UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Tesis de Licenciatura

Métodos algebraicos en sistemas biológicos

Mercedes Soledad Pérez Millán
mercedes.s.pm@gmail.com

Directora: Dra. Alicia Dickenstein

Marzo de 2007

Expreso mis agradecimientos ...

A mi familia, por haberme brindado la posibilidad de estudiar matemática y por haberme dado su apoyo y su amor durante toda la carrera.

A Álvaro, aquel que empezó siendo un compañero más de Análisis I y que a lo largo de estos años se fue convirtiendo en una de las personas más importantes de mi vida. Por todo su amor, por la increíble paciencia que me tiene y por aceptarme como soy.

A Alicia, por dirigir esta tesis, por todas las oportunidades que me dio, por enseñarme tantas cosas, por dedicarse con tanto empeño y por alentarme en todo momento.

A Gabriela Gil, mi profesora de matemática del IMA y principal responsable de que haya seguido esta carrera, por toda la confianza que me tuvo desde el primer momento.

A Carolina, Analía, Julián, Nico S., Nico B., Dani, David, Karina, Paula, Georgina, Laura B., Laura N., Isa, Lucio, Nicolás O., Martín S., Emilia, Paula, Marcela, Virginia, Gisele, Sergio, Roberta, Eliana, Ezequiel, Manuel, Lucas y Adriana, por acompañarme durante todos estos años en el pabellón, por las horas de estudio, por los llamados de desesperación, y por todos los momentos compartidos.

A los que me supieron brindar soporte técnico a lo largo de este trabajo. Particularmente a Enrique y a Nico B. Y a Angélica por haberme dado una gran mano con los papeles.

A Ezra Miller, Brandilyn Stigler y Abdul Jarrah por contestar tan amablemente nuestras consultas matemáticas.

A mis amigos los físicos, por las charlas compartidas dentro y fuera de la facultad, por las horas de estudio y por toda la ayuda que me dieron.

A mi amiga de la infancia, Cynthia, por siempre estar ahí, a pesar de todo.

A Flora, Marta, mis compañeros docentes del CBC, Daniel P., Dani, Santiago, Victoria, Juan Pablo, Eliana, Nico S., Constanza y Gabriela, por haberme ayudado a crecer profesionalmente.

A la gente del taller de tango, con quienes aprendí mucho más que pasos de baile. Por todos los preciosos momentos que vivimos en el salón y por todo lo que compartimos fuera.

Y a muchos más que me acompañan en la vida, me ayudan a crecer y son responsables de muchas pequeñas alegrías.

Índice general

1 Introducción **1**
1.1 Ingeniería Reversa de Sistemas Polinomiales sobre Cuerpos Finitos 2

2 Ideales Monomiales y Descomposición Primaria **4**
2.1 Ideales Monomiales 4
2.2 Descomposición Primaria de Ideales Monomiales Libres de Cuadrados . . . 4
2.2.1 Primer algoritmo 5
2.2.2 Segundo algoritmo 6

3 Interpolación **12**
3.1 Orden Monomial y Bases de Gröbner 12
3.2 Variedades en k^n 14
3.3 Algoritmo de Buchberger-Möller 15
3.3.1 Algoritmo 15
3.3.2 Algunos lemas previos 16
3.3.3 Teorema 18
3.3.4 Interpoladores 19
3.3.5 Cuando la cantidad de puntos es mucho menor que la cantidad de variables 19

4 Ejemplo de ingeniería reversa utilizando bases de Gröbner **20**
4.1 Aplicación 21

5 Ejemplo de ingeniería reversa utilizando descomposición primaria de ideales monomiales **26**
5.1 Algoritmo 28
5.2 Elección del modelo 29
5.3 Aplicación 31
5.4 Otra forma de encontrar la estructura minimal 32

6 Discretización y trabajo futuro **37**
6.1 Discretización 37
6.2 Errores en la medición 45
6.3 Trabajo futuro 45

A Programa **50**

Índice de cuadros

4.1 Funciones Booleanas usadas en el modelo 22
4.2 Funciones Booleanas dadas en forma polinomial 23

Índice de figuras

4.1 Grafo de interacciones en una célula de la red con interacciones celulares . . 24
4.2 Grafo del modelo consensual 25

6.1 Grafo de dependencia de g 37
6.2 Posibles grafos de dependencia para la primera discretización en $\mathbb{F}_4$ 39
6.3 Posibles grafos de dependencia para la segunda discretización en $\mathbb{F}_4$ 41
6.4 Más posibles grafos de dependencia para la segunda discretización en $\mathbb{F}_4$ . . 42
6.5 Posibles grafos de dependencia para la discretización en $\mathbb{F}_4$ de la segunda serie 43

Capítulo 1

Introducción

Lo que se pretende a lo largo de este trabajo es mostrar algunas herramientas matemáticas que se aplican hoy en día para resolver problemas que surgen en biología al intentar reconstruir redes reguladoras de genes. También se mostrarán ventajas y desventajas de cada una de ellas.

Uno de los objetivos más importantes en la biología de sistemas es descubrir y modelar las relaciones causales entre los componentes de dichas redes y los mecanismos que gobiernan sus dinámicas. Los métodos existentes de ingeniería reversa se pueden separar en continuos vs. discretos y en determinísticos vs. estocásticos. El objetivo de algunos métodos es describir la dinámica de la red. Y otros solo pretenden descubrir la topología de la misma, es decir, qué genes regulan cuáles otros, con un grafo dirigido (o "wiring diagram") como resultado, posiblemente indicando activación o inhibición.

La manera más común de modelar la dinámica de de las redes es mirarlas como redes bioquímicas de productos de los genes, generalmente mRNA y proteínas, y describir sus tasas de cambio a través de un sistema de ecuaciones diferenciales ordinarias. Por lo tanto, el marco en el cual se modela es el de un sistema dinámico determinístico, a tiempo continuo.

El otro extremo del espectro de modelos ve a las redes reguladoras de genes como redes lógicas. Por ejemplo, los modelos de redes Booleanas, que tienen la ventaja de ser más intuitivos que los modelos de EDO. Difieren de estas últimas al tomar al tiempo como discreto y la expresión de los genes se discretiza en solo dos estados cuantitativos, según si están presentes o ausentes.

Ejemplo 1.1. Las funciones booleanas se pueden representar como funciones polinomiales con coeficientes en $\mathbb{Z}_2$. En efecto, si x e y son variables Booleanas, entonces

$$x \wedge y = x.y,\ x \vee y = x + y + x.y,\ \neg x = x + 1$$

Observemos que la suma corresponde al O exclusivo (XOR).

Veamos, por ejemplo, el metabolismo de la lactosa en E. Coli [Eriksson]. Hay cuatro actores principales en este sistema: lactosa, glucosa, β-galactosidasa, y una proteina represora. La lactosa es un azúcar compuesto y se parte en azúcares usables por la enzima

β-galactosidasa. La proteina represora detiene la producción de β-galactosidasa excepto cuando ésta se necesita para metabolizar la lactosa.

