiente: $\{x_1, x_2 + x_1, x_2^2 + x_1x_2 + x_1^2, -x_2\}$, que como vemos, es distinta a la obtenida con el orden anterior.

Es posible obtener siempre una única base de Gröbner sobre un conjunto dado de polinomios, esto se consigue aplicando un algoritmo de reducción sobre la base obtenida.

Def 7. Una base de Gröbner $G = \{g_1, g_2, ..., g_t\}$ está **reducida** si para todo i, $lc(g_i) = 1$ y g_i es irreducible con respecto a $G - \{g_i\}$. Es decir, no existe ningún $lp(g_j)$ que divida a $lp(g_i)$ para cualquier $i \neq j$.

Dada esta definición, proponemos el siguiente algoritmo (Algoritmo 5). En él se va analizando cada $g_i \in G$, siendo G base de Gröbner. Se reduce este polinomio sobre el conjunto de polinomios $\{g_j : j \neq i_j^g \in G\}$ y se añade al conjunto resultante el resto obtenido multiplicado por su coeficiente director para hacerlo mónico. De esta manera, obtenemos un conjunto reducido de polinomios irreducibles con respecto a G - $\{g_i\}$ mónicos.

Utilizando este algoritmo, vamos a reducir la base de Gröbner obtenida en el Ejemplo 6.

Algoritmo 5 *Algoritmo de reducción*

INPUT: $G = \{g_1, ..., g_t\}$ base de Gröbner para el ideal I.

OUTPUT: $H = \{h_1, ..., h_t\}$, Base de Gröbner reducida para el ideal I.

INICIALIZACIÓN: $R := G, H := \emptyset$

1: **for all** g en G **do**
2: $R = R - \{g\}$
3: $g \xrightarrow{R}_+ h$, donde h es irreducible con respecto de R
4: $h = \frac{h}{lc(h)}$
5: $H = H \cup \{h\}$
6: **end for**

```
EL POLINOMIO ELIMINADO ES: x_1^2 + x_2^2 + x_1x_2

H:

R:
x_1 + x_2
x_1
1x_2
1x_2^2

EL POLINOMIO ELIMINADO ES: x_1 + x_2

H:

R:
x_1
1x_2
1x_2^2

EL POLINOMIO ELIMINADO ES: x_1

H:

R:
1x_2
1x_2^2
El resto de la division es x_1
```

```
EL POLINOMIO ELIMINADO ES:1x_2

H:
x_1

R:
1x_2^2
x_1
El resto de la division es x_2

EL POLINOMIO ELIMINADO ES:1x_2^2

H:
x_1
1x_2

R:
x_1
1x_2

La base reducida R es:
x_1
1x_2
```

Código 3.7: Ejemplo Algoritmo de Reducción en Java

Hemos mencionado al principio de esta sección la importancia de las bases reducidas, ya que obtienen una única base de Gröbner para cada conjunto de polinomios, sea cual sea el método elegido para obtenerla. El ejemplo en Java que acabamos de mostrar es el que ha reducido la base obtenida utilizando el orden de eliminación de pares del Ejemplo 6. Si calculábamos la base de Gröbner alterando el orden de eliminación de los pares, ésta era distinta, $\{x_1, x_2 + x_1, x_2^2 + x_1x_2 + x_1^2, -x_2\}$.

Veamos ahora en el resultado 3.4.3, que al reducir esta segunda base, la base reducida que obtenemos es la misma, por tanto, aplicar este segundo algoritmo sobre las bases de Gröbner que obtengamos nos proporcionará unicidad y espacios aún más fáciles sobre los que trabajar ya que serán lo más pequeños posibles.

```
EL POLINOMIO ELIMINADO ES:x_1

H:

R:
x_1 + x_2
1x_2^2 + x_1x_2 + x_1^2
```

```
 -1x_2

EL POLINOMIO ELIMINADO ES: x_1 + x_2

H:

R:
x_1x_2 + x_2^2 + x_1^2
 -1x_2
El resto de la division es x_1

EL POLINOMIO ELIMINADO ES:1x_2^2 + x_1x_2 + x_1^2

H:
x_1

R:
 -1x_2
x_1

EL POLINOMIO ELIMINADO ES: -1x_2

H:
x_1

R:
x_1
El resto de la division es  -1x_2

La base reducida R es:
x_1
x_2
```

Código 3.8: Ejemplo Algoritmo de Reducción en Java

4

Aplicación al problema del k-coloreado y sudokus

Una vez visto cómo obtener bases de Gröbner reducidas, vamos a estudiar en este capítulo algunas de las aplicaciones que tienen sobre grafos y sudokus. Para el desarrollo de este capítulo, nos hemos basado en la siguiente fuente [2].

En primer lugar, definimos algunos de los conceptos básicos necesarios para introducir los problemas que se plantearán.

Def 8. Un **grafo** G se define como un par $G = (V, E)$, donde V está formado por un conjunto de puntos llamados **vértices** y E es el conjunto de los pares no ordenados de esos vértices. Estos pares se conocen como **aristas del** G. Se dice que son **adyacentes** dos vértices que están unidos por una misma arista.

Def 9. Se define un grafo G como **conexo** si para todo par de vértices $u, v \in V$, siempre existe una sucesión de aristas que conectan el vértice v con el vértice u. A esta sucesión de aristas se le llama camino entre u y v.

Def 10. Dado G un grafo conexo y C un conjunto de elementos finito. Llamaremos **coloración de** G a la asignación de los elementos de C a los que llamamos colores a cada uno de los vértices del grafo, de tal manera que dos vértices adyacentes no pueden tener asignado un mismo color.

4.1. Problema del k-coloreado

Def 11. Se dice que un grafo G es **k-coloreable** si existe una coloración para G donde el conjunto C tiene k colores.

El problema del k-coloreado busca si para un grafo $G = (V, E)$ existe una coloración con k elementos. Para resolverlo utilizando bases de Gröbner, se representa G como sistema de ecuaciones polinómicas.

Dado un grafo $G = (V, E)$ con n vértices, donde $\{x_1, x_2, ..., x_n\}$ representa el conjunto de las variables asignadas a cada vértice de V. Tomamos el conjunto de colores $C = \{c_1, c_2, ..., c_k\}$, con $k < n$. Solo estudiaremos el caso en que $k < n$, ya que de lo contrario la coloración sería trivial. Al haber mayor o igual número de colores que número de vértices, podríamos asignar siempre un color distinto a cada vértice. Por ejemplo, $x_1 = c_1, x_2 = c_2, ..., x_n = c_n$. De esta manera se cumplirían las dos condiciones para que fuera una k-coloración. Siempre estarían todos los vértices coloreados y aquellos pares adyacentes tendrían colores distintos.

4.1.1. Sistema de ecuaciones

- La primera de las ecuaciones que definiremos parte de la necesidad de asignar un color a la variable x_i, $i \in \{1, ..., n\}$. El valor de x_i debe ser uno de los elementos de C, por tanto, el productorio debe ser igual a 0.

$$F(x_i) = \prod_{t=1}^{k} (x_i - c_t) \tag{4.1}$$

$$F(x_i) = 0, \quad \forall i \in \{1, ..., n\} \tag{4.2}$$

- En segundo lugar, se necesita controlar que aquellos vértices adyacentes no tengan un mismo color asignado, es decir, si $\{x_i, x_j\} \in E$, entonces $x_i \neq x_j$.

  Para ello, partiremos de esta primera ecuación obtenida. Como sabemos que $F(x_i)$ es igual a cero, la resta de $F(x_i) - F(x_j)$ también lo será. Definimos entonces $G(x_i, x_j)$ como:

$$F(x_i) - F(x_j) = (x_i - x_j)\frac{F(x_i) - F(x_j)}{x_i - x_j} \overset{(*)}{=} (x_i - x_j)G(x_i, x_j) \tag{4.3}$$

$$G(x_i, x_j) = \frac{F(x_i) - F(x_j)}{x_i - x_j} \tag{4.4}$$

Como sabemos que $x_i \neq x_j$ para los pares adyacentes, $x_i - x_j \neq 0$, por tanto, $\frac{F(x_i)-F(x_j)}{x_i-x_j} = 0 \ \forall \{x_i, x_j\} \in E$.

Analicemos ahora por qué podemos realizar el paso (*) y afirmar que $G(x_i, x_j)$ es también un polinomio.

En primer lugar nos centramos únicamente en el desarrollo del numerador de $\frac{F(x_i)-F(x_j)}{x_i-x_j}$.

$$\begin{aligned} &F(x_i) - F(x_j) = (x_i - c_1)(x_i - c_2)\cdots(x_i - c_n) - (x_j - c_1)(x_j - c_2)\cdots(x_j - c_n) \\ &= (x_i^n + a_1 x_i^{n-1} + \cdots + a_{n-1}x_i + a_n) - (x_j^n + a_1 x_j^{n-1} + \cdots + a_{n-1}x_j + a_n) \\ &= (x_i^n - x_j^n) + (a_1 x_i^{n-1} - a_1 x_j^{n-1}) + \cdots + (a_{n-1}x_i - a_{n-1}x_j) + (a_n - a_n) \\ &= (x_i^n - x_j^n) + a_1(x_i^{n-1} - x_j^{n-1}) + \cdots + a_{n-1}(x_i - x_j) \end{aligned}$$

Utilizando ahora la identidad notable de diferencia de exponentes donde se cumple que:

$$(a^n - b^n) = (a - b)(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \cdots + ab^{n-2} + b^{n-1})$$

Obtenemos que todos los factores que conforman nuestra ecuación tienen como factor común $(x_i - x_j)$, por tanto, podemos dividirlo por este factor y el resultado obtenido sigue siendo un polinomio.

También sobre este planteamiento, vamos a ver que siempre que exista un conjunto de soluciones para este sistema de ecuaciones, estas van a proporcionar una k-coloración para el grafo:

- Supongamos que tenemos una solución para el sistema de ecuaciones 4.1 donde x_i toma un valor $c_k \notin C$. Entonces, si $F(x_i) = (x_i - c_1)(x_i - c_2)\cdots(c_i - c_t) = 0$, tiene que ocurrir que alguno de los factores del productorio sea 0. Es decir, $x_i - c_1 = 0$, o $x_i - c_2 = 0$, ..., o $x_i - c_t = 0$. Pero como partíamos de que $c_k \notin C$, el factor $x_i - c_k$ no estará en el productorio que forma $F(x_i)$ y por tanto, nunca tomará ningún factor el valor 0. Queda demostrado así que si tenemos una solución de $F(x_i)$, esta tendrá un valor contenido en C.
- Veamos ahora un ejemplo ilustrativo en $k = 2$ con el conjunto de colores $C = \{a, b\}$, donde siempre que tenemos una solución de $G(x_i, x_j)$, esto obliga a que los valores que toman x_i y x_j no sean los mismos. Construyamos entonces la ecuación $G(x_i, x_j)$.

$$\begin{aligned} F(x_i) - F(x_j) &= x_i^2 - (a+b)x_i + ab - x_j^2 + (a+b)x_j - ab \\ &= (x_i^2 - x_j^2) - (a+b)(x_i - x_j) + ab - ab \\ &= (x_i + x_j)(x_i - x_j) - (a+b)(x_i - x_j) \end{aligned}$$

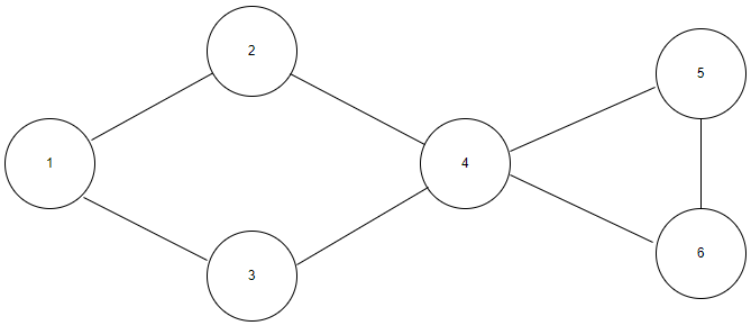

Figura 4.1: Problema 3-coloreado

$$\begin{aligned} G(x_i, x_j) =& \frac{F(x_i) - F(x_j)}{x_i - x_j} = \frac{(x_i + x_j)(x_i - x_j) - (a+b)(x_i - x_j)}{x_i - x_j} \\ =& (x_i + x_j) - (a+b) \end{aligned}$$

Supongamos que tenemos una solución que cumple que $G(x_i, x_j) = 0$ donde $x_i = x_j$. Por ejemplo, tomemos $x_i = a$.

$$\begin{aligned} G(x_i, x_j) = (x_i + x_j) - (a+b) &= 0 \\ x_i + x_j &= a + b \\ 2a &= a + b \\ a &= b \end{aligned}$$

Hemos llegado así a una contradicción, ya que los elementos de C deben ser distintos entre ellos. Concluimos entonces que siempre que tengamos una solución del sistema de ecuaciones 4.4 para $k = 2$, esto cumplirá la condición de que $x_i \neq x_j$.

Ejemplo 7. Veamos ahora un ejemplo donde resolveremos el problema de 3-coloreado en el grafo de la Figura 4.1.

Nuestro conjunto V está formado por los seis vértices del grafo, representados por el conjunto de variables $\{x_1, x_2, x_3, x_4, x_5, x_6\}$. Con esta notación, el conjunto E que define los pares conectados por una arista estaría formado por $\{x_1, x_2\}$, $\{x_1, x_3\}$, $\{x_2, x_4\}$, $\{x_4, x_5\}$,$\{x_4, x_6\}$, $\{x_5, x_6\}$

Como vamos a resolver el problema del k-coloreado con $k = 3$, el conjunto de colores C estará formado por tres colores que representaremos con números $1, 2, 3$.