Representamos esto con una red dinámica de cuatro nodos sobre $\mathbb{Z}_2$, donde los valores $1, 0$ corresponden a si una de las componentes está presente o ausente, respectivamente.

Si x_1 mide la glucosa, x_2 la lactosa, x_3 la β-galactosidasa, y x_4 la proteina represora. A partir de lo que los biólogos saben sobre este sistema, se pueden escribir los cuatro polinomios que dan la red dinámica

$$\begin{aligned} f_1 &= x_1 \\ f_2 &= x_2 \\ f_3 &= x_2(1 + x_4) \\ f_4 &= 1 + x_2 \end{aligned}$$

Por ejemplo, f_3 significa que la β-galactosidasa se produce si la lactosa (x_2) está presente y el represor (x_4), ausente.

Una de las mayores desventajas de los modelos Booleanos es que necesitan discretizar los datos expresados en valores reales en dos categorías, encendido/apagado, perdiendo así mucha información. En respuesta a esta deficiencia, se han desarrollado modelos discretos con más estados.

En adelante trabajaremos con los modelos a tiempo discreto, que han surgido como alternativa a los modelos a tiempo continuo cuando se pretende reconstruir algo de grandes dimensiones contando con muy poca información. Tendremos como hipótesis que los intervalos de tiempo son siempre iguales (es decir, será lo mismo medir en un determinado intervalo de tiempo que en cualquier otro) y que las observaciones no dependen de lo que pudo haber sucedido en los intervalos de tiempo anteriores.

1.1 Ingeniería Reversa de Sistemas Polinomiales sobre Cuerpos Finitos

Supongamos que tomamos como marco para modelar a los sistemas dinámicos a tiempo discreto de varios estados. Sea A el conjunto de posibles estados de los nodos de la red, y asumimos que A es un conjunto finito (aunque quizás muy grande). Por ejemplo, se puede pensar A como el conjunto de los niveles de expresión discretizados (hablaremos de la discretización de los datos en el capítulo 6). Sea

$$f : A^n \to A^n$$

esto se conoce como un *sistema dinámico discreto* sobre A de dimensión n. La iteración de f resulta en un sistema dinámico a tiempo discreto. Y f se puede describir en términos de sus funciones coordenadas $f_i : A^n \to A$, para $i = 1, \ldots, n$. Es decir, si $\mathbf{x} = (x_1, \ldots, x_n) \in A^n$ es un estado, entonces $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$. Nos referimos a un tal sistema como *sistema dinámico finito*.

Si A tiene $q = p^r$ elementos, para algún primo p, y tiene la estructura de un cuerpo finito (denotamos $k := A$), entonces cualquier función coordenada $f_i : k^n \to k$ se puede

describir con un único polinomio en $k[x_1, \ldots, x_n]$ de grado menor que q en cada variable. Pero, a menos que se conozcan todas las correspondencias, este único polinomio no se podrá determinar. Generalmente, al trabajar con estos problemas que surgen de la "vida real", solo se dispone de un conjunto (bastante acotado) de *pares de transición*. Es decir, los datos son de la forma $\mathbf{s}_1, \ldots \mathbf{s}_m, \mathbf{t}_1, \ldots, \mathbf{t}_m \in k^n$, donde $f(\mathbf{s}_i) = \mathbf{t}_i$, y usualmente $m << q$. Luego, se suelen generar varias opciones de posibles modelos al aplicar cualquier método de ingeniería reversa (es decir, al usar cualquier método que pretenda reconstruir a partir de lo observado).

En este trabajo concentraremos nuestra atención en dos trabajos donde se modelan la dinámica o las relaciones causales de las redes reguladoras de genes a través de sistemas polinomiales dinámicos finitos. Los desarrollaremos en detalle en los capítulos 4 y 5.

El primero de ellos, desarrollado en [L-S], intenta describir la dinámica de la red. Partiendo de datos como los anteriormente descriptos, busca para cada coordenada un polinomio interpolador minimal, en el sentido de que no hay un polinomio no nulo $g_i \in k[x_1, \ldots, x_n]$, que surja de reordenar y reagrupar los términos de f_i, tal que $f_i = h_i + g_i$ y $g_i(\mathbf{s}_j) = 0 \, \forall j, 1 \leq j \leq m$, $i = 1, \ldots, n$. Para esto, se elige la forma normal (resto) de algún interpolador con respecto a una base de Gröbner (ver definición 3.8) para el ideal de $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ (ver definición 3.17). Uno de los problemas de esta elección es que depende fuertemente de la elección de un orden monomial (ver definición 3.1) particular para calcular la base de Gröbner, y no hay una elección de orden monomial que sea canónica.

Tratando de evitar este problema, el segundo trabajo se basa en la descomposición primaria de un ideal monomial (ver capítulo 2) que se genera fácilmente a partir de los datos. Pero aquí solo se pretende describir las relaciones causales entre los nodos de la red. Por eso, el objetivo es encontrar, para cada coordenada, conjuntos minimales (según la inclusión) de variables para los cuales existe un interpolador. Comentamos también, en el capítulo 5, un algoritmo desarrollado en [Krupa] que intenta resolver el problema de hallar una estructura minimal con una cantidad polinomial de operaciones, aunque con menos precisión.

En los capítulos 2 y 3 describiremos las herramientas matemáticas necesarias para comprender los dos ejemplos arriba mencionados.

Y finalmente, en el capítulo 6 se considera el problema de la discretización de los datos y se plantea una posible teoría a desarrollar para intentar minimizar la pérdida de información y a su vez mantener la consistencia (ver sección 6.3).

Capítulo 2

Ideales Monomiales y Descomposición Primaria

2.1 Ideales Monomiales

Sea k un cuerpo, y $k[x_1, \ldots, x_n]$ el anillo polinomial en n indeterminadas.

Definición 2.1. Un ***monomio*** en $k[x_1, \ldots, x_n]$ es un producto $x_1^{m_1} x_2^{m_2} \ldots x_n^{m_n}$, con $m_i \in \mathbb{N}_0, i = 1, \ldots, n$.

Un ideal $M \subseteq k[x_1, \ldots, x_n]$ se dice ***ideal monomial*** si está generado por monomios.

Observación 2.2. Son equivalentes

1. M es un ideal monomial.
2. Si $f = \sum_{(\alpha_1,\ldots,\alpha_n)\in\mathbb{N}_0^n} c_{\alpha_1,\ldots,\alpha_n} x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ pertenece a M entonces $x_1^{\alpha_1} \ldots x_n^{\alpha_n} \in M$ si $c_{\alpha_1,\ldots,\alpha_n} \neq 0$.

Luego, un ideal monomial está unívocamente determinado por los monomios que contiene.