En primer lugar, definimos las seis ecuaciones $F(x)$ que justifican el coloreado de cada vértice.

$$F(x_1) = \prod_{t=1}^{3}(x_1 - c_t) = x_1^3 - 6x_1^2 + x_1 - 6 \tag{4.5}$$

$$F(x_2) = \prod_{t=1}^{3}(x_2 - c_t) = x_2^3 - 6x_2^2 + x_2 - 6 \tag{4.6}$$

$$F(x_3) = \prod_{t=1}^{3}(x_3 - c_t) = x_3^3 - 6x_3^2 + x_3 - 6 \tag{4.7}$$

$$F(x_4) = \prod_{t=1}^{3}(x_4 - c_t) = x_4^3 - 6x_4^2 + x_4 - 6 \tag{4.8}$$

$$F(x_5) = \prod_{t=1}^{3}(x_5 - c_t) = x_5^3 - 6x_5^2 + x_5 - 6 \tag{4.9}$$

$$F(x_6) = \prod_{t=1}^{3}(x_6 - c_t) = x_6^3 - 6x_6^2 + x_6 - 6 \tag{4.10}$$

En segundo lugar, definimos las ecuaciones $G(x_i, x_j)$ para cada par de vértices adyacentes.

$$G(x_1, x_2) = \frac{F(x_1) - F(x_2)}{x_1 - x_2} = x_1^2 + x_1x_2 + x_2^2 - 6x_1 - 6x_2 + 11 \tag{4.11}$$

$$G(x_1, x_3) = \frac{F(x_1) - F(x_3)}{x_1 - x_3} = x_1^2 + x_1x_3 + x_3^2 - 6x_1 - 6x_3 + 11 \tag{4.12}$$

$$G(x_2, x_4) = \frac{F(x_2) - F(x_4)}{x_2 - x_4} = x_2^2 + x_2x_4 + x_4^2 - 6x_2 - 6x_4 + 11 \tag{4.13}$$

$$G(x_3, x_4) = \frac{F(x_3) - F(x_4)}{x_3 - x_4} = x_3^2 + x_3x_4 + x_4^2 - 6x_3 - 6x_4 + 11 \tag{4.14}$$

$$G(x_4, x_5) = \frac{F(x_4) - F(x_5)}{x_4 - x_5} = x_4^2 + x_4x_5 + x_5^2 - 6x_4 - 6x_5 + 11 \tag{4.15}$$

$$G(x_4, x_6) = \frac{F(x_4) - F(x_6)}{x_4 - x_6} = x_4^2 + x_4x_6 + x_6^2 - 6x_4 - 6x_6 + 11 \tag{4.16}$$

$$G(x_5, x_6) = \frac{F(x_5) - F(x_6)}{x_5 - x_6} = x_5^2 + x_5x_6 + x_6^2 - 6x_5 - 6x_6 + 11 \tag{4.17}$$

Una vez tenemos las funciones definidas, calculamos la base de Gröbner del ideal formado por $F(x_i)$, $\forall i \in V$ y $G(x_i, x_j)$, $\forall i, j \in E$ aplicando el algoritmo de

Buchberger.

Introduciendo esto en la aplicación de Java se obtiene la siguiente base de Gröbner reducida:

$$x_6^3 - 6x_6^2 + x_6 - 6 \tag{4.18}$$
$$-x_1x_3 - x_3x_6 + x_1x_2 - 6x_2 - x_3x_5 + x_2x_6 + x_2x_5 + 6x_3 \tag{4.19}$$
$$-6x_3 - x_5x_6 + x_3x_5 + x_3x_6 + 11 - 6x_1 + x_1x_3 + x_1^2 \tag{4.20}$$
$$x_5x_6 + x_2^2 - x_2x_5 - x_2x_6 \tag{4.21}$$
$$-x_3x_6 + x_5x_6 + x_3^2 - x_3x_5 \tag{4.22}$$
$$11 - 6x_6 + x_5^2 + x_5x_6 + x_6^2 - 6x_5 \tag{4.23}$$
$$x_6 + x_5 - 6 + x_4 \tag{4.24}$$

Como vemos, es un sistema de ecuaciones mucho más simple de resolver que el de partida. Existen distintas coloraciones para 4.1 que cumplen el sistema de ecuaciones obtenido.

Vamos a suponer $x_6 = 1$, utilizando el método de sustitución en el sistema, una de las posibles coloraciones sería:

$$\begin{cases} x_1 = 2 \\ x_2 = 3 \\ x_3 = 1 \\ x_4 = 2 \\ x_5 = 3 \\ x_6 = 1 \end{cases} \tag{4.25}$$

Con esta coloración el grafo quedaría coloreado como se muestra en 4.2

Es obvio que para resolver problemas de k-coloreado siempre será necesario suponer el color de una de las variables, pues estos son arbitrarios. Podríamos haber partido de que el valor de x_6 fuera 2 o 3 y habríamos obtenido coloraciones análogas.

Es por esto que es posible añadir en el propio sistema de ecuaciones al que aplicamos posteriormente los algoritmos de Buchberger y de reducción una ecuación extra que indique el valor de una de las variables. Introduciendo la ecuación $x_6 - 1$ en el sistema anteior, el resultado que se obtendría en la aplicación sería la siguiente base reducida:

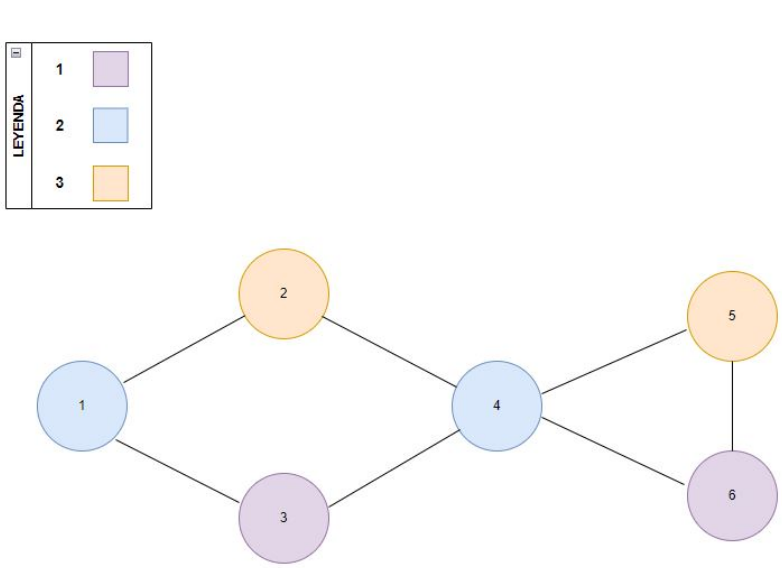

Figura 4.2: Problema 3-coloreado

$$x_6 - 1 \tag{4.26}$$
$$-5x_2 + x_1x_2 - x_1x_3 + 5x_3 - x_3x_5 + x_2x_5 \tag{4.27}$$
$$x_3x_5 - x_5 + x_1^2 + x_1x_3 - 5x_3 - 6x_1 + 11 \tag{4.28}$$
$$x_2^2 + x_5 - x_2x_5 - x_2 \tag{4.29}$$
$$-x_3x_5 - x_3 + x_5 + x_3^2 \tag{4.30}$$
$$6 + x_5^2 - 5x_5 \tag{4.31}$$
$$-5 + x_5 + x_4 \tag{4.32}$$

Con estos resultados podemos concluir que el grafo 4.1 es 3-coloreable. Pero, si no lo fuera, ¿qué resultado obtendríamos al aplicar el algoritmo de Buchberger?

Probemos ahora a introducir las ecuaciones correspondientes para resolver el problema del 2-coloreado sobre el grafo del ejemplo. Las ecuaciones $F(x)$ tendrían ahora la siguiente forma:

$$F(x_i) = x_i^2 - 3x_i + 2, \quad \forall x_i \in V \tag{4.33}$$

Mientras que las $G(x_i, x_j)$ serían:

$$G(x_i, x_j) = x_i + x_j - 3, \quad \forall (x_i, x_j) \in E \tag{4.34}$$

El resultado que se obtiene tras aplicar el algoritmo es que la base reducida es $Gr = 1$. Esto quiere decir que el sistema de ecuaciones proporcionado no tiene soluciones.

4.2. Sudokus

En esta sección se va a aplicar el algoritmo de Buchberger a la resolución de Sudokus basándose en la idea de resolución problema del k-coloreado.

Def 12. El **Sudoku** es un juego que tradicionalmente consiste de un tablero de 9x9 casillas dividido en 9 columnas, 9 filas y 9 regiones. Se entiende como región a cada cuadrícula de 3x3 casillas. Cada casilla debe ser rellenada con números de entre el 1 y el 9, no pudiendo repetir número en una misma columna, fila o región. El juego comienza con una serie de casillas ya rellenas.

Es posible plantear la resolución de un Sudoku basándose en el problema del k-coloreado.

Para ello se modela el Sudoku como un grafo conexo donde cada casilla es representada por un vértice x_i con $i = \{1, ..., 81\}$ y el conjunto de aristas lo forman cada una de las relaciones entre aquellas casillas que no pueden tener el mismo valor. Por ejemplo, dos casillas que están en la misma fila tendrán un par en el conjunto E formado por los dos vértices que las representan.

Se plantea así un sistema de ecuaciones análogo al que se ha descrito en la sección anterior.

- El primer conjunto de ecuaciones es el formado por aquellas que modelan la necesidad de que cada casilla tome un valor entre 1 y 9. El conjunto de colores será ahora $C = \{c_k : 1 \leqslant k \leqslant 9\}$

  De esta forma, se define $F(x_i)$ como:

$$F(x_i) = \prod_{t=1}^{9}(x_i - c_t) = (x_i - 1)(x_i - 2)\cdots(x_i - 9), \quad \forall\ 1 \leqslant i \leqslant 81$$

$$\begin{aligned} F(x_i) =& x_i^9 - 45x_i^8 + 855x_i 7 - 9450x_i^6 + 63273x_i^5 - 269325x_i^4 \\ &+ 723680x_i^3 - 1164240x_i^2 + 984960x_i - 362880 \end{aligned}$$

- El segundo conjunto de ecuaciones es el formado por aquellas que modelan el hecho de que dos casillas de un mismo grupo no pueden tener el mismo color

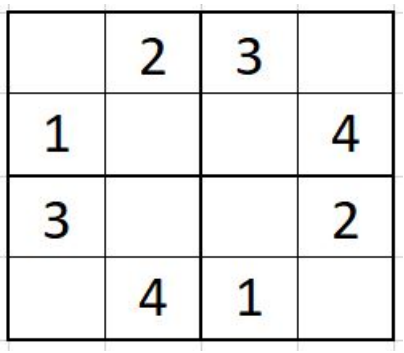

Figura 4.3: Shidoku

o número. Entendemos como grupo fila, columna o cuadrado. Planteando $G(x_i, x_j)$ de la misma manera que antes, obtenemos:

$$G(x_i, x_j) = \frac{F(x_i) - F(x_j)}{x_i - x_j}, \quad \forall \ \{x_i, x_j\} \in E$$

$$\begin{aligned} G(x_i, x_j) =& x_i^8 + x_j^8 + x_i^7 x_j - 38 x_i^6 x_j^2 - 253 x_i^5 x_j^3 \\ &+ 1725 x_i^4 x_j^4 + 253 x_i^3 x_j^5 - 38 x_i^2 x_j^6 + x_i x_j^7 \end{aligned}$$

- Para este tipo de problemas vamos a tener un tercer conjunto de ecuaciones que no teníamos en el problema del k-coloreado. Estas son las ecuaciones que representan aquellas casillas que ya están rellenas al iniciar el juego, las pistas. Este conjunto será de la forma: $H(x_i) = x_i - c_k$, siendo c_k el valor impuesto para esa casilla en el inicio.

Por tanto, para resolver un sudoku planteado, habrá que calcular la base de Gröbner del ideal formado por las $F(x_i), G(x_i, x_j)$ y $H(x_i)$.

4.2.1. Otros tipos de Sudokus

Shidokus

Los Shidokus son un tipo particular de Sudokus donde el tablero es de tamaño 4x4 en lugar de 9x9. El conjunto de colores, por tanto, está formado por $C : \{c_k : 1 \leqslant k \leqslant 4\}$. Vamos a utilizar este tipo de Sudokus para analizar la resolución de este problema, ya que es un caso más simple y fácil de ver que el caso del Sudoku tradicional.

Ejemplo 8. En este ejemplo plantearemos el sistema de ecuaciones y lo resolveremos usando el algoritmo de Buchberger y reduciendo la base obtenida en la aplicación sobre el tablero planteado en la Figura 4.3.

El conjunto de variables en este caso es el siguiente: $V = \{x_i : 1 \leqslant i \leqslant 16\}$, asignando a cada casilla de izquierda a derecha y de arriba hacia abajo una variable distinta.