Con esta equivalencia se puede ver fácilmente que la intersección de ideales monomiales es un ideal monomial. De hecho, si M_1 y M_2 son dos ideales monomiales y $f = \sum_{(\alpha_1,\ldots,\alpha_n)\in\mathbb{N}_0^n} c_{\alpha_1,\ldots,\alpha_n} x_1^{\alpha_1} \ldots x_n^{\alpha_n} \in M_1 \cap M_2$. Sea $x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ tal que $c_{\alpha_1,\ldots,\alpha_n} \neq 0$. Como $f \in M_1, x_1^{\alpha_1} \ldots x_n^{\alpha_n} \in M_1$, y como $f \in M_2, x_1^{\alpha_1} \ldots x_n^{\alpha_n} \in M_2$. Luego, $x_1^{\alpha_1} \ldots x_n^{\alpha_n} \in M_1 \cap M_2$.

Definición 2.3. Un monomio $x_1^{m_1} x_2^{m_2} \ldots x_n^{m_n}$ es ***libre de cuadrados*** si $0 \leq m_i \leq 1$, $\forall i$.

Un ***ideal monomial libre de cuadrados*** es un ideal generado por monomios libres de cuadrados.

2.2 Descomposición Primaria de Ideales Monomiales Libres de Cuadrados

Definición 2.4. Un ideal p se dice ***primo*** si $r_1 r_2 \in p$ implica que $r_1 \in p$ ó $r_2 \in p$

Definición 2.5. Una ***descomposición primaria*** de un ideal I es una expresión de I como una intersección finita de ideales primarios; un ideal q se dice ***primario*** si $r_1 r_2 \in q$ implica que $r_1 \in q$ ó $r_2 \in \sqrt{q}$ (donde $\sqrt{q}$ denota el radical del ideal q).

Observación 2.6.

- Si un ideal p es primo, entonces p es radical. En efecto, $p \subseteq \sqrt{p}$ vale siempre. Y sea r tal que $r^m \in p$, con $m \in \mathbb{N}$ el mínimo para el cual se verifica la pertenencia. Supongamos que $m > 1$. Entonces $r^m = r.r^{m-1}$ y como p es primo, $r \in p$ ó $r^{m-1} \in p$. Pero esto es un absurdo por ser $m > 1$ un mínimo. Luego, $m = 1$ y por lo tanto $\sqrt{p} \subseteq p$.
- Si el ideal p es primo, y $r_1 r_2 \in p \Rightarrow r_1 \in p$ *ó* $r_2 \in p \Rightarrow r_1 \in p$ *ó* $r_2 \in \sqrt{p} = p$. Luego, primo implica primario.

Como solo necesitaremos trabajar con ideales monomiales libres de cuadrados, su descomposición primaria consistirá únicamente de ideales primos.

En lo que sigue, describiremos dos algoritmos para calcular la descomposición primaria de un ideal monomial libre de cuadrados. Para más referencia sobre este tema, consultar [H-S].

2.2.1 Primer algoritmo

Lema 2.7. *Sea M un ideal monomial en $k[x_1, \ldots, x_n]$. Si $r \in M$ tal que $r = r_1 r_2$ donde r_1 y r_2 son coprimos, entonces $M = (M + \langle r_1 \rangle) \cap (M + \langle r_2 \rangle)$.*

Demostración. Como M es un ideal monomial, alcanza con ver que M y $(M + \langle r_1 \rangle) \cap (M + \langle r_2 \rangle)$ contienen los mismos monomios. Un monomio r' pertenece a $(M + \langle r_j \rangle)$ si y sólo si $r' \in M$ ó $r_j \mid r'$. Como r_1 y r_2 son coprimos, se tiene

$$r' \in (M + \langle r_1 \rangle) \cap (M + \langle r_2 \rangle) \Leftrightarrow r' \in M \text{ ó } r_1 r_2 \mid r' \Leftrightarrow r' \in M$$

□

Lema 2.8. *Sea $q = \langle x_{i_1}, \ldots, x_{i_r} \rangle \subset k[x_1, \ldots, x_n]$, entonces q es primo.*

Demostración. Sea $r_1.r_2 \in q$, veamos que $r_1 \in q$ ó $r_2 \in q$. Sean

$$r_1 = r_1^{(1)} + r_1^{(2)}\,,\ r_2 = r_2^{(1)} + r_2^{(2)}\,,$$

con $r_i^{(1)} \in k[x_j : j \neq i_l, l = 1, \ldots, r]$ y $r_i^{(2)} \in q, i = 1, 2$. Luego,

$$r_1.r_2 = r_1^{(1)}.r_2^{(1)} + (r_1^{(1)}.r_2^{(2)} + r_1^{(2)}.r_2^{(1)} + r_1^{(2)}.r_2^{(2)})$$

y vale que $(r_1^{(1)}.r_2^{(2)} + r_1^{(2)}.r_2^{(1)} + r_1^{(2)}.r_2^{(2)}) \in q$. Por lo tanto, $r_1^{(1)}.r_2^{(1)} \in q \cap k[x_j : j \neq i_l, l = 1, \ldots, r] = \{0\}$, y como $k[x_j : j \neq i_l, l = 1, \ldots, r]$ es dominio íntegro, $r_1^{(1)} = 0$ ó $r_2^{(1)} = 0$. Y esto implica $r_1 \in q$ ó $r_2 \in q$, como se quería probar.

□

A partir de estos lemas se puede generar un algoritmo sencillo para conseguir una descomposición primaria de un ideal monomial libre de cuadrados.

Ejemplo 2.9. Sea $M = \langle x_1x_3, x_2, x_1x_4\rangle \subseteq k[x_1, x_2, x_3, x_4]$ un ideal monomial libre de cuadrados.

Por el lema 2.7

$M = \langle x_1, x_2, x_1x_4\rangle \cap \langle x_3, x_2, x_1x_4\rangle = \langle x_1, x_2, x_1\rangle \cap \langle x_1, x_2, x_4\rangle \cap \langle x_3, x_2, x_1\rangle \cap \langle x_3, x_2, x_4\rangle$

Y como, por el lema 2.8, cada intersecando es primario, esta es una descomposición primaria para M.

Observación 2.10. Si M es un ideal monomial libre de cuadrados, sus componentes primarias son ideales primos de la forma $\langle x_{i_1}, \ldots, x_{i_r}\rangle$.

Observación 2.11. Como podemos ver en el ejemplo, este procedimiento no siempre nos da una descomposición irredundante ($\langle x_1, x_2, x_1\rangle = \langle x_1, x_2\rangle \subseteq \langle x_1, x_2, x_4\rangle$).

2.2.2 Segundo algoritmo

En esta sección mostraremos un algoritmo de descomposición primaria basado en el dual de Alexander del ideal monomial libre de cuadrados. Primero desarrollaremos un poco de teoría sobre complejos simpliciales y adaptaremos la notación utilizada en [Miller]:

Sea $\Delta = \mathcal{P}(\{1, \ldots, n\})$, el conjunto de partes de $\{1, \ldots, n\}$.

Definición 2.12. $\Gamma \subseteq \Delta$ es un *complejo simplicial* si es cerrado por inclusiones. Es decir, si $X \in \Gamma$ e $Y \subseteq X$, entonces $Y \in \Gamma$.

Definición 2.13. Γ complejo simplicial, definimos

$$\widetilde{\Gamma} := \{F \in \Delta : F^C \notin \Gamma\}$$

donde $F^C = \{1, \ldots, n\} \setminus F$.

Ejemplo 2.14. Para $n = 4$

$\Gamma = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{2, 3\}, \{3, 4\}\, \{2, 4\}, \{2, 3, 4\}\}$

$F^C \notin \Gamma \Leftrightarrow F^C \in \{\{1, 4\}, \{1, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{1, 2, 3\}, \{1, 2, 3, 4\}\}$

Luego, $\widetilde{\Gamma} = \{\{2, 3\}, \{2, 4\}, \{3\}, \{2\}, \{4\}, \emptyset\}$.

Lema 2.15. *$\widetilde{\Gamma}$ es un complejo simplicial.*

Demostración. Sea $X \in \widetilde{\Gamma}$ e $Y \subseteq X$, veamos que $Y \in \widetilde{\Gamma}$.

$Y \subseteq X$, luego $Y^C \supseteq X^C$. Si $Y^C \in \Gamma$, como Γ es complejo simplicial, entonces $X^C \in \Gamma$, pero esto no es cierto pues $X \in \widetilde{\Gamma}$. Por lo tanto, $Y^C \notin \Gamma$ e $Y \in \widetilde{\Gamma}$. □

Lema 2.16. $\widetilde{\left(\widetilde{\Gamma}\right)} = \Gamma$

Demostración. $\widetilde{\left(\widetilde{\Gamma}\right)} = \{F \in \Delta : F^C \notin \widetilde{\Gamma}\} = \{F \in \Delta : F \in \Gamma\} = \Gamma$ □

Definición 2.17. Sea $b \in \mathbb{N}_0^n$, definiremos

$\mathbf{x}^b = x_1^{b_1} \dots x_n^{b_n}$

$\mathfrak{m}^b = \langle x_i^{b_i} : b_i \neq 0 \rangle = \langle x_i^{b_i} : b_i \neq 0 \rangle$

Ejemplo 2.18. Supongamos que $n = 4$ y $b = (1, 1, 0, 0)$. Luego,

$\mathbf{x}^b = x_1^1 x_2^1 x_3^0 x_4^0 = x_1 x_2 \quad$ y $\quad \mathfrak{m}^b = \langle x_1, x_2 \rangle$

Observación 2.19. Sea $F \in \Delta$, le asociaremos $f \in \{0,1\}^n$, $f_i = \begin{cases} 0 & si \quad i \notin F \\ 1 & si \quad i \in F \end{cases}$

Y notaremos también $\mathbf{x}^F := \mathbf{x}^f$.

Por ejemplo, si $F \subseteq \{1, \dots, 4\}, F = \{1, 4\}$, $\mathbf{x}^F = x_1 x_4$.

Definición 2.20. Dado $\alpha \in \mathbb{N}_0^n$, definimos el *soporte de* α de la siguiente manera:

$$sop(\alpha) = \{i : \alpha_i \neq 0\}$$

Por ejemplo, $sop((1, 0, 3, 0, 1)) = \{1, 3, 5\}$.

Definición 2.21. Sea Γ complejo simplicial, definimos un ideal monomial

$$I_\Gamma = \langle \mathbf{x}^F : F \notin \Gamma \rangle = \langle \prod_{i \in F} x_i : F \notin \Gamma \rangle$$

En el ejemplo 2.14, $I_\Gamma = \langle x_1x_4,\ x_1x_3,\ x_1x_2x_4,\ x_1x_3x_4,\ x_1x_2x_3,\ x_1x_2x_3x_4 \rangle = \langle x_1x_4,\ x_1x_3 \rangle = \langle x_3, x_4 \rangle \cap \langle x_1 \rangle$

Definición 2.22. Sea M un ideal monomial libre de cuadrados, $\widetilde{M}$ es el siguiente ideal monomial libre de cuadrados:

$M = \langle \mathbf{x}^F : F \in A \rangle = \langle \prod_{i \in F} x_i : F \in A \rangle$

$\widetilde{M} := \bigcap_{F \in A} \mathfrak{m}^F = \bigcap_{F \in A} \langle x_i, i \in F \rangle$

Donde $A \subseteq \Delta$.

En el ejemplo 2.14, $M = I_\Gamma = \langle x_1x_4, x_1x_3 \rangle$. Entonces

$\widetilde{M} = \langle x_1, x_4 \rangle \cap \langle x_1, x_3 \rangle = \langle x_1, x_3x_4 \rangle$

Lema 2.23. $I_\Gamma = \bigcap_{F^C \in \Gamma} \mathfrak{m}^F$

Demostración. Probemos las dos inclusiones:

$\subseteq$) Sea $\mathbf{x}^G$ con $G \notin \Gamma$ y sea F tal que $F^C \in \Gamma$. Queremos ver que $\mathbf{x}^G \in \langle x_i : i \in F \rangle$. O sea, queremos ver que existe $i \in F$ tal que $i \in G$. Es decir, dados $G \notin \Gamma, F^C \in \Gamma$, queremos probar que existe $i \in F$ tal que $i \in G$. Si no, $G \subseteq F^C$ y como Γ es complejo simplicial, $G \in \Gamma$ y esto es absurdo.

$\supseteq$) Sea $\mathbf{x}^\alpha \in \bigcap_{F^C \in \Gamma} \mathfrak{m}^F$, queremos ver que existe $G \notin \Gamma$ tal que $\mathbf{x}^G | \mathbf{x}^\alpha$ ($\Leftrightarrow G \subseteq sop(\alpha)$). Pero $\mathbf{x}^\alpha \in \bigcap_{F^C \in \Gamma} \mathfrak{m}^F$ si y solo si $sop(\alpha) \cap F \neq \emptyset \ \forall F$ tal que $F^C \in \Gamma$. Si tomamos $G = sop(\alpha)$,

entonces $sop(\alpha) \subseteq sop(\alpha)$ y también vale que $sop(\alpha) \notin \Gamma$. De hecho, si $sop(\alpha) \in \Gamma$, esto equivale a decir que $(sop(\alpha)^C)^C \in \Gamma$, y si llamamos $F = sop(\alpha)^C$ se tiene que cumplir que $sop(\alpha) \cap sop(\alpha)^C \neq \emptyset$, y esto es absurdo.