De esta manera, el conjunto de aristas estaría formado por la unión de los siguientes conjuntos. $E = E_{FILAS} \cup E_{COLUMNAS} \cup E_{CUADRADOS}$

$$\begin{aligned} E_{FILAS} = &\{\{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_4\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_4, x_4\}\}, \\ &\{\{x_5, x_6\}, \{x_5, x_7\}\, \{x_5, x_8\}, \{x_6, x_7\}, \{x_6, x_8\}, \{x_7, x_8\}\}, \\ &\{\{x_9, x_{10}\}, \{x_9, x_{11}\}\, \{x_9, x_{12}\}, \{x_{10}, x_{11}\}, \{x_{10}, x_{12}\}, \{x_{11}, x_{12}\}\}, \\ &\{\{x_{13}, x_{14}\}, \{x_{13}, x_{15}\}\, \{x_{13}, x_{16}\}, \{x_{14}, x_{15}\}, \{x_{14}, x_{15}\}, \{x_{15}, x_{16}\}\} \end{aligned}$$

$$\begin{aligned} E_{COLUMNAS} = &\{\{x_1, x_5\}, \{x_1, x_9\}, \{x_1, x_{13}\}, \{x_5, x_9\}, \{x_5, x_{13}\}, \{x_9, x_{13}\}\}, \\ &\{\{x_2, x_6\}, \{x_2, x_{10}\}\, \{x_2, x_{14}\}, \{x_6, x_{10}\}, \{x_6, x_{14}\}, \{x_{10}, x_{14}\}\}, \\ &\{\{x_3, x_7\}, \{x_3, x_{11}\}\, \{x_3, x_{15}\}, \{x_7, x_{11}\}, \{x_7, x_{15}\}, \{x_{11}, x_{15}\}\}, \\ &\{\{x_4, x_8\}, \{x_4, x_{12}\}\, \{x_4, x_{16}\}, \{x_8, x_{12}\}, \{x_8, x_{12}\}, \{x_{12}, x_{16}\}\} \end{aligned}$$

$$\begin{aligned} E_{CUADRADOS} = &\{\{x_1, x_2\}, \{x_1, x_5\}, \{x_1, x_6\}, \{x_2, x_5\}, \{x_2, x_6\}, \{x_5, x_6\}\}, \\ &\{\{x_3, x_4\}, \{x_3, x_7\}\, \{x_3, x_8\}, \{x_4, x_7\}, \{x_4, x_8\}, \{x_7, x_8\}\}, \\ &\{\{x_9, x_{10}\}, \{x_9, x_{13}\}\, \{x_9, x_{14}\}, \{x_{10}, x_{13}\}, \{x_{10}, x_{14}\}, \{x_{13}, x_{14}\}\}, \\ &\{\{x_{11}, x_{12}\}, \{x_{11}, x_{15}\}\, \{x_{11}, x_{16}\}, \{x_{12}, x_{15}\}, \{x_{12}, x_{16}\}, \{x_{15}, x_{16}\}\} \end{aligned}$$

Para calcular ahora el ideal sobre el que se aplicará el algoritmo de Buchberger, debemos obtener las $H(x_i)$, $F(x_i)$ y $G(x_i, x_j)$. En este caso serán de la siguiente forma:

$$H(x_i) = x_i - c_k$$

$$F(x_i) = x_i^4 - 10x_i^3 + 35x_i^2 - 50x_i + 24, 1 \leqslant i \leqslant 16$$

$$G(x_i, x_j) = x_i^3 + x_j^3 + x_i^2 x_j + x_i x_j^2 - 10x_i^2 - 10x_j^2 - 10x_i x_j + 35x_i + 35x_j - 50$$

Como ya partimos de algunas celdas coloreadas y estas vienen definidas por las ecuaciones $H(x_i)$, no es necesario añadir al sistema las $F(x_i)$ de estas variables, ya que estas ecuaciones modelaban la necesidad de que tomen uno de los valores de los colores propuestos, por tanto, si las añadimos estaríamos introduciendo ecuaciones redundantes. No obstante, de cualquier manera se obtendría el mismo resultado, la única diferencia es que al partir de un mayor número de ecuaciones, tardaríamos más en resolver el problema.

Si se quiere ver en detalle las ecuaciones introducidas, ver en Apéndice B.

Aplicamos ahora el algoritmo sobre el ideal $I = J + L = \langle F(x_i), G(x_i, x_j)\rangle + \langle x_i - c_k\rangle$.

El resultado obtenido al ejecutar la aplicación es el que se muestra a continuación:

```
OUTPUT:
x_1 -4
x_2 -2
x_3 -3
x_4 - 1
x_5 -1
x_6 -3
x_7 -2
x_8 - 4
x_9 - 3
x_10 - 1
x_11 -4
x_12 - 2
x_13 -2
x_14 -4
x_15 -1
x_16 - 3
```

Código 4.1: Ejemplo Shidoku en Java

Por tanto, el Shidoku relleno quedaría como se muestra en la Figura 4.4.

Sudoku Killer

En este tipo de Sudokus, se añaden otro tipo de conjuntos de casillas, normalmente delimitados por lineas discontinuas o de otro color, donde se indica con un

4	2	3	1
1	3	2	4
3	1	4	2
2	4	1	3

Figura 4.4: Shidoku

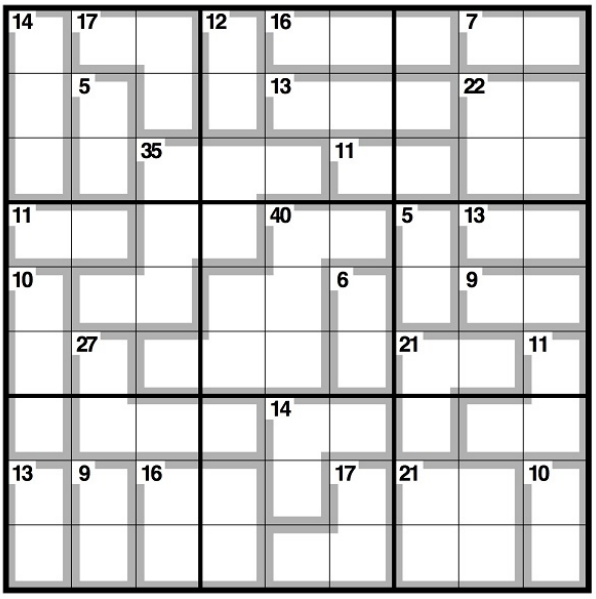

Figura 4.5: Sudoku Killer

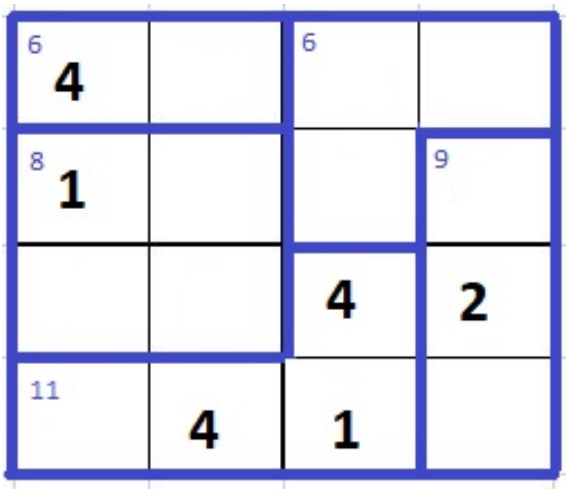

Figura 4.6: Shidoku Killer

número más pequeño lo que debe sumar el contenido de ese grupo. Se muestra un ejemplo en 4.5.

Para plantear la resolución de los Sudoku Killer, solo habría que añadir un cuarto tipo de ecuaciones a las mencionadas en la sección anterior. Estas modelarían la necesidad de que el valor de un grupo de casillas sumara el número indicado.

Veamos un caso más simple planteando las ecuaciones que lo modelarían e introduciéndolo en la aplicación. Este se presenta en la Figura 4.6.

Si introducimos las ecuaciones que modelan este shidoku en la aplicación de Java, pero, sin tener en cuenta las condiciones que lo hacen Killer. Es decir, sin tener en cuenta lo que suman los conjuntos de casillas, no obtendríamos una solución única, sino que tendríamos el siguiente resultado:

```
OUTPUT:
La base reducida Gr es:
x_1 -4
x_5 -1
x_11 -4
x_12 -2
 -4 + x_14
x_15 -1
 -5x_7 + 6 + x_7^2
x_16 -3
x_9 -3
 -1 + x_4
 -4 + x_8
x_2 -x_7
```

```
x_7 -5 + x_3
 -5 + x_7 + x_6
x_13 -2
x_10 -1
```

Código 4.2: Ejemplo Shidoku Killer en Java varias soluciones

De esta manera, podríamos tener una solución con $x_2 = 2, x_3 = 3, x_6 = 3, x_7 = 2$, y otra opción con $x_2 = 3, x_3 = 2, x_6 = 2, x_7 = 3$.

Para añadir las condiciones de las sumas de las casillas y así obtener una solución única, debemos añadir las siguientes ecuaciones al modelo.

$$x_1 + x_2 - 6 \tag{4.35}$$
$$x_3 + x_4 + x_7 - 6 \tag{4.36}$$
$$x_5 + x_6 + x_9 + x_{10} - 8 \tag{4.37}$$
$$x_8 + x_{12} + x_{16} - 9 \tag{4.38}$$
$$x_{11} + x_{13} + x_{14} + x_{15} - 11 \tag{4.39}$$

El sistema de ecuaciones completo quedaría como se muestra en el Anexo C.

El resultado obtenido al calcular la base es el siguiente:

```
OUTPUT:
x_1 -4
x_2 -2
x_3 -3
x_4 - 1
x_5 -1
x_6 -3
x_7 -2
x_8 - 4
x_9 - 3
x_10 - 1
x_11 -4
x_12 - 2
x_13 -2
x_14 -4
x_15 -1
x_16 - 3
```

Código 4.3: Ejemplo Shidoku Killer en Java

El shidoku completo con las soluciones quedaría como se muestra en la Figura 4.7.

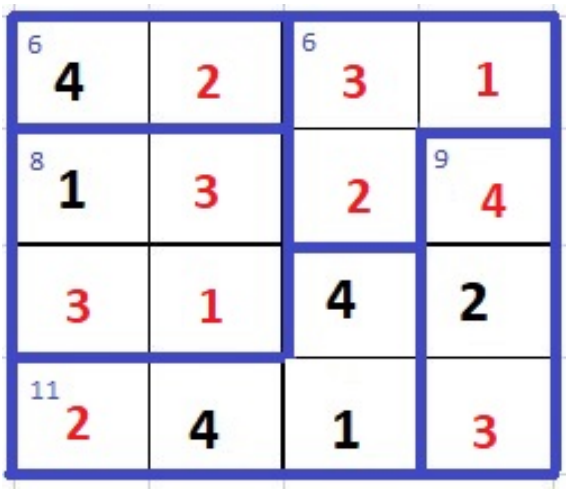

Figura 4.7: Shidoku Killer Solución

Sudoku Diagonal

Este es un sudoku donde las casillas de las diagonales principales no deben tampoco repetir sus valores. Como se aprecia en la Figura 4.8, ambas diagonales están marcadas por una línea diferenciada discontinua o de color en la cual no pueden repetirse los números.

De igual manera, podemos modelar este tipo de sudokus añadiendo un cuarto tipo de ecuaciones que definan la incopatibilidad de mismos valores en la diagonal. Esto lo haremos de la misma manera que cuando se modela imposibilidad de dos valores iguales en una misma fila, columna o cuadrado. Se añade una ecuación $G(x_i, x_j)$ por cada nueva arista o par conectado. En este caso, también existirán aristas entre los vértices que representan las casillas de la diagonal, por tanto, habrá que añadir una ecuación de tipo G por todos los pares en ambas diagonales.

Para ver un ejemplo de este tipo de Sudokus más sencillo, se plantea el siguiente Shidoku 4.9, en un tablero de 4×4.

Si aplicamos el algoritmo de Buchberger sobre las ecuaciones que modelan este shidoku, sin añadir las restricciones de ser un shidoku diagonal, obtendremos un sistema de ecuaciones con más de una solución posible como se muestra en el siguiente fragmento de código:

```
OUTPUT:
x_2 -4
x_3 -2
x_4 -3
x_7 -1
x_10^2 - 5x_10 + 6
x_12 -1
```

		9	2		1	6		
				6				
2								1
7				9				8
	1		7		5		3	
3				1				4
9								2
				2				
		2	6		9	3		

Figura 4.8: Sudoku Diagonal

	4		3
		1	
			1
4			

Figura 4.9: Shidoku Diagonal

```
x_13 -4
x_1 - 1
x_6 + x_10 - 5
x_9 + x_10 - 5
x_8 - 4
x_16 -2
x_14 - 1
x_15 -3
x_5 - x_10
x_11 -4
```

Código 4.4: Ejemplo Shidoku varias soluciones en Java

En este caso, la variable x_{10} puede tomar los valores 2 y 3, de manera que x_5, x_6 y x_9 también pueden variar sus valores.

Si convertimos este en un shidoku diagonal, sí obtendremos una única solución, ya que añadimos cuatro ecuaciones más que condicionan que los pares situados en las diagonales principales no deban tener el mismo valor.

Se indican todas las ecuaciones que modelan el shidoku en el Apéndice D. Concretamente, las que se han incluido son las $G(x_i, x_j)$ para los pares

$$\{\{x_1, x_{11}\}, \{x_1, x_{16}\}, \{x_4, x_{10}\}, \{x_4, x_{13}\}\}.$$

El resultado obtenido al calcular la base de Gröbner reducida es el siguiente:

```
OUTPUT:
x_2 -4
x_3 -2
x_4 -3
x_7 -1
x_10 - 2
x_12 -1
x_13 -4
x_1 - 1
x_6 -3
x_9 - 3
x_8 - 4
x_16 -2
x_14 - 1
x_15 -3
x_5 -2
x_11 -4
```

Código 4.5: Ejemplo Shidoku Diagonal en Java

El resultado del Shidoku relleno con las soluciones se muestra en la Figura

1	4	2	3
2	3	1	4
3	2	4	1
4	1	3	2

Figura 4.10: Shidoku Diagonal Solución

4.10.

4.2.2. Observaciones

Dificultad computacional

En un ejemplo tan simple como el hemos resuelto en 8, ya partimos de 72 ecuaciones para nuestro sistema. Recordando el funcionamiento del algoritmo de Buchberger, vemos que tenemos que trabajar sobre el conjunto de sus pares. Esto hace un total de 2556 pares. En un ordenador corriente es posible trabajar con estos números, pero, planteando el mismo problema sobre un sudoku tradicional de 81 variables, se obtiene un conjunto de pares de 396.495 elementos en un ejemplo con 30 pistas. Esto lo hace inviable en un ordenador con baja potencia.

El algoritmo de Buchberger no es la opción óptima para obtener la base de Gröbner de ideales tan grandes. Para este tipo de problemas existen otros métodos más rápidos que van reduciendo el ideal introducido más eficientemente.

Por ejemplo, se podría utilizar el algoritmo F4 que mejora la arbitrariedad existente en el algoritmo de Buchberger al seleccionar un nuevo par a eliminar del conjunto de pares. Esto se hace sin ningún orden de selección. En el algoritmo F4 se plantea la opción de reducir de una vez un subconjunto de elementos, además de introducir el uso de matrices que representen el sistema, de manera que se aplica algo similar al método de reducción de Gauss - Jordan. Al ir reduciendo el sistema inicial utilizando ciertos criterios, esto se hace mucho más eficiente. Para profundizar sobre el funcionamiento de este algoritmo se puede visitar la siguiente publicación [3].

5

Desarrollo de la aplicación en Java: Implementación y análisis

En este capítulo vamos a analizar el código desarrollado en Java para implementar los algoritmos utilizados a lo largo de todo el trabajo.

Algunos de los métodos más relevantes para este trabajo se encuentran en el Apéndice A. El código completo de la aplicación se encuentra en el siguiente enlace de GitHub siendo este de dominio público para cualquier consulta, ver Fuente [4].

5.1. Estructura de la aplicación

La estructura de la aplicación se basa en la definición de clases y métodos necesarios para el uso de polinomios, ya que todos los algoritmos que se plantean posteriormente se basan en el uso y manipulación de éstos.

Se divide en tres clases, dos de ellas destinadas a la definición de polinomios y de los algoritmos de tratado de bases de Gröbner y una tercera clase de definición de ecuaciones de sudokus de dimensiones 9×9 y 4×4.

5.1.1. Clase Monomio

Esta clase consta de dos atributos que definen un monomio.

Atributos

- **Coeficiente** (ArrayList$\langle Integer \rangle$). Este atributo representa el coeficiente de un monomio a través de una fracción. Se ha elegido el uso de fracciones en lugar de números decimales para evitar errores en el redondeo.
- **Exponentes** (ArrayList$\langle Integer \rangle$). Este atributo consta de una lista de valores enteros con tantos elementos como distintas variables haya en el polinomio. Cada número entero representa el exponente de cada variable en el monomio, estando estos ordenados de mayor a menor variables.

  Por ejemplo, si se quiere representar $3x^2y$, dentro de un espacio de tres variables donde $x > y > z$, el elemento de la clase monomio estaría compuesto por el coeficiente $[3, 1]$ y los exponentes $[2, 1, 0]$.

Dentro de esta clase se definen los métodos necesarios para operar entre monomios, ya que los coeficientes que tratamos están definidos en forma de fracción, previamente se definen también los métodos para operar fracciones.

Operadores fracciones

- Suma fracciones
- Resta fracciones
- Multiplicación fracciones
- División fracciones

Operadores monomios

- Suma monomios
- Resta monomios
- Multiplicación monomios
- División monomios

También se definen algunos métodos más, propios de los algoritmos que se han implementado relacionados con este trabajo.

Otros métodos necesarios para el trabajo con monomios

- Exponente de un monomio: calcula el exponente de un monomio dado.
- Monomio mayor deg: dados dos monomios, devuelve el monomio mayor según el orden deg.
- Monomio mayor lex: dados dos monomios, devuelve el monomio mayor según el orden lex.
- Monomio divisible: comprueba si un monomio divide a otro dado.

5.1.2. Clase Polinomio

En un segundo nivel, tenemos la clase Polinomio.

Atributos

- **Monomios**: (ArrayList$\langle Monomio \rangle$): lista de monomios que comprenden el polinomio.

A continuación se van a comentar algunos de los métodos implementados más significativos para el trabajo.

Cálculo de términos relevantes

- ltPolinomio: calcula el término director de un monomio. Va comparando uno a uno los monomios del polinomio quedándose con el mayor.
- lcPolinomio: obtiene el término director del polinomio y devuelve su coeficiente.
- lpPolinomio: calcula el término director del polinomio sin coeficiente.

Operadores polinomios

- Suma polinomios: se estudia si hay monomios iguales y en tal caso, se suman esos monomios. Finalmente se simplifica el resultado.
- Resta polinomios: análogo a la suma.
- Multiplicación polinomios: se aplica la multiplicación de monomios para cada par de los polinomios dados. Se simplifica el resultado.

- División univariable. Este método está basado en el Algoritmo 1 devolviendo en una lista el polinomio resultante de la división en la posición [0] y el resto de la division en [1]. Se encuentra el código detallado en el Apéndice A.1.

- Algoritmo Euclídeo. Se basa en el Algoritmo 2 para el cálculo del máximo común divisor entre dos polinomios. Se detalla el código en el Apéndice A.2.

Bases de Gröbner

- División multivariable. Este método aplica el algoritmo presentado en el Algoritmo 3. Se realiza la división de un polinomio f sobre un conjunto de n polinomios. Tras aplicar el algoritmo se devuelve una lista de polinomios donde los n primeros elementos representan los polinomios u_i, mientras que el último elemento representa el resto de la división. Se puede ver en detalle este código en el Apéndice A.3.

- Cálculo S-Polinomio. Este método calcula el S-Polinomio de dos polinomios dados obteniendo los términos directores de cada uno de los polinomios y el término L, el mínimo común múltiplo de sus productos directores.

- Algoritmo de Buchberger. Este método desarrolla el algoritmo definido en el Algoritmo 4 para obtener la base de Gröbner de un conjunto de polinomios dado. Va eliminando los pares de polinomios en orden según se hayan introducido en la lista de entrada. El código en detalle de esta función se encuentra en el Apéndice A.4.

- Algoritmo de reducción. En esta funcion se introduce como parámetro el conjunto de polinomios que forman la base de Gröbner. Se aplica sobre este el algoritmo de reducción mencionado en el Algoritmo 5. El código detallado se encuentra en el Apéndice A.5.

5.1.3. Clase Sudoku

Por último, tenemos la clase Sudoku donde dada una matriz 4×4 o 9×9 que representa un sudoku, transforma esta información en un sistema de ecuaciones.

Atributos

- **Tamaño**: Integer: tamaño de un lado del tablero

- **Tablero**: Integer[][]: Matriz donde se introduce el sudoku. Todos los valores son 0 salvo las pistas que van rellenas con el número que corresponda.

En esta clase hay un único método principal, **getSudokuEquations()**, el cual calcula las distintas ecuaciones que modelan el sudoku.

En primer lugar, va recorriendo cada elemento del tablero revisando si es una pista o un valor vacío. En caso de ser una pista, genera un polinomio del tipo $x_i - c$, donde c representa el color, o en este caso número de la casilla. Si el elemento que se encuentra no está relleno, genera la ecuación $f(x_i)$ correspondiente.

En segundo lugar, se generan las ecuaciones $g(x_i, x_j), \{(x_i, x_j)\} \in E$. Esto se hace en tres partes distintas.

En un primer bucle se va comprobando aquellos pares de elementos que están en una mista fila. En un segundo bucle se comprueban los pares de elementos que comparten columna. En un tercero se estudia si el par está en un mismo cuadrado, entendiendo cuadrado como una división del tablero $n \times n$ en cajas de tamaño $\sqrt{n}$. Es interesante analizar la función que comprueba esta última condición.

```
    public boolean enMismoCuadrado(int i, int j){

    //Coordenadas
    int i1 = i%tamano;
    int j1 = j%tamano;
    int i2 = i/tamano;
    int j2 = j/tamano;

    //Modulos
    int modi1 = (int) (i1/ sqrt(tamano));
    int modj1 = (int) (j1/ sqrt(tamano));
    int modi2 = (int) (i2/ sqrt(tamano));
    int modj2 = (int) (j2/ sqrt(tamano));

    // Verificamos si las variables estan en el mismo cuadrado
        return (modi1 == modj1) & (modi2 == modj2);
}
```

Código 5.1: Función mismo cuadrado

Las casillas del tablero se numeran entre 0 y $n^2 - 1$ de izquierda a derecha y de arriba hacia abajo. Es necesario por tanto, inicializar las coordenadas de cada casilla dentro del tablero. La coordenada x se consigue calculando el elemento módulo n, mientras que la coordenada y se calcula realizando la división entera por n. Tomando en un tablero 4x4 las casillas 0 y 1, vamos a analizar si estas están en el mismo cuadrado.

Las coordenadas que representan el elemento 0 serían $(0\,\%4, 0/4)$= (0,0) y las de 1 = $(1\,\%4, 1/4)$ = (1,0). Ahora debemos calcular cada x en módulo 2, ya que será el número de cuadrados por fila que habrá en el tablero. Esto es

$mod_{i1} = 0/2 = 0$ y $mod_{j1} = 1/2 = 0$, a su vez hacemos lo mismo con las y para ver si se encuentran en el mismo cuadrado según la columna, $mod_{i2} = 0/2 = 0$ y $mod_{j2} = 0/2 = 0$. Por tanto, todos se encuentran en $[0]_2$, lo que quiere decir que comparten cuadrado.

6

Conclusiones

A lo largo de todo este trabajo hemos exporado el mundo de las bases de Gröbner y el algoritmo de Buchberger para obtenerlas. Hemos analizado qué hay previo a este algoritmo, casos de espacios más simples a los que tratamos posteriormente como son el caso de sistemas de polinomios lineales o polinomios en una sola variable. Los algoritmos que encuentran mejores representaciones para estos espacios sientan las bases del que será el algoritmo que trata sistemas con polinomios más complejos no lineales y en varias variables. Por tanto, las bases de Gröbner cobran un gran poder en la resolución de sistemas de ecuaciones de este tipo.

Además, hemos desarrollado una aplicación que comprende todos los algoritmos y conceptos que se han presentado a lo largo del trabajo. Al haber tenido que implementar todos y cada uno de los métodos necesarios desde lo más básico como pueden ser la suma y resta de monomios, se ha conseguido desgranar al máximo cada concepto que se ha presentado. Es cierto que hoy en día ya contamos con softwares que introduciendo una serie de polinomios, calculan su base de Gröbner como puede ser Python, pero lo que aporta nuestra aplicación es el poder ir analizando lo que va ocurriendo en cada iteración o cada paso del algoritmo, de manera que quien lo utilice pueda comprender en mayor medida cómo funciona el algoritmo.

Estudiando algunas de sus aplicaciones hemos demostrado cómo las bases de Gröbner se pueden aplicar en problemas concretos de grafos como el problema del k-coloreado. Sobre el sistema de ecuaciones planteado para modelizar el problema, hemos ilustrado para el caso $k = 2$ que siempre que tenemos una solución, se

cumplen las condiciones para que esa solución sea una k-coloración. Se plantea como problema abierto demostrarlo para el caso genérico $k = n$.

También hemos visto la aplicación de las bases de Gröbner a lo que para muchos es un pasatiempo, la resolución de sudokus. El algoritmo de Buchberger puede facilitarnos la resolución de sistemas con numerosas ecuaciones como son los sistemas de ecuaciones que modelan los shidokus. Al analizar este problema en detalle hemos visto que resolviendo este tipo de sistemas obtenemos conjuntos que tratan cerca de 2560 pares de elementos, lo que puede resultar desafiante en un ordenador corriente. Esta dificultad se evidencia aún más cuando se consideran sudokus tradicionales de 81 variables. Podemos llegar a contar con conjuntos de casi 400.000 pares de elementos, lo cual se vuelve inviable ante una computadora estándar.

Ante estos resultados, es interesante investigar y considerar alternativas más eficientes para obtener bases de Gröbner cuando se trata de tamaños tan superiores. Uno de estos métodos es el algoritmo F4 donde desaparece la arbitrariedad que existe en el algoritmo de Buchberger al elegir los pares del conjunto. Al reducir el sistema de ecuaciones con unos criterios específicos, se consigue una mayor eficiencia en el proceso.

Por tanto, durante este trabajo hemos podido ver los logros y limitaciones de utilizar el algoritmo de Buchberger como método para obtener bases de Gröbner. Además hemos visto ejemplos prácticos de este algoritmo resolviendo cuestiones de grafos como el problema del k-coloreado y aplicando este mismo método para resolver sudokus.

Bibliografía

[1] W. Adams and P. Loustaunau, *An introduction to Gröbner Bases*. American Mathematical Society, 2000.