□

Definición 2.24. Sea M ideal monomial, si $M = \bigcap_{F \in S} \mathfrak{m}^F$, donde $S \subseteq \Delta$, definimos

$$\Gamma_M = \{F \in \Delta : F^C \in S\}$$

Proposición 2.25. Γ *complejo simplicial,*

$$\widetilde{(I_\Gamma)} = I_{\widetilde{\Gamma}}$$

Demostración. Recordemos:

$I_\Gamma = \langle \mathbf{x}^F : F \notin \Gamma \rangle$

$\widetilde{(I_\Gamma)} = \bigcap_{F \notin \Gamma} \mathfrak{m}^F$

$\widetilde{\Gamma} = \{F : F^C \notin \Gamma\}$

$I_{\widetilde{\Gamma}} = \langle \mathbf{x}^F : F \notin \widetilde{\Gamma} \rangle = \langle \mathbf{x}^F : F^C \in \Gamma \rangle$

O sea, queremos ver que

$$\bigcap_{F \notin \Gamma} \mathfrak{m}^F = \langle \mathbf{x}^F : F^C \in \Gamma \rangle$$

Veamos la doble inclusión:

$\subseteq$) Si $\mathbf{x}^\alpha \in \bigcap_{G \notin \Gamma} \mathfrak{m}^G$, queremos ver que existe F tal que $F^C \in \Gamma$ y $\mathbf{x}^F | \mathbf{x}^\alpha$. Si tomamos $F = sop(\alpha)$, entonces $sop(\alpha) \subseteq sop(\alpha)$ y también vale que $sop(\alpha)^C \in \Gamma$. De hecho, si $sop(\alpha)^C \notin \Gamma$, entonces $\mathbf{x}^\alpha \in \mathfrak{m}^{sop(\alpha)^C}$, y por lo tanto $sop(\alpha) \cap sop(\alpha)^C \neq \emptyset$, y esto es absurdo.

$\supseteq$) Sea $F \notin \Gamma$ queremos ver que si $G^C \in \Gamma, \mathbf{x}^G \in \mathfrak{m}^F$. Y esto pasa si y solo si $F \cap G \neq \emptyset$. Supongamos que no, entonces $F \subseteq G^C$, pero Γ es complejo simplicial y entonces $F \in \Gamma$. Contradicción. □

Corolario 2.26. $\widetilde{\left(\widetilde{M}\right)} = M$

Demostración. Primero observemos que $M = I_{\Gamma_M}$.Y usando la proposición 2.25 y el lema 2.16, vale que

$$\widetilde{\left(\widetilde{M}\right)} = \widetilde{\left(\widetilde{I_{\Gamma_M}}\right)} = \widetilde{\left(I_{\widetilde{\Gamma_M}}\right)} = I_{\widetilde{(\widetilde{\Gamma_M})}} = I_{\Gamma_M} = M$$ □

Sea ahora M ideal monomial libre de cuadrados.

$\mathbf{1} = (1, \ldots, 1) \in \{0,1\}^n$.

$a_M \in \{0,1\}^n$ será el exponente del mínimo común múltiplo de los generadores minimales de M.

Sea $Irr(M)$ el conjunto de las componentes primarias irredundantes de M (es decir, aquellas componentes minimales, para la inclusión, en el conjunto de componentes primarias de M).

Definición 2.27. Se define el ***dual de Alexander*** de M como

$$M^\vee = \langle \mathbf{x}^b : \mathfrak{m}^b \in Irr(M) \rangle$$

Luego, si encontramos los generadores minimales de $M^\vee$, tendremos las componentes primarias irredundantes de M.

Para hallar los generadores de $M^\vee$ (sin conocer las componentes primarias irredundantes de M, por supuesto), introducimos la siguiente definición (ver [A-M]):

Definición 2.28. Sean I y J ideales de un anillo A, su *ideal cociente* (o *transportador*) es:

$$(I : J) = \{f \in A : fJ \subseteq I\}$$

y este conjunto es un ideal.

Proposición 2.29. $\left(\mathfrak{m}^{a_M+\mathbf{1}} : M^\vee\right) = M + \mathfrak{m}^{a_M+\mathbf{1}}$

Observación 2.30. Antes de demostrar la Proposición 2.29, veamos algunos ejemplos y observaciones:

- Supongamos que $M = \langle x_1x_2, x_5 \rangle \subseteq k[x_1, \ldots, x_5]$. Luego, el mínimo común múltiplo de los generadores minimales de M es $x_1x_2x_5$, y por lo tanto $a_M = (1, 1, 0, 0, 1)$. De esta forma,

  $$\mathfrak{m}^{a_M+\mathbf{1}} = \langle x_1^2, x_2^2, x_3, x_4, x_5^2 \rangle$$

  Es decir, aquellas variables involucradas entre los generadores de M, aparecen al cuadrado. Y las que no estaban involucradas, en este ideal aparecen (a la primera).

- Teniendo presente el primer algoritmo de descomposición primaria (visto en la sección 2.2.1), podemos notar que si $\mathfrak{m}^b \in Irr(M)$ y $b_i = 1$, entonces x_i aparece entre los generadores de M y, por lo tanto, $(a_M)_i = 1$.

  Luego, $(a_M + \mathbf{1} - b)_i = \begin{cases} 1 & si\ b_i = 1 \\ (a_M)_i + 1 & si\ b_i = 0 \end{cases}$

  Entonces, $\mathfrak{m}^{a_M+\mathbf{1}-b} = \mathfrak{m}^b + \mathfrak{m}^{a_M+\mathbf{1}}$

  En el ejemplo anterior, si tomamos $\mathfrak{m}^b = \langle x_1, x_5 \rangle, b = (1, 0, 0, 0, 1)$. Tenemos:

  $a_M + \mathbf{1} - b = (1, 2, 1, 1, 1)$

  $\mathfrak{m}^{a_M+\mathbf{1}-b} = \langle x_1, x_2^2, x_3, x_4, x_5 \rangle = \langle x_1, x_5, x_1^2, x_2^2, x_3, x_4, x_5^2 \rangle = \mathfrak{m}^b + \mathfrak{m}^{a_M+\mathbf{1}}$

- Vale que $(I : \sum_i J_i) = \bigcap_i (I : J_i)$

- Si I es monomial, $(I : \mathbf{x}^b)$ es monomial. De hecho, $f\mathbf{x}^b \in I$ si y solo si cada monomio de $f\mathbf{x}^b$ pertenece a I. Pero cada monomio de $f\mathbf{x}^b$ es un monomio de f por $\mathbf{x}^b$. Entonces, $f\mathbf{x}^b \in I$ si y solo si cada monomio de $f \in (I : \mathbf{x}^b)$. Luego, $(I : \mathbf{x}^b)$ es monomial.

Demostración. (de la Proposición 2.29)

$\left(\mathfrak{m}^{a_M+\mathbf{1}} : M^\vee\right) = \bigcap_{\substack{b/\mathbf{x}^b generador \\ minimal\, de\, M^\vee}} \left(\mathfrak{m}^{a_M+\mathbf{1}} : \mathbf{x}^b\right)$

Como $\left(\mathfrak{m}^{a_M+\mathbf{1}} : \mathbf{x}^b\right)$ es monomial, veamos qué sucede al tomar un monomio de la forma $\mathbf{x}^c$:

$\mathbf{x}^c.\mathbf{x}^b \in \mathfrak{m}^{a_M+\mathbf{1}} \Leftrightarrow \mathbf{x}^c \in \mathfrak{m}^{a_M+\mathbf{1}-b}$.