[2] B. Z. Zarroca, "Bases de gröbner y su aplicación a la resolución de sudokus," 2021. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=5998394

[3] C. R. A. C. Campos, "Aplicación de bases de gröbner para programación entera y álgebra," 2017. [Online]. Available: http://zaloamati.azc.uam.mx/bitstream/handle/11191/5707/Aplicacion_de_bases_de_Grobner_2017_CastroCampos_DOP.pdf?sequence=1&isAllowed=y

[4] M. Villalba, "Tfg gröbner basis," 2023. [Online]. Available: https://github.com/MariaVillalba23/TFG_Grobner_Basis.git

Apéndices

A

Código de Java

A.1. División univariable

```
public Polinomio[] divisionUnivariable (Polinomio g){
    // Primero simplificamos los polinomios
    this.simplificacionPolinomios();
    g.simplificacionPolinomios();

    // Creamos q y r inicializadas a o y f
    // El polinomio q va a tener tantas variables como f
    int var = 0;
    for(Monomio monomio : this.monomios){
        if(monomio.m_exp.size() > var){
            var = monomio.m_exp.size();
        }
    }
    Polinomio q = new Polinomio(var);
    Polinomio r = this;

    boolean r_vacio = false;
    int grado_r = r.lpPolinomio().exponenteMonomio();

    // Hemos terminado cuando f es vacio o el grado de r
    //es menor que el de g

    while (!r_vacio && grado_r >=
    g.lpPolinomio().exponenteMonomio()){

        // Division de lt(r)/lt(g)
         Monomio m_suma = r.ltPolinomio().divisionMonomios(g.ltPolinomio());
```

```
        Polinomio p_suma = new Polinomio(new ArrayList<>(Arrays.asList(m_suma)));

        // q = q + suma de lt(r)/lt(g)
        q = q.sumaPolinomio(p_suma);

        // r = r - suma de lt(r)/lt(g) * g
         Polinomio p_resta = p_suma.multiplicacionPolinomios(g);
         //para restarlo, le cambio el signo a los coeficientes de los monomios
         for (Monomio monomio : p_resta.monomios) {
              monomio.coeficiente.set(0, monomio.coeficiente.get(0) * (-1));
         }
         r = r.sumaPolinomio(p_resta);

        r_vacio = r.monomios.isEmpty();
        grado_r = r.lpPolinomio().exponenteMonomio();
    }

    q.simplificacionPolinomios();
    r.simplificacionPolinomios();

    Polinomio[] resultado = {q, r};
    return resultado;
}
```

A.2. Algoritmo Euclídeo

```
public Polinomio algoritmoEuclideo(Polinomio p){
        Polinomio[] division;

        Polinomio f = this;
        Polinomio g = p;
        boolean g_vacio = false;

        while(!g_vacio){
         f.simplificacionPolinomios();
         g.simplificacionPolinomios();
            division = f.divisionUnivariable(g);

            f = g;
            g = division[1];

        g_vacio = true;
        if(!g.monomios.isEmpty()){
            for(Monomio m : g.monomios){
            for( Integer i:m.m_exp ){
               if(i != 0){
                   g_vacio = false;
               }
           }
           }
```

```
            }
        }

        ArrayList<Integer> lc_pol = f.lcPolinomio();
        ArrayList<Integer> lc_pol_invertido = new ArrayList<Integer>();
        lc_pol_invertido.add(lc_pol.get(1));
        lc_pol_invertido.add(lc_pol.get(0));


        ArrayList<Integer> d0 = new ArrayList<Integer>(){{
        add(0);
        }};
        Monomio m0 = new Monomio(lc_pol_invertido,d0);
        ArrayList<Monomio> am0 = new ArrayList<Monomio>() {
            {
                add(m0);
            }
        };
        Polinomio p0 = new Polinomio(am0);

        return f.multiplicacionPolinomios(p0);

    }
```

A.3. Algoritmo división multivariable

```
public ArrayList<Polinomio> divisionMultivariable
(ArrayList<Polinomio> arrayPolinomios) {

        int s = arrayPolinomios.size();
        int var_max = 0;
        for (Polinomio p : arrayPolinomios) {
            for (Monomio m : p.monomios) {
                if (m.m_exp.size() > var_max) {
                    var_max = m.m_exp.size();
                }
            }
        }
        Polinomio p = new Polinomio(var_max);
        ArrayList<Polinomio> arrayU = new ArrayList<>();
        for (int i = 0; i < s; i++) {
            arrayU.add(p);
        }

        Polinomio r = new Polinomio(var_max);
        Polinomio h = this;
        boolean h_vacio = false;

        while (!h_vacio) {
            h.simplificacionPolinomios();
            r.simplificacionPolinomios();
```

```
Monomio lp_h = h.lpPolinomio();

Monomio lt_m = h.ltPolinomio();
Polinomio lt_p = new Polinomio(lt_m);

boolean encontrado = false;
int i = 0;
while (!encontrado & i < arrayPolinomios.size()) {
    Monomio lp_f = arrayPolinomios.get(i).lpPolinomio();

    if (lp_h.esDivisible(lp_f)) {
        encontrado = true;
    } else {
        i++;
    }
}

if (encontrado) {
    Monomio lt_f = arrayPolinomios.get(i).ltPolinomio();

    Monomio division_suma_m = lt_m.divisionMonomios(lt_f);
    Monomio division_resta_m = lt_m.divisionMonomios(lt_f);

    Polinomio division_suma_p = new Polinomio(division_suma_m);
    Polinomio division_resta = new Polinomio(division_resta_m);

    arrayU.set(i, arrayU.get(i).sumaPolinomio(division_suma_p));

    for (Monomio monomio : division_resta.monomios) {
        monomio.coeficiente.set
        (0, monomio.coeficiente.get(0) * (-1));
    }
    Polinomio division_resta_p =
    division_resta.multiplicacionPolinomios(arrayPolinomios.get(i));
    h = h.sumaPolinomio(division_resta_p);
  //System.out.println("h = " + h.leerPolinomio());
} else {
    r = r.sumaPolinomio(lt_p);
    //System.out.println("r = " + r.leerPolinomio());
    ArrayList<Monomio> _monomios = new ArrayList<Monomio>();
    for (Monomio monomio : lt_p.monomios) {
        ArrayList<Integer> coef = new ArrayList<Integer>();
        coef.add(monomio.coeficiente.get(0) * (-1));
        coef.add(monomio.coeficiente.get(1));
        Monomio m = new Monomio(coef, monomio.m_exp);
        _monomios.add(m);
    }
    Polinomio resta = new Polinomio(_monomios);
    h = h.sumaPolinomio(resta);
   // System.out.println("h = " + h.leerPolinomio());
}

h_vacio = true;
```

```
            if(!h.monomios.isEmpty()){
                  for(Monomio m : h.monomios){
                      for( Integer n:m.m_exp ){
                          if(n != 0){
                          h_vacio = false;
                           }
                      }
                      if(m.coeficiente.get(0) != 0){h_vacio = false;}
                  }
             }
        }

        arrayU.add(r);

        return arrayU;
    }
```

A.4. Algoritmo Buchberger

```
public ArrayList<Polinomio> calcularBasesGrobner (ArrayList<Polinomio> F){
          int contador = 1;

          //INICIALIZACION
          //Definimos G
          ArrayList<Polinomio> G = F;
          //Definimos el espacio de parejas de todos los polinomios de G
          ArrayList<ArrayList<Polinomio>> G_pares =
          new ArrayList<ArrayList<Polinomio>>();

          for (int i=0; i<G.size();i++){
              for (int j= i + 1; j<G.size();j++){
                  if (i!=j){
                      ArrayList<Polinomio> par = new ArrayList<Polinomio>();
                      par.add(G.get(i));
                      par.add(G.get(j));
                      G_pares.add(par);
                  }
              }
          }

          while (!G_pares.isEmpty()){

              System.out.println();

              System.out.println("PASO " + contador);
              System.out.println(G_pares.size());
              if (contador==12){
                    System.out.println("PASO " + contador);
              }

              contador++;
```

```
//Elegimos un par cualquiera y lo excluimos de G_pares
ArrayList<Polinomio> par_eliminado = G_pares.get(0);
G_pares.remove(0);

//Obtenemos el Spolinomio del par
// System.out.println("Eliminamos el par {" +
par_eliminado.get(0).leerPolinomio() + ", " +
par_eliminado.get(1).leerPolinomio() + "}" );

par_eliminado.get(0).simplificacionPolinomios();
par_eliminado.get(1).simplificacionPolinomios();

Polinomio S = par_eliminado.get(0).
calcularSPolinomio(par_eliminado.get(1));


if (!S.monomios.isEmpty()){
    // System.out.println("El S-Polinomio del par elegido es "
    + S.leerPolinomio());

  //Usamos el algoritmo de division multivariable para
  reducir S con respecto de G

  //Obtenemos el resto de la division (h)
  ArrayList<Polinomio> resultadoDivision =
  S.divisionMultivariable(G);

  Polinomio resto =
  resultadoDivision.get(resultadoDivision.size()-1);

  /*if (resto.monomios.isEmpty()){
      System.out.println("El resto de la division es 0");
  }else{
      System.out.println("El resto de la division es "
      + resto.leerPolinomio());
  }*/
  //Vemos si el resto es distinto de 0
  if (!resto.monomios.isEmpty()){
      //Anadimos a G_pares {{u,resto} | para todo u en G}
      for (Polinomio g:G){
          ArrayList<Polinomio> par = new ArrayList<>();
          par.add(g);
          par.add(resto);
          if(!G_pares.contains(par)){
              G_pares.add(par);
          }
      }
      //Anadimos resto a G
     if(!G.contains(resto)){
         G.add(resto);
      /*
      System.out.println("Los polinomios que estan en G son:");
       for (Polinomio p:G){
```

```
                        System.out.println( p.leerPolinomio());
                    }
                    */
                    }
                }
                if (!resto.monomios.isEmpty()){
                    //Anadimos resto a G
                    if(!G.contains(resto)){
                        G.add(resto);
                    }
                    //Anadimos a G_pares {{u,resto} | para todo u en G}
                    HashSet<Polinomio> G_hash = new HashSet<>(G);
                    // convertir G a HashSet para busqueda rapida
                    for (Polinomio g : G_hash){
                        ArrayList<Polinomio> par = new ArrayList<>();
                        if (!g.equals(resto)){
                            par.add(g);
                            par.add(resto);
                            if(!G_pares.contains(par)){
                                G_pares.add(par);
                            }
                        }
                    }
                }
                }else{
              //System.out.println("El S-Polinomio es 0, G queda igual" );
                }
                /*
                System.out.println("Los pares de polinomios
                que estan en G_par son:");
                for (ArrayList<Polinomio> p:G_pares){
                    System.out.println( "{" + p.get(0).leerPolinomio()
                    + ", " + p.get(1).leerPolinomio() + "}");
                }
                */
            }

            return G;
        }
```

A.5. Algoritmo Reducción

```
public ArrayList<Polinomio> reducirGrobner (ArrayList<Polinomio> G) {
    ArrayList<Polinomio> H = new ArrayList<>();

    //hacemos una copia
     ArrayList<Polinomio> R = new ArrayList<>();
     for (Polinomio p:G){
         Polinomio copia = new Polinomio(p.monomios);
         R.add(copia);
     }
```

```
for(int i=0; i<G.size(); i++){
    Polinomio g = new Polinomio(G.get(i).monomios);
    R.remove(0);

    System.out.println();
      System.out.println("EL POLINOMIO ELIMINADO ES:" + g.leerPolinomio());

      System.out.println();
      System.out.println("H:");

      for (Polinomio p:H){
          System.out.println( p.leerPolinomio());
      }

      System.out.println();
      System.out.println("R:");

      for (Polinomio p:R){
          System.out.println( p.leerPolinomio());
      }


    //Usamos el algoritmo de division multivariable para
    reducir g con respecto de R
     //Obtenemos el resto de la division (h)
      ArrayList<Polinomio> resultadoDivision = g.divisionMultivariable(R);
      Polinomio resto = resultadoDivision.get(resultadoDivision.size()-1);
      if (!resto.monomios.isEmpty()){

          System.out.println("El resto de la division es "
          + resto.leerPolinomio());

          //Dividimos el resto por su lc para que sea monico

          ArrayList<Integer> lc_double = new ArrayList<>(resto.lcPolinomio());

          for(Monomio m: resto.monomios){
              ArrayList<Integer> monico =
              m.divisionFraccion(m.coeficiente,lc_double);
              m.coeficiente.set(0, monico.get(0));
              m.coeficiente.set(1, monico.get(1));
          }

          resto.simplificacionPolinomios();
          //Metemos el resto en la base reducida
          H.add(resto);

          //Metemos el resto en la base con la que estamos trabajando
          R.add(resto);

      }
```

```
  }

  return H;
}
```

B

Ecuaciones Shidoku

B.1. Ecuaciones de pistas

$$
\begin{aligned}
x_2 - 2 \\
x_3 - 3 \\
x_5 - 1 \\
x_8 - 4 \\
x_9 - 3 \\
x_{12} - 2 \\
x_{14} - 4 \\
x_{15} - 1
\end{aligned}
$$

B.2. Ecuaciones $F(x_i)$

$$
\begin{aligned}
x_1^4 - 10x_1^3 + 35x_1^2 - 50x_1 + 24 \\
x_4^4 - 10x_4^3 + 35x_4^2 - 50x_4 + 24 \\
x_6^4 - 10x_6^3 + 35x_6^2 - 50x_6 + 24 \\
x_7^4 - 10x_7^3 + 35x_7^2 - 50x_7 + 24 \\
x_{10}^4 - 10x_{10}^3 + 35x_{10}^2 - 50x_{10} + 24 \\
x_{11}^4 - 10x_{11}^3 + 35x_{11}^2 - 50x_{11} + 24 \\
x_{13}^4 - 10x_{13}^3 + 35x_{13}^2 - 50x_{13} + 24 \\
x_{16}^4 - 10x_{16}^3 + 35x_{16}^2 - 50x_{16} + 24
\end{aligned}
$$