Luego, tomando todas las intersecciones sobre b tal que $\mathbf{x}^b$ es generador minimal de $M^\vee$,

$\bigcap \left(\mathfrak{m}^{a_M+\mathbf{1}} : \mathbf{x}^b\right) = \bigcap \mathfrak{m}^{a_M+\mathbf{1}-b} = \bigcap \left(\mathfrak{m}^b + \mathfrak{m}^{a_M+\mathbf{1}}\right) = \bigcap \mathfrak{m}^b + \mathfrak{m}^{a_M+\mathbf{1}} = M + \mathfrak{m}^{a_M+\mathbf{1}}$

□

Lema 2.31. $(M^\vee)^\vee = M$

Demostración. Observemos que la definición de $M^\vee$ es la "inversa" de la definición 2.22:

Si $I = \langle \mathbf{x}^F : F \in A\rangle \Rightarrow \widetilde{I} = \bigcap_{F\in A} \mathfrak{m}^F$

Si $M = \bigcap_{\mathfrak{m}^F \in Irr(M)} \mathfrak{m}^F \Rightarrow M^\vee = \langle \mathbf{x}^F : \mathfrak{m}^F \in Irr(M)\rangle$

Entonces, $\widetilde{M^\vee} = M$ y por el corolario 2.26, $M^\vee = \widetilde{M}$. Por lo tanto, usando este corolario nuevamente,

$(M^\vee)^\vee = \widetilde{\left(\widetilde{M}\right)} = M$.

□

Teorema 2.32. *Los generadores de $M^\vee$ son los generadores del ideal $\left(\mathfrak{m}^{a_M+\mathbf{1}} : M\right)$ que no son divisibles por $x_i^{(a_M)_i+1}$ para $1 \le i \le n$.*

Demostración. Primero observemos que $a_M = a_{M^\vee}$ (esto es claro porque el mínimo común múltiplo de los generadores minimales de M es el producto de todas las variables que están involucradas en dichos generadores, y estas son las mismas que luego forman los generadores minimales de $M^\vee$ - recordar el primer algoritmo).

A partir del Lema 2.31 y de la Proposición 2.29, se tiene que

$\left(\mathfrak{m}^{a_M+\mathbf{1}} : M\right) = \left(\mathfrak{m}^{a_M+\mathbf{1}} : (M^\vee)^\vee\right) = M^\vee + \mathfrak{m}^{a_M+\mathbf{1}}$ □

Observación 2.33. Equivalentemente, los generadores de $M^\vee$ son aquellos generadores de $\left(\mathfrak{m}^{a_M+\mathbf{1}} : M\right)$ que solo involucran variables que aparecen entre los generadores de M, y que son libres de cuadrados.

Ejemplo 2.34. Sea $M = \langle x_1x_3, x_2, x_1x_4\rangle \subseteq k[x_1, x_2, x_3, x_4]$ el ideal monomial libre de cuadrados del ejemplo 2.9.

$a_M = \mathbf{1} \Rightarrow \mathfrak{m}^{a_M+\mathbf{1}} = \langle x_1^2, x_2^2, x_3^2, x_4^2\rangle$

$\Rightarrow \left(\mathfrak{m}^{a_M+\mathbf{1}} : M\right) = \left(\langle x_1^2, x_2^2, x_3^2, x_4^2\rangle : \langle x_1x_3, x_2, x_1x_4\rangle\right) = \langle x_1x_2, x_3x_2x_4, x_1^2, x_2^2, x_3^2, x_4^2\rangle$

Como $x_i^2 \mid x_i^2$, pero $x_i^2 \nmid x_1x_2$ y $x_i^2 \nmid x_3x_2x_4$, $\forall i$, $M^\vee = \langle x_1x_2, x_3x_2x_4\rangle$.

Y, por lo tanto, $M = \langle x_1, x_2\rangle \cap \langle x_3, x_2, x_4\rangle$ es su descomposición primaria.

Observación 2.35. Un ventaja de este algoritmo es que devuelve una descomposición irredundante.

Pero una desventaja es que implica calcular los generadores del ideal cociente, y aunque se lo puede considerar como la intersección de ideales cocientes más sencillos aún sigue siendo difícil de llevar a cabo.

Para agilizar las cuentas al calcular la descomposición primaria irredundante de ideales monomiales con miles de generadores, en [Milowski] se presentan dos algoritmos basados en lo que el autor denomina *"árboles monomiales"*. El primero de los métodos usa el dual de Alexander del ideal monomial. El segundo utiliza el complejo de Scarf del ideal monomial.

De todos modos, el algoritmo para descomposición primaria de ideales monomiales está implementado en [CoCoA], [GPS05] y [Macaulay 2].

Capítulo 3

Interpolación

3.1 Orden Monomial y Bases de Gröbner

Notemos con $\mathbf{x}^\alpha := x_1^{\alpha_1} \dots x_n^{\alpha_n}$, $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}_0^n$, $|\alpha| := \alpha_1 + \cdots + \alpha_n$

Definición 3.1. Un ***orden monomial*** σ en el conjunto de los monomios de $k[x_1, \dots, x_n]$ es

1. Un orden estricto total entre los monomios (denotado por $\prec_\sigma$):
   - $\mathbf{x}^\alpha \not\prec_\sigma \mathbf{x}^\alpha$ (arreflexivo)
   - $\mathbf{x}^\alpha \prec_\sigma \mathbf{x}^\beta$ y $\mathbf{x}^\alpha \prec_\sigma \mathbf{x}^\beta$ no pueden ocurrir simultáneamente (antisimétrico)
   - $\mathbf{x}^\alpha \prec_\sigma \mathbf{x}^\beta$ y $\mathbf{x}^\beta \prec_\sigma \mathbf{x}^\gamma \Rightarrow \mathbf{x}^\alpha \prec_\sigma \mathbf{x}^\gamma$ (transitivo)
   - total: $\alpha, \beta \in \mathbb{N}_0$, o bien $\mathbf{x}^\alpha \prec_\sigma \mathbf{x}^\beta$, o bien $\mathbf{x}^\alpha = \mathbf{x}^\beta$ (donde $\mathbf{x}^\alpha = \mathbf{x}^\beta \Leftrightarrow \alpha = \beta$), o bien $\mathbf{x}^\beta \prec_\sigma \mathbf{x}^\alpha$.
2. Compatible con el producto. Es decir, si $\mathbf{x}^\alpha \prec_\sigma \mathbf{x}^\beta$, entonces $\mathbf{x}^\alpha \mathbf{x}^\gamma \prec_\sigma \mathbf{x}^\beta \mathbf{x}^\gamma$ para todo $\mathbf{x}^\gamma$ monomio.
3. Un buen orden: Todo conjunto no vacío de monomios tiene un menor elemento para σ.

Esta última propiedad se puede reemplazar por
3'. $1 \preceq_\sigma \mathbf{x}^\gamma$, $\forall\, \gamma \in \mathbb{N}_0$

Ejemplos 3.2.

- Órdenes lexicográficos puros, con $x_1 \succ_{lex} x_2 \succ_{lex} \cdots \succ_{lex} x_n$

  $\mathbf{x}^\alpha \prec_{lex} \mathbf{x}^\beta \underset{def}{\Leftrightarrow} \alpha_1 < \beta_1$ ó $\alpha_1 = \beta_1$ y $\alpha_2 < \beta_2$ ó ...

  Por ejemplo, $x_1^2 x_2^2 x_3^7 \prec_{lex} x_1^2 x_2^3 x_3$

- Órdenes lexicográficos graduados u órdenes diagonales, con $x_1 \succ_{deglex} x_2 \succ_{deglex} \cdots \succ_{deglex} x_n$

  $\mathbf{x}^\alpha \prec_{deglex} \mathbf{x}^\beta \underset{def}{\Leftrightarrow} |\alpha| < |\beta|$ ó $|\alpha| = |\beta|$ y se aplica orden lexicográfico puro con $x_1 \succ_{lex} x_2 \succ_{lex} \cdots \succ_{lex} x_n$.

Por ejemplo, $x_1^2x_2^2x_3^7 \succ_{deglex} x_1^2x_2^3x_3$
$x_1x_3^2 \succ_{deglex} x_2^2x_3$

- Orden "grevlex" (graded reverse lexicographic)
  $\mathbf{x}^\alpha \prec_{grevlex} \mathbf{x}^\beta \underset{def}{\Leftrightarrow} |\alpha| < |\beta|$ ó $|\alpha| = |\beta|$ y $\alpha - \beta = (\alpha_1 - \beta_1, \ldots, \alpha_n - \beta_n) \in \mathbb{Z}^n$ y el primer elemento a la derecha no nulo es positivo.
  Por ejemplo, $x_1x_3^2 \prec_{grevlex} x_2^2x_3$

Antes de llegar al concepto de Base de Gröbner, conviene tener presentes algunos resultados (ver [C-L-O]):

Teorema 3.3. *(Lema de Dickson)*
Si M es un ideal monomial, entonces M está generado por finitos de sus monomios generadores.

Fijado un orden monomial σ, notamos con $Lt_\sigma(f)$ al término de cabeza de f (i.e. aquel término cuyo monomio es mayor que todos los monomios de los demás términos de f). Y definimos

$$Lt_\sigma(I) := \langle Lt_\sigma(f), f \in I \rangle,$$

el ideal inicial de I con respecto a σ.

A partir de esta definición, y haciendo uso del Lema de Dickson, se prueba fácilmente el siguiente

Teorema 3.4. *(de la base de Hilbert o noetherianidad de $k[x_1, \ldots, x_n]$)*
Sea $I \subseteq k[x_1, \ldots, x_n]$ ideal. Entonces existen $f_1, \ldots, f_n \in I$ tales que $I = \langle f_1, \ldots, f_n \rangle$.

Definición 3.5. Sean $\{f_1, \ldots, f_s\} \subseteq k[x_1, \ldots, x_n]$. Se tiene un ***algoritmo de división*** con respecto a un orden σ fijo cuando se tiene un algoritmo tal que dado $f \in k[x_1, \ldots, x_n]$ produce $q_1, \ldots, q_s, r$ tales que

$$f = q_1f_1 + \cdots + q_sf_s + r$$

con

1. Si $q_i \neq 0, Lt_\sigma(q_if_i) \preceq_\sigma Lt_\sigma(f)$
2. Ningún monomio de r pertenece a $\langle Lt_\sigma(f_1), \ldots, Lt_\sigma(f_s) \rangle$

Observación 3.6. Si dividimos f por $\{f_1, \ldots, f_s\}$ y obtengo
$f = q_1f_1 + \cdots + q_sf_s + r$
Entonces, $f \in \langle f_1, \ldots, f_s \rangle \Leftrightarrow r \in \langle f_1, \ldots, f_s \rangle$

Observación 3.7. Dados $\{f_1, \ldots, f_s\} = F$ polinomios, un orden σ y un algoritmo de división. Notemos con $r_F(f)$ al resto de dividir a f por $\{f_1, \ldots, f_s\}$. Sea $I = \langle f_1, \ldots, f_s \rangle$. Son equivalentes

- $f \in I \Leftrightarrow r_F(f) = 0$

- $Lt_\sigma(I) = \langle Lt_\sigma(f_1), \ldots, Lt_\sigma(f_s)\rangle$

Definición 3.8. Sea I ideal de $k[x_1, \ldots, x_n]$. Fijado un orden monomial σ, se dice que $\{g_1, \ldots, g_t\}$ es una ***base de Gröbner*** de I si

- $g_1, \ldots, g_t \in I$
- $Lt_\sigma(I) = \langle Lt_\sigma(g_1), \ldots, Lt_\sigma(g_t)\rangle$

o, equivalentemente, si dado un algoritmo de división para el orden σ,

- $g_1, \ldots, g_t \in I$
- $f \in I \Leftrightarrow r_{\{g_1,\ldots,g_t\}}(f) = 0$

Observación 3.9. $G = \{g_1, \ldots, g_t\}$ es un sistema de generadores (finito) del ideal I

Observación 3.10. Existe una tal base de Gröbner por el Lema de Dickson: $Lt_\sigma(I)$ está generado por finitos de sus generadores.

Definición 3.11. Sea σ orden fijo, I ideal de $k[x_1, \ldots, x_n]$. Se dice que $G = \{g_1, \ldots, g_t\}$ es ***base de Gröbner reducida*** de I para σ si

1. g_i es mónico para todo $1 \leq i \leq t$
2. Dados i, j si $Lt_\sigma(g_i) \mid Lt_\sigma(g_j)$, entonces $i = j$
3. Dado i, ningún monomio de g_i pertenece a $\langle Lt_\sigma(g_j), j \neq i\rangle$

Teorema 3.12. *Dado σ orden monomial e $I \subseteq k[x_1, \ldots, x_n]$ ideal. Existe una Base de Gröbner reducida para I y es única.*

3.2 Variedades en k^n

Definición 3.13. Se dice que $V \subseteq k^n$ es una ***variedad*** si existe un conjunto de polinomios $\mathcal{F} \subseteq k[x_1, \ldots, x_n]$ tal que $V = \{\mathbf{x} \in k^n : f(\mathbf{x}) = 0 \; \forall f \in \mathcal{F}\}$

Observación 3.14. $f(\mathbf{x}) = 0 \; \forall f \in \mathcal{F} \Leftrightarrow f(\mathbf{x}) = 0 \; \forall f \in \langle \mathcal{F}\rangle = I$ ideal.
Luego, se nota
$V = \mathbb{V}(I)$, con I ideal.