B.3. Ecuaciones $G(x_i, x_j)$

$$
\begin{aligned}
x_1^3 + x_2^3 + x_1^2 x_2 + x_1 x_2^2 - 10x_1^2 - 10x_2^2 - 10x_1 x_2 + 35x_1 + 35x_2 - 50 \\
x_1^3 + x_3^3 + x_1^2 x_3 + x_1 x_3^2 - 10x_1^2 - 10x_3^2 - 10x_1 x_3 + 35x_1 + 35x_3 - 50 \\
x_1^3 + x_4^3 + x_1^2 x_4 + x_1 x_4^2 - 10x_1^2 - 10x_4^2 - 10x_1 x_4 + 35x_1 + 35x_4 - 50 \\
x_2^3 + x_3^3 + x_2^2 x_3 + x_2 x_3^2 - 10x_2^2 - 10x_3^2 - 10x_2 x_3 + 35x_2 + 35x_3 - 50 \\
x_2^3 + x_4^3 + x_2^2 x_4 + x_2 x_4^2 - 10x_2^2 - 10x_4^2 - 10x_2 x_4 + 35x_2 + 35x_4 - 50 \\
x_3^3 + x_4^3 + x_3^2 x_4 + x_3 x_4^2 - 10x_3^2 - 10x_4^2 - 10x_3 x_4 + 35x_3 + 35x_4 - 50 \\
x_5^3 + x_6^3 + x_5^2 x_6 + x_5 x_6^2 - 10x_5^2 - 10x_6^2 - 10x_5 x_6 + 35x_5 + 35x_6 - 50 \\
x_5^3 + x_7^3 + x_5^2 x_7 + x_5 x_7^2 - 10x_5^2 - 10x_7^2 - 10x_5 x_7 + 35x_5 + 35x_7 - 50 \\
x_5^3 + x_8^3 + x_5^2 x_8 + x_5 x_8^2 - 10x_5^2 - 10x_8^2 - 10x_5 x_8 + 35x_5 + 35x_8 - 50 \\
x_6^3 + x_7^3 + x_6^2 x_7 + x_6 x_7^2 - 10x_6^2 - 10x_7^2 - 10x_6 x_7 + 35x_6 + 35x_7 - 50 \\
x_6^3 + x_8^3 + x_6^2 x_8 + x_6 x_8^2 - 10x_6^2 - 10x_8^2 - 10x_6 x_8 + 35x_6 + 35x_8 - 50 \\
x_7^3 + x_8^3 + x_7^2 x_8 + x_7 x_8^2 - 10x_7^2 - 10x_8^2 - 10x_7 x_8 + 35x_7 + 35x_8 - 50 \\
x_9^3 + x_{10}^3 + x_9^2 x_{10} + x_9 x_{10}^2 - 10x_9^2 - 10x_{10}^2 - 10x_9 x_{10} + 35x_9 + 35x_{10} - 50 \\
x_9^3 + x_{11}^3 + x_9^2 x_{11} + x_9 x_{11}^2 - 10x_9^2 - 10x_{11}^2 - 10x_9 x_{11} + 35x_9 + 35x_{11} - 50 \\
x_9^3 + x_{12}^3 + x_9^2 x_{12} + x_9 x_{12}^2 - 10x_9^2 - 10x_{12}^2 - 10x_9 x_{12} + 35x_9 + 35x_{12} - 50 \\
x_{10}^3 + x_{11}^3 + x_{10}^2 x_{11} + x_{10} x_{11}^2 - 10x_{10}^2 - 10x_{11}^2 - 10x_{10} x_{11} + 35x_{10} + 35x_{11} - 50 \\
x_{10}^3 + x_{12}^3 + x_{10}^2 x_{12} + x_{10} x_{12}^2 - 10x_{10}^2 - 10x_{12}^2 - 10x_{10} x_{12} + 35x_{10} + 35x_{12} - 50 \\
x_{11}^3 + x_{12}^3 + x_{11}^2 x_{12} + x_{11} x_{12}^2 - 10x_{11}^2 - 10x_{12}^2 - 10x_{11} x_{12} + 35x_{11} + 35x_{12} - 50 \\
x_{13}^3 + x_{14}^3 + x_{13}^2 x_{14} + x_{13} x_{14}^2 - 10x_{13}^2 - 10x_{14}^2 - 10x_{13} x_{14} + 35x_{13} + 35x_{14} - 50 \\
x_{13}^3 + x_{15}^3 + x_{13}^2 x_{15} + x_{13} x_{15}^2 - 10x_{13}^2 - 10x_{15}^2 - 10x_{13} x_{15} + 35x_{13} + 35x_{15} - 50 \\
x_{13}^3 + x_{16}^3 + x_{13}^2 x_{16} + x_{13} x_{16}^2 - 10x_{13}^2 - 10x_{16}^2 - 10x_{13} x_{16} + 35x_{13} + 35x_{16} - 50 \\
x_{14}^3 + x_{15}^3 + x_{14}^2 x_{15} + x_{14} x_{15}^2 - 10x_{14}^2 - 10x_{15}^2 - 10x_{14} x_{15} + 35x_{14} + 35x_{15} - 50 \\
x_{14}^3 + x_{16}^3 + x_{14}^2 x_{16} + x_{14} x_{16}^2 - 10x_{14}^2 - 10x_{16}^2 - 10x_{14} x_{16} + 35x_{14} + 35x_{16} - 50 \\
x_{15}^3 + x_{16}^3 + x_{15}^2 x_{16} + x_{15} x_{16}^2 - 10x_{15}^2 - 10x_{16}^2 - 10x_{15} x_{16} + 35x_{15} + 35x_{16} - 50 \\
x_1^3 + x_5^3 + x_1^2 x_5 + x_1 x_5^2 - 10x_1^2 - 10x_5^2 - 10x_1 x_5 + 35x_1 + 35x_5 - 50 \\
x_1^3 + x_9^3 + x_1^2 x_9 + x_1 x_9^2 - 10x_1^2 - 10x_9^2 - 10x_1 x_9 + 35x_1 + 35x_9 - 50 \\
x_1^3 + x_{13}^3 + x_1^2 x_{13} + x_1 x_{13}^2 - 10x_1^2 - 10x_{13}^2 - 10x_1 x_{13} + 35x_1 + 35x_{13} - 50 \\
x_2^3 + x_6^3 + x_2^2 x_6 + x_2 x_6^2 - 10x_2^2 - 10x_6^2 - 10x_2 x_6 + 35x_2 + 35x_6 - 50 \\
x_2^3 + x_{10}^3 + x_2^2 x_{10} + x_2 x_{10}^2 - 10x_2^2 - 10x_{10}^2 - 10x_2 x_{10} + 35x_2 + 35x_{10} - 50 \\
x_2^3 + x_{14}^3 + x_2^2 x_{14} + x_2 x_{14}^2 - 10x_2^2 - 10x_{14}^2 - 10x_2 x_{14} + 35x_2 + 35x_{14} - 50 \\
x_3^3 + x_7^3 + x_3^2 x_7 + x_3 x_7^2 - 10x_3^2 - 10x_7^2 - 10x_3 x_7 + 35x_3 + 35x_7 - 50
\end{aligned}
$$

$$x_3^3 + x_{11}^3 + x_3^2 x_{11} + x_3 x_{11}^2 - 10x_3^2 - 10x_{11}^2 - 10x_3 x_{11} + 35x_3 + 35x_{11} - 50$$
$$x_3^3 + x_{15}^3 + x_3^2 x_{15} + x_3 x_{15}^2 - 10x_3^2 - 10x_{15}^2 - 10x_3 x_{15} + 35x_3 + 35x_{15} - 50$$
$$x_4^3 + x_8^3 + x_4^2 x_8 + x_4 x_8^2 - 10x_4^2 - 10x_8^2 - 10x_4 x_8 + 35x_4 + 35x_8 - 50$$
$$x_4^3 + x_{12}^3 + x_4^2 x_{12} + x_4 x_{12}^2 - 10x_4^2 - 10x_{12}^2 - 10x_4 x_{12} + 35x_4 + 35x_{12} - 50$$
$$x_4^3 + x_{16}^3 + x_4^2 x_{16} + x_4 x_{16}^2 - 10x_4^2 - 10x_{16}^2 - 10x_4 x_{16} + 35x_4 + 35x_{16} - 50$$
$$x_5^3 + x_9^3 + x_5^2 x_9 + x_5 x_9^2 - 10x_5^2 - 10x_9^2 - 10x_5 x_9 + 35x_5 + 35x_9 - 50$$
$$x_5^3 + x_{13}^3 + x_5^2 x_{13} + x_5 x_{13}^2 - 10x_5^2 - 10x_{13}^2 - 10x_5 x_{13} + 35x_5 + 35x_{13} - 50$$
$$x_6^3 + x_{10}^3 + x_6^2 x_{10} + x_6 x_{10}^2 - 10x_6^2 - 10x_{10}^2 - 10x_6 x_{10} + 35x_6 + 35x_{10} - 50$$
$$x_6^3 + x_{14}^3 + x_6^2 x_{14} + x_6 x_{14}^2 - 10x_6^2 - 10x_{14}^2 - 10x_6 x_{14} + 35x_6 + 35x_{14} - 50$$
$$x_7^3 + x_{11}^3 + x_7^2 x_{11} + x_7 x_{11}^2 - 10x_7^2 - 10x_{11}^2 - 10x_7 x_{11} + 35x_7 + 35x_{11} - 50$$
$$x_7^3 + x_{15}^3 + x_7^2 x_{15} + x_7 x_{15}^2 - 10x_7^2 - 10x_{15}^2 - 10x_7 x_{15} + 35x_7 + 35x_{15} - 50$$
$$x_8^3 + x_{12}^3 + x_8^2 x_{12} + x_8 x_{12}^2 - 10x_8^2 - 10x_{12}^2 - 10x_8 x_{12} + 35x_8 + 35x_{12} - 50$$
$$x_8^3 + x_{16}^3 + x_8^2 x_{16} + x_8 x_{16}^2 - 10x_8^2 - 10x_{16}^2 - 10x_8 x_{16} + 35x_8 + 35x_{16} - 50$$
$$x_9^3 + x_{13}^3 + x_9^2 x_{13} + x_9 x_{13}^2 - 10x_9^2 - 10x_{13}^2 - 10x_9 x_{13} + 35x_9 + 35x_{13} - 50$$
$$x_{10}^3 + x_{14}^3 + x_{10}^2 x_{14} + x_{10} x_{14}^2 - 10x_{10}^2 - 10x_{14}^2 - 10x_{10} x_{14} + 35x_{10} + 35x_{14} - 50$$
$$x_{11}^3 + x_{15}^3 + x_{11}^2 x_{15} + x_{11} x_{15}^2 - 10x_{11}^2 - 10x_{15}^2 - 10x_{11} x_{15} + 35x_{11} + 35x_{15} - 50$$
$$x_{12}^3 + x_{16}^3 + x_{12}^2 x_{16} + x_{12} x_{16}^2 - 10x_{12}^2 - 10x_{16}^2 - 10x_{12} x_{16} + 35x_{12} + 35x_{16} - 50$$
$$x_1^3 + x_6^3 + x_1^2 x_6 + x_1 x_6^2 - 10x_1^2 - 10x_6^2 - 10x_1 x_6 + 35x_1 + 35x_6 - 50$$
$$x_2^3 + x_5^3 + x_2^2 x_5 + x_2 x_5^2 - 10x_2^2 - 10x_5^2 - 10x_2 x_5 + 35x_2 + 35x_5 - 50$$
$$x_3^3 + x_8^3 + x_3^2 x_8 + x_3 x_8^2 - 10x_3^2 - 10x_8^2 - 10x_3 x_8 + 35x_3 + 35x_8 - 50$$
$$x_4^3 + x_7^3 + x_4^2 x_7 + x_4 x_7^2 - 10x_4^2 - 10x_7^2 - 10x_4 x_7 + 35x_4 + 35x_7 - 50$$
$$x_9^3 + x_{14}^3 + x_9^2 x_{14} + x_9 x_{14}^2 - 10x_9^2 - 10x_{14}^2 - 10x_9 x_{14} + 35x_9 + 35x_{14} - 50$$
$$x_{10}^3 + x_{13}^3 + x_{10}^2 x_{13} + x_{10} x_{13}^2 - 10x_{10}^2 - 10x_{13}^2 - 10x_{10} x_{13} + 35x_{10} + 35x_{13} - 50$$
$$x_{11}^3 + x_{16}^3 + x_{11}^2 x_{16} + x_{11} x_{16}^2 - 10x_{11}^2 - 10x_{16}^2 - 10x_{11} x_{16} + 35x_{11} + 35x_{16} - 50$$
$$x_{12}^3 + x_{15}^3 + x_{12}^2 x_{15} + x_{12} x_{15}^2 - 10x_{12}^2 - 10x_{15}^2 - 10x_{12} x_{15} + 35x_{12} + 35x_{15} - 50$$

C

Ecuaciones Shidoku Killer

C.1. Ecuaciones Killer

$$
\begin{aligned}
x_1 + x_2 - 6 \\
x_3 + x_4 + x_7 - 6 \\
x_5 + x_6 + x_9 + x_10 - 8 \\
x_8 + x_12 + x_16 - 9 \\
x_11 + x_13 + x_14 + x_15 - 11
\end{aligned}
$$

C.2. Ecuaciones de pistas

$$
\begin{aligned}
x_1 - 4 \\
x_5 - 1 \\
x_11 - 4 \\
x_12 - 2 \\
x_14 - 4 \\
x_15 - 1
\end{aligned}
$$

C.3. Ecuaciones $F(x_i)$

$$
\begin{aligned}
x_2^4 - 10x_2^3 + 35x_2^2 - 50x_2 + 24 \\
x_3^4 - 10x_3^3 + 35x_3^2 - 50x_3 + 24 \\
x_4^4 - 10x_4^3 + 35x_4^2 - 50x_4 + 24 \\
x_6^4 - 10x_6^3 + 35x_6^2 - 50x_6 + 24 \\
x_7^4 - 10x_7^3 + 35x_7^2 - 50x_7 + 24 \\
x_8^4 - 10x_8^3 + 35x_8^2 - 50x_8 + 24 \\
x_9^4 - 10x_9^3 + 35x_9^2 - 50x_9 + 24 \\
x_10^4 - 10x_10^3 + 35x_10^2 - 50x_10 + 24 \\
x_13^4 - 10x_13^3 + 35x_13^2 - 50x_13 + 24 \\
x_16^4 - 10x_16^3 + 35x_16^2 - 50x_16 + 24
\end{aligned}
$$

C.4. Ecuaciones $G(x_i, x_j)$

$$x_1^3 + x_2^3 + x_1^2x_2 + x_1x_2^2 - 10x_1^2 - 10x_2^2 - 10x_1x_2 + 35x_1 + 35x_2 - 50$$
$$x_1^3 + x_3^3 + x_1^2x_3 + x_1x_3^2 - 10x_1^2 - 10x_3^2 - 10x_1x_3 + 35x_1 + 35x_3 - 50$$
$$x_1^3 + x_4^3 + x_1^2x_4 + x_1x_4^2 - 10x_1^2 - 10x_4^2 - 10x_1x_4 + 35x_1 + 35x_4 - 50$$
$$1x_2^3 + x_3^3 + x_2^2x_3 + x_2x_3^2 - 10x_2^2 - 10x_3^2 - 10x_2x_3 + 35x_2 + 35x_3 - 50$$
$$1x_2^3 + x_4^3 + x_2^2x_4 + x_2x_4^2 - 10x_2^2 - 10x_4^2 - 10x_2x_4 + 35x_2 + 35x_4 - 50$$
$$1x_3^3 + x_4^3 + x_3^2x_4 + x_3x_4^2 - 10x_3^2 - 10x_4^2 - 10x_3x_4 + 35x_3 + 35x_4 - 50$$
$$1x_5^3 + x_6^3 + x_5^2x_6 + x_5x_6^2 - 10x_5^2 - 10x_6^2 - 10x_5x_6 + 35x_5 + 35x_6 - 50$$
$$1x_5^3 + x_7^3 + x_5^2x_7 + x_5x_7^2 - 10x_5^2 - 10x_7^2 - 10x_5x_7 + 35x_5 + 35x_7 - 50$$
$$1x_5^3 + x_8^3 + x_5^2x_8 + x_5x_8^2 - 10x_5^2 - 10x_8^2 - 10x_5x_8 + 35x_5 + 35x_8 - 50$$
$$1x_6^3 + x_7^3 + x_6^2x_7 + x_6x_7^2 - 10x_6^2 - 10x_7^2 - 10x_6x_7 + 35x_6 + 35x_7 - 50$$
$$1x_6^3 + x_8^3 + x_6^2x_8 + x_6x_8^2 - 10x_6^2 - 10x_8^2 - 10x_6x_8 + 35x_6 + 35x_8 - 50$$
$$1x_7^3 + x_8^3 + x_7^2x_8 + x_7x_8^2 - 10x_7^2 - 10x_8^2 - 10x_7x_8 + 35x_7 + 35x_8 - 50$$
$$1x_9^3 + x_{10}^3 + x_9^2x_10 + x_9x_10^2 - 10x_9^2 - 10x_10^2 - 10x_9x_10 + 35x_9 + 35x_10 - 50$$
$$1x_9^3 + x_{11}^3 + x_9^2x_11 + x_9x_11^2 - 10x_9^2 - 10x_11^2 - 10x_9x_11 + 35x_9 + 35x_11 - 50$$
$$1x_9^3 + x_{12}^3 + x_9^2x_12 + x_9x_12^2 - 10x_9^2 - 10x_12^2 - 10x_9x_12 + 35x_9 + 35x_12 - 50$$
$$x_{10}^3 + x_{11}^3 + x_{10}^2x_11 + x_10x_11^2 - 10x_10^2 - 10x_11^2 - 10x_10x_11 + 35x_10 + 35x_11 - 50$$
$$x_{10}^3 + x_{12}^3 + x_{10}^2x_12 + x_10x_12^2 - 10x_10^2 - 10x_12^2 - 10x_10x_12 + 35x_10 + 35x_12 - 50$$
$$x_{11}^3 + x_{12}^3 + x_{11}^2x_12 + x_11x_12^2 - 10x_11^2 - 10x_12^2 - 10x_11x_12 + 35x_11 + 35x_12 - 50$$
$$x_{13}^3 + x_{14}^3 + x_{13}^2x_14 + x_13x_14^2 - 10x_13^2 - 10x_14^2 - 10x_13x_14 + 35x_13 + 35x_14 - 50$$
$$x_{13}^3 + x_{15}^3 + x_{13}^2x_15 + x_13x_15^2 - 10x_13^2 - 10x_15^2 - 10x_13x_15 + 35x_13 + 35x_15 - 50$$
$$x_{13}^3 + x_{16}^3 + x_{13}^2x_16 + x_13x_16^2 - 10x_13^2 - 10x_16^2 - 10x_13x_16 + 35x_13 + 35x_16 - 50$$
$$x_{14}^3 + x_{15}^3 + x_{14}^2x_15 + x_14x_15^2 - 10x_14^2 - 10x_15^2 - 10x_14x_15 + 35x_14 + 35x_15 - 50$$
$$x_{14}^3 + x_{16}^3 + x_{14}^2x_16 + x_14x_16^2 - 10x_14^2 - 10x_16^2 - 10x_14x_16 + 35x_14 + 35x_16 - 50$$
$$x_{15}^3 + x_{16}^3 + x_{15}^2x_16 + x_15x_16^2 - 10x_15^2 - 10x_16^2 - 10x_15x_16 + 35x_15 + 35x_16 - 50$$
$$x_1^3 + x_5^3 + x_1^2x_5 + x_1x_5^2 - 10x_1^2 - 10x_5^2 - 10x_1x_5 + 35x_1 + 35x_5 - 50$$
$$x_1^3 + x_9^3 + x_1^2x_9 + x_1x_9^2 - 10x_1^2 - 10x_9^2 - 10x_1x_9 + 35x_1 + 35x_9 - 50$$
$$x_1^3 + x_{13}^3 + x_1^2x_13 + x_1x_{13}^2 - 10x_1^2 - 10x_13^2 - 10x_1x_{13} + 35x_1 + 35x_13 - 50$$

$$1x_2^3 + x_6^3 + x_2^2x_6 + x_2x_6^2 - 10x_2^2 - 10x_6^2 - 10x_2x_6 + 35x_2 + 35x_6 - 50$$
$$1x_2^3 + x_{10}^3 + x_2^2x_10 + x_2x_10^2 - 10x_2^2 - 10x_10^2 - 10x_2x_10 + 35x_2 + 35x_10 - 50$$
$$1x_2^3 + x_{14}^3 + x_2^2x_14 + x_2x_14^2 - 10x_2^2 - 10x_14^2 - 10x_2x_14 + 35x_2 + 35x_14 - 50$$
$$1x_3^3 + x_7^3 + x_3^2x_7 + x_3x_7^2 - 10x_3^2 - 10x_7^2 - 10x_3x_7 + 35x_3 + 35x_7 - 50$$
$$1x_3^3 + x_{11}^3 + x_3^2x_11 + x_3x_11^2 - 10x_3^2 - 10x_11^2 - 10x_3x_11 + 35x_3 + 35x_11 - 50$$
$$1x_3^3 + x_{15}^3 + x_3^2x_15 + x_3x_15^2 - 10x_3^2 - 10x_15^2 - 10x_3x_15 + 35x_3 + 35x_15 - 50$$
$$1x_4^3 + x_8^3 + x_4^2x_8 + x_4x_8^2 - 10x_4^2 - 10x_8^2 - 10x_4x_8 + 35x_4 + 35x_8 - 50$$
$$1x_4^3 + x_{12}^3 + x_4^2x_12 + x_4x_12^2 - 10x_4^2 - 10x_12^2 - 10x_4x_12 + 35x_4 + 35x_12 - 50$$
$$1x_4^3 + x_{16}^3 + x_4^2x_16 + x_4x_16^2 - 10x_4^2 - 10x_16^2 - 10x_4x_16 + 35x_4 + 35x_16 - 50$$
$$1x_5^3 + x_9^3 + x_5^2x_9 + x_5x_9^2 - 10x_5^2 - 10x_9^2 - 10x_5x_9 + 35x_5 + 35x_9 - 50$$
$$1x_5^3 + x_{13}^3 + x_5^2x_13 + x_5x_13^2 - 10x_5^2 - 10x_13^2 - 10x_5x_13 + 35x_5 + 35x_13 - 50$$
$$1x_6^3 + x_{10}^3 + x_6^2x_10 + x_6x_10^2 - 10x_6^2 - 10x_10^2 - 10x_6x_10 + 35x_6 + 35x_10 - 50$$
$$1x_6^3 + x_{14}^3 + x_6^2x_14 + x_6x_14^2 - 10x_6^2 - 10x_14^2 - 10x_6x_14 + 35x_6 + 35x_14 - 50$$
$$1x_7^3 + x_{11}^3 + x_7^2x_11 + x_7x_11^2 - 10x_7^2 - 10x_11^2 - 10x_7x_11 + 35x_7 + 35x_11 - 50$$
$$1x_7^3 + x_{15}^3 + x_7^2x_15 + x_7x_15^2 - 10x_7^2 - 10x_15^2 - 10x_7x_15 + 35x_7 + 35x_15 - 50$$
$$1x_8^3 + x_{12}^3 + x_8^2x_12 + x_8x_12^2 - 10x_8^2 - 10x_12^2 - 10x_8x_12 + 35x_8 + 35x_12 - 50$$
$$1x_8^3 + x_{16}^3 + x_8^2x_16 + x_8x_16^2 - 10x_8^2 - 10x_16^2 - 10x_8x_16 + 35x_8 + 35x_16 - 50$$
$$1x_9^3 + x_{13}^3 + x_9^2x_13 + x_9x_13^2 - 10x_9^2 - 10x_13^2 - 10x_9x_13 + 35x_9 + 35x_13 - 50$$
$$x_{10}^3 + x_{14}^3 + x_{10}^2x_14 + x_{10}x_14^2 - 10x_10^2 - 10x_14^2 - 10x_10x_14 + 35x_10 + 35x_14 - 50$$
$$x_{11}^3 + x_{15}^3 + x_{11}^2x_15 + x_{11}x_15^2 - 10x_11^2 - 10x_15^2 - 10x_11x_15 + 35x_11 + 35x_15 - 50$$
$$x_{12}^3 + x_{16}^3 + x_{12}^2x_16 + x_{12}x_16^2 - 10x_12^2 - 10x_16^2 - 10x_12x_16 + 35x_12 + 35x_16 - 50$$
$$x_1^3 + x_6^3 + x_1^2x_6 + x_1x_6^2 - 10x_1^2 - 10x_6^2 - 10x_1x_6 + 35x_1 + 35x_6 - 50$$
$$1x_2^3 + x_5^3 + x_2^2x_5 + x_2x_5^2 - 10x_2^2 - 10x_5^2 - 10x_2x_5 + 35x_2 + 35x_5 - 50$$
$$1x_3^3 + x_8^3 + x_3^2x_8 + x_3x_8^2 - 10x_3^2 - 10x_8^2 - 10x_3x_8 + 35x_3 + 35x_8 - 50$$
$$1x_4^3 + x_7^3 + x_4^2x_7 + x_4x_7^2 - 10x_4^2 - 10x_7^2 - 10x_4x_7 + 35x_4 + 35x_7 - 50$$
$$1x_9^3 + x_{14}^3 + x_9^2x_14 + x_9x_14^2 - 10x_9^2 - 10x_14^2 - 10x_9x_14 + 35x_9 + 35x_14 - 50$$
$$x_{10}^3 + x_{13}^3 + x_{10}^2x_13 + x_{10}x_13^2 - 10x_10^2 - 10x_13^2 - 10x_10x_13 + 35x_10 + 35x_13 - 50$$
$$x_{11}^3 + x_{16}^3 + x_{11}^2x_16 + x_{11}x_16^2 - 10x_11^2 - 10x_16^2 - 10x_11x_16 + 35x_11 + 35x_16 - 50$$
$$x_{12}^3 + x_{15}^3 + x_{12}^2x_15 + x_{12}x_15^2 - 10x_12^2 - 10x_15^2 - 10x_12x_15 + 35x_12 + 35x_15 - 50$$

D

Ecuaciones Shidoku Diagonal

D.1. Ecuaciones de pistas

$$
\begin{aligned}
x_2 - 4 \\
x_3 - 2 \\
x_4 - 3 \\
x_7 - 1 \\
x_{10} - 2 \\
x_{12} - 1 \\
x_{13} - 4
\end{aligned}
$$

D.2. Ecuaciones $F(x_i)$

$$
\begin{aligned}
x_1^4 - 10x_1^3 + 35x_1^2 - 50x_1 + 24 \\
x_5^4 - 10x_5^3 + 35x_5^2 - 50x_5 + 24 \\
x_6^4 - 10x_6^3 + 35x_6^2 - 50x_6 + 24 \\
x_8^4 - 10x_8^3 + 35x_8^2 - 50x_8 + 24 \\
x_9^4 - 10x_9^3 + 35x_9^2 - 50x_9 + 24 \\
x_{11}^4 - 10x_{11}^3 + 35x_{11}^2 - 50x_{11} + 24 \\
x_{14}^4 - 10x_{14}^3 + 35x_{14}^2 - 50x_{14} + 24 \\
x_{15}^4 - 10x_{15}^3 + 35x_{15}^2 - 50x_{15} + 24 \\
x_{16}^4 - 10x_{16}^3 + 35x_{16}^2 - 50x_{16} + 24
\end{aligned}
$$