Proposición 3.15. $\mathbb{V}(I) \cup \mathbb{V}(J) = \mathbb{V}(I.J)$

Demostración. Probemos las dos inclusiones.
$\subseteq) I.J \subseteq I$, luego si $\mathbf{x}$ es tal que $f(\mathbf{x}) = 0 \; \forall f \in I$, en particular $f(\mathbf{x}) = 0 \; \forall f \in I.J \Rightarrow \mathbb{V}(I) \subseteq \mathbb{V}(I.J)$. Análogamente $\mathbb{V}(J) \subseteq \mathbb{V}(I.J)$. Y por lo tanto, $\mathbb{V}(I) \cup \mathbb{V}(J) \subseteq \mathbb{V}(I.J)$
$\supseteq)$ Sea $\mathbf{x}$ tal que $f(\mathbf{x}) = 0 \; \forall f \in I.J$. Supongamos que $\mathbf{x} \notin \mathbb{V}(I)$, veamos que $\mathbf{x} \in \mathbb{V}(J)$. En efecto, sea $g \in I$ tal que $g(\mathbf{x}) \neq 0$. Sea $h \in J$. Como $g.h \in I.J$, entonces $0 = g.h(\mathbf{x}) = g(\mathbf{x})h(\mathbf{x})$. Luego, $h(\mathbf{x}) = 0$. Y esto vale para cualquier $h \in J$. Finalmente, $\mathbb{V}(I.J) \subseteq \mathbb{V}(I) \cup \mathbb{V}(J)$ □

Observación 3.16. Un conjunto finito de puntos es una variedad. En efecto, si $X = \{P_1, \ldots, P_s\}$, con $P_i = (P_{i1}, \ldots, P_{in})$. Sea $I_i = \langle x_1 - P_{i1}, \ldots, x_n - P_{in} \rangle, i = 1, \ldots, s$. Entonces $X = \mathbb{V}(I_1) \cup \cdots \cup \mathbb{V}(I_s)$

Definición 3.17. Sea $X \subseteq k^n$ un conjunto de puntos. Se define el ***ideal de*** X como

$$\mathbb{I}(X) = \{f \in k[x_1, \ldots, x_n] : f(\mathbf{x}) = 0 \; \forall \mathbf{x} \in X\}$$

3.3 Algoritmo de Buchberger-Möller

Este algoritmo nos ayudará a encontrar el conjunto de los polinomios interpoladores de un conjunto finito de puntos en k^n. Primero veamos algunas definiciones. (Para más detalles, ver [Robbiano], [A-K-R])

Dado un conjunto finito de puntos en $k^n, \chi = \{P_1, \ldots, P_s\}$, definimos

$$\mathbb{I}(\chi) := \{f \in k[x_1, \ldots, x_n] : f(P_i) = 0, i = 1, \ldots, s\}$$

el ideal del conjunto, que notaremos con I.

Sea σ un orden monomial, denotamos con $Lt_\sigma(I)$ al ideal generado por los monomios de cabeza, según σ, de los polinomios de I.

Sea ahora $\mathcal{O}_\sigma(I)$ el conjunto de los monomios que no pertenecen a $Lt_\sigma(I)$. Este conjunto nos da una base de $k[x_1, \ldots, x_n]/I$ como k-espacio vectorial, y tiene cardinal s.

Decimos que un poliomio s_i es un separador de P_i de χ si $s_i(P_i) = 1$ y $s_i(P_j) = 0$ para cada $j \neq i$.

El algoritmo de Buchberger-Möller nos permite encontrar la base reducida de I para σ, y al mismo tiempo nos muestra $\mathcal{O}_\sigma(I)$ y un conjunto de separadores de P_i de χ para cada $i = 1, \ldots, s$.

3.3.1 Algoritmo

Algoritmo 1. *(Algoritmo de Buchberger-Möller)*

ENTRADA:
$\chi = \{P_1, \ldots, P_s\} \subseteq k^n, \sigma$ *orden monomial*
SALIDA:
$GB =$*la base de Gröbner reducida de* $I(\chi)$ *para* σ, $\mathcal{O} = \mathcal{O}_\sigma(I(\chi))$,
$S =$*una lista de separadores de* P_i *de* χ *para* $i = 1, \ldots, s$

$GB := \emptyset, \mathcal{O} := \emptyset, S := \emptyset$
$r := 0; L := [1]$
Mientras $L \neq \emptyset$ *hacer:*
$T := min_\sigma(L)$
$L := L \setminus \{T\}$
$f := T - \sum_{i=1}^{r} T(P_{\pi(i)}) S_i$
Si $f(P_i) = 0 \, \forall i$
$GB := GB \cup \{f\}$

Si no

$\mathcal{O} := \mathcal{O} \cup \{T\}$
$r := r + 1$
$\pi(r) := min\{i : f(P_i) \neq 0\}$
$S_r := \frac{f}{f(P_{\pi(r)})}$
$S := S \cup S_r$
para $i = 1, \ldots, r-1$
$S_i := S_i - S_i(P_{\pi(r)})S_r$
fin
$L := L \cup \{x_i T :$ *$x_i T$ no es múltiplo de un elemento de L ni de $Lt_\sigma(g)$ con $g \in GB$*$\}$

fin
mostrar $GB, \mathcal{O}, S$
fin

Intentaremos demostrar que la salida de este algoritmo es realmente lo que esperamos.

3.3.2 Algunos lemas previos

Primero fijemos un poco de notación.
Sean $\chi = \{P_1, \ldots, P_s\}$, $I = \mathbb{I}(\chi)$, σ orden monomial
B base de Gröbner reducida de I para σ
$\mathcal{O}_\sigma(I) = k[X] \setminus Lt_\sigma(I)$
$GB^k := GB$ al finalizar el paso k
$T^k := minL^{k-1}$ con $L^{k-1} := L$ al finalizar el paso $k-1$
$f^k := T^k - \sum_{j=1}^{r} T^k(P_{\pi(j)})S_j^{k-1}$ con $S_j^{k-1} := S_j$ al finalizar el paso $k-1$
$\mathcal{O}^k := \mathcal{O}$ al finalizar el paso k

Lema 3.18. $Lt_\sigma(f^k) = T^k \prec_\sigma T^{k+1} = Lt_\sigma(f^{k+1})$

Demostración. Primero veamos la desigualdad:
Sea $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$, definimos $X^\alpha := x_1^{\alpha_1} \ldots x_n^{\alpha_n}$. Si $X^\alpha \in L^k$, veamos que $X^\alpha \succ_\sigma T^k$ y por lo tanto $T^{k+1} = minL^k \succ_\sigma T^k$.
$L^k = (L^{k-1} \setminus