D.3. Ecuaciones $G(x_i, x_j)$

$$x_1^3 + x_2^3 + x_1^2x_2 + x_1x_2^2 - 10x_1^2 - 10x_2^2 - 10x_1x_2 + 35x_1 + 35x_2 - 50$$
$$x_1^3 + x_3^3 + x_1^2x_3 + x_1x_3^2 - 10x_1^2 - 10x_3^2 - 10x_1x_3 + 35x_1 + 35x_3 - 50$$
$$x_1^3 + x_4^3 + x_1^2x_4 + x_1x_4^2 - 10x_1^2 - 10x_4^2 - 10x_1x_4 + 35x_1 + 35x_4 - 50$$
$$x_2^3 + x_3^3 + x_2^2x_3 + x_2x_3^2 - 10x_2^2 - 10x_3^2 - 10x_2x_3 + 35x_2 + 35x_3 - 50$$
$$x_2^3 + x_4^3 + x_2^2x_4 + x_2x_4^2 - 10x_2^2 - 10x_4^2 - 10x_2x_4 + 35x_2 + 35x_4 - 50$$
$$x_3^3 + x_4^3 + x_3^2x_4 + x_3x_4^2 - 10x_3^2 - 10x_4^2 - 10x_3x_4 + 35x_3 + 35x_4 - 50$$
$$x_5^3 + x_6^3 + x_5^2x_6 + x_5x_6^2 - 10x_5^2 - 10x_6^2 - 10x_5x_6 + 35x_5 + 35x_6 - 50$$
$$x_5^3 + x_7^3 + x_5^2x_7 + x_5x_7^2 - 10x_5^2 - 10x_7^2 - 10x_5x_7 + 35x_5 + 35x_7 - 50$$
$$x_5^3 + x_8^3 + x_5^2x_8 + x_5x_8^2 - 10x_5^2 - 10x_8^2 - 10x_5x_8 + 35x_5 + 35x_8 - 50$$
$$x_6^3 + x_7^3 + x_6^2x_7 + x_6x_7^2 - 10x_6^2 - 10x_7^2 - 10x_6x_7 + 35x_6 + 35x_7 - 50$$
$$x_6^3 + x_8^3 + x_6^2x_8 + x_6x_8^2 - 10x_6^2 - 10x_8^2 - 10x_6x_8 + 35x_6 + 35x_8 - 50$$
$$x_7^3 + x_8^3 + x_7^2x_8 + x_7x_8^2 - 10x_7^2 - 10x_8^2 - 10x_7x_8 + 35x_7 + 35x_8 - 50$$
$$x_9^3 + x_{10}^3 + x_9^2x_{10} + x_9x_{10}^2 - 10x_9^2 - 10x_{10}^2 - 10x_9x_{10} + 35x_9 + 35x_{10} - 50$$
$$x_9^3 + x_{11}^3 + x_9^2x_{11} + x_9x_{11}^2 - 10x_9^2 - 10x_{11}^2 - 10x_9x_{11} + 35x_9 + 35x_{11} - 50$$
$$x_9^3 + x_{12}^3 + x_9^2x_{12} + x_9x_{12}^2 - 10x_9^2 - 10x_{12}^2 - 10x_9x_{12} + 35x_9 + 35x_{12} - 50$$
$$x_{10}^3 + x_{11}^3 + x_{10}^2x_{11} + x_{10}x_{11}^2 - 10x_{10}^2 - 10x_{11}^2 - 10x_{10}x_{11} + 35x_{10} + 35x_{11} - 50$$
$$x_{10}^3 + x_{12}^3 + x_{10}^2x_{12} + x_{10}x_{12}^2 - 10x_{10}^2 - 10x_{12}^2 - 10x_{10}x_{12} + 35x_{10} + 35x_{12} - 50$$
$$x_{11}^3 + x_{12}^3 + x_{11}^2x_{12} + x_1x_{12}^2 - 10x_{11}^2 - 10x_{12}^2 - 10x_1x_{12} + 35x_{11} + 35x_{12} - 50$$
$$x_{13}^3 + x_{14}^3 + x_{13}^2x_{14} + x_{13}x_{14}^2 - 10x_{13}^2 - 10x_{14}^2 - 10x_{13}x_{14} + 35x_{13} + 35x_{14} - 50$$
$$x_{13}^3 + x_{15}^3 + x_{13}^2x_{15} + x_{13}x_{15}^2 - 10x_{13}^2 - 10x_{15}^2 - 10x_{13}x_{15} + 35x_{13} + 35x_{15} - 50$$
$$x_{13}^3 + x_{16}^3 + x_{13}^2x_{16} + x_{13}x_{16}^2 - 10x_{13}^2 - 10x_{16}^2 - 10x_{13}x_{16} + 35x_{13} + 35x_{16} - 50$$
$$x_{14}^3 + x_{15}^3 + x_{14}^2x_{15} + x_{14}x_{15}^2 - 10x_{14}^2 - 10x_{15}^2 - 10x_{14}x_{15} + 35x_{14} + 35x_{15} - 50$$
$$x_{14}^3 + x_{16}^3 + x_{14}^2x_{16} + x_{14}x_{16}^2 - 10x_{14}^2 - 10x_{16}^2 - 10x_{14}x_{16} + 35x_{14} + 35x_{16} - 50$$
$$x_{15}^3 + x_{16}^3 + x_{15}^2x_{16} + x_{15}x_{16}^2 - 10x_{15}^2 - 10x_{16}^2 - 10x_{15}x_{16} + 35x_{15} + 35x_{16} - 50$$
$$x_1^3 + x_5^3 + x_1^2x_5 + x_1x_5^2 - 10x_1^2 - 10x_5^2 - 10x_1x_5 + 35x_1 + 35x_5 - 50$$
$$x_1^3 + x_9^3 + x_1^2x_9 + x_1x_9^2 - 10x_1^2 - 10x_9^2 - 10x_1x_9 + 35x_1 + 35x_9 - 50$$
$$x_1^3 + x_{13}^3 + x_1^2x_{13} + x_1x_{13}^2 - 10x_1^2 - 10x_{13}^2 - 10x_1x_{13} + 35x_1 + 35x_{13} - 50$$
$$x_2^3 + x_6^3 + x_2^2x_6 + x_2x_6^2 - 10x_2^2 - 10x_6^2 - 10x_2x_6 + 35x_2 + 35x_6 - 50$$
$$x_2^3 + x_{10}^3 + x_2^2x_{10} + x_2x_{10}^2 - 10x_2^2 - 10x_{10}^2 - 10x_2x_{10} + 35x_2 + 35x_{10} - 50$$
$$x_2^3 + x_{14}^3 + x_2^2x_{14} + x_2x_{14}^2 - 10x_2^2 - 10x_{14}^2 - 10x_2x_{14} + 35x_2 + 35x_{14} - 50$$
$$x_3^3 + x_7^3 + x_3^2x_7 + x_3x_7^2 - 10x_3^2 - 10x_7^2 - 10x_3x_7 + 35x_3 + 35x_7 - 50$$
$$x_3^3 + x_{11}^3 + x_3^2x_{11} + x_3x_{11}^2 - 10x_3^2 - 10x_{11}^2 - 10x_3x_{11} + 35x_3 + 35x_{11} - 50$$
$$x_3^3 + x_{15}^3 + x_3^2x_{15} + x_3x_{15}^2 - 10x_3^2 - 10x_{15}^2 - 10x_3x_{15} + 35x_3 + 35x_{15} - 50$$
$$x_4^3 + x_8^3 + x_4^2x_8 + x_4x_8^2 - 10x_4^2 - 10x_8^2 - 10x_4x_8 + 35x_4 + 35x_8 - 50$$
$$x_4^3 + x_{12}^3 + x_4^2x_{12} + x_4x_{12}^2 - 10x_4^2 - 10x_{12}^2 - 10x_4x_{12} + 35x_4 + 35x_{12} - 50$$
$$x_4^3 + x_{16}^3 + x_4^2x_{16} + x_4x_{16}^2 - 10x_4^2 - 10x_{16}^2 - 10x_4x_{16} + 35x_4 + 35x_{16} - 50$$
$$x_5^3 + x_9^3 + x_5^2x_9 + x_5x_9^2 - 10x_5^2 - 10x_9^2 - 10x_5x_9 + 35x_5 + 35x_9 - 50$$
$$x_5^3 + x_{13}^3 + x_5^2x_{13} + x_5x_{13}^2 - 10x_5^2 - 10x_{13}^2 - 10x_5x_{13} + 35x_5 + 35x_{13} - 50$$
$$x_6^3 + x_{10}^3 + x_6^2x_{10} + x_6x_{10}^2 - 10x_6^2 - 10x_{10}^2 - 10x_6x_{10} + 35x_6 + 35x_{10} - 50$$

$$x_6^3 + x_{14}^3 + x_6^2 x_{14} + x_6 x_{14}^2 - 10x_6^2 - 10x_{14}^2 - 10x_6 x_{14} + 35x_6 + 35x_{14} - 50$$
$$x_7^3 + x_{11}^3 + x_7^2 x_{11} + x_7 x_{11}^2 - 10x_7^2 - 10x_{11}^2 - 10x_7 x_{11} + 35x_7 + 35x_{11} - 50$$
$$x_7^3 + x_{15}^3 + x_7^2 x_{15} + x_7 x_{15}^2 - 10x_7^2 - 10x_{15}^2 - 10x_7 x_{15} + 35x_7 + 35x_{15} - 50$$
$$x_8^3 + x_{12}^3 + x_8^2 x_{12} + x_8 x_{12}^2 - 10x_8^2 - 10x_{12}^2 - 10x_8 x_{12} + 35x_8 + 35x_{12} - 50$$
$$x_8^3 + x_{16}^3 + x_8^2 x_{16} + x_8 x_{16}^2 - 10x_8^2 - 10x_{16}^2 - 10x_8 x_{16} + 35x_8 + 35x_{16} - 50$$
$$x_9^3 + x_{13}^3 + x_9^2 x_{13} + x_9 x_{13}^2 - 10x_9^2 - 10x_{13}^2 - 10x_9 x_{13} + 35x_9 + 35x_{13} - 50$$
$$x_{10}^3 + x_{14}^3 + x_{10}^2 x_{14} + x_{10} x_{14}^2 - 10x_{10}^2 - 10x_{14}^2 - 10x_{10} x_{14} + 35x_{10} + 35x_{14} - 50$$
$$x_{11}^3 + x_{15}^3 + x_{11}^2 x_{15} + x_1 x_{15}^2 - 10x_{11}^2 - 10x_{15}^2 - 10x_1 x_{15} + 35x_{11} + 35x_{15} - 50$$
$$x_{12}^3 + x_{16}^3 + x_{12}^2 x_{16} + x_{12} x_{16}^2 - 10x_{12}^2 - 10x_{16}^2 - 10x_{12} x_{16} + 35x_{12} + 35x_{16} - 50$$
$$x_1^3 + x_6^3 + x_1^2 x_6 + x_1 x_6^2 - 10x_1^2 - 10x_6^2 - 10x_1 x_6 + 35x_1 + 35x_6 - 50$$
$$x_2^3 + x_5^3 + x_2^2 x_5 + x_2 x_5^2 - 10x_2^2 - 10x_5^2 - 10x_2 x_5 + 35x_2 + 35x_5 - 50$$
$$x_3^3 + x_8^3 + x_3^2 x_8 + x_3 x_8^2 - 10x_3^2 - 10x_8^2 - 10x_3 x_8 + 35x_3 + 35x_8 - 50$$
$$x_4^3 + x_7^3 + x_4^2 x_7 + x_4 x_7^2 - 10x_4^2 - 10x_7^2 - 10x_4 x_7 + 35x_4 + 35x_7 - 50$$
$$x_9^3 + x_{14}^3 + x_9^2 x_{14} + x_9 x_{14}^2 - 10x_9^2 - 10x_{14}^2 - 10x_9 x_{14} + 35x_9 + 35x_{14} - 50$$
$$x_{10}^3 + x_{13}^3 + x_{10}^2 x_{13} + x_{10} x_{13}^2 - 10x_{10}^2 - 10x_{13}^2 - 10x_{10} x_{13} + 35x_{10} + 35x_{13} - 50$$
$$x_{11}^3 + x_{16}^3 + x_{11}^2 x_{16} + x_1 x_{16}^2 - 10x_{11}^2 - 10x_{16}^2 - 10x_1 x_{16} + 35x_{11} + 35x_{16} - 50$$
$$x_{12}^3 + x_{15}^3 + x_{12}^2 x_{15} + x_{12} x_{15}^2 - 10x_{12}^2 - 10x_{15}^2 - 10x_{12} x_{15} + 35x_{12} + 35x_{15} - 50$$
$$x_1^3 + x_{11}^3 + x_1^2 x_{11} + x_1 x_{11}^2 - 10x_1^2 - 10x_{11}^2 - 10x_1 x_{11} + 35x_1 + 35x_{11} - 50$$
$$x_1^3 + x_{16}^3 + x_1^2 x_{16} + x_1 x_{16}^2 - 10x_1^2 - 10x_{16}^2 - 10x_1 x_{16} + 35x_1 + 35x_{16} - 50$$
$$x_4^3 + x_{10}^3 + x_4^2 x_{10} + x_4 x_{10}^2 - 10x_4^2 - 10x_{10}^2 - 10x_4 x_{10} + 35x_4 + 35x_{10} - 50$$
$$x_4^3 + x_{13}^3 + x_4^2 x_{13} + x_4 x_{13}^2 - 10x_4^2 - 10x_{13}^2 - 10x_4 x_{13} + 35x_4 + 35x_{13} - 50$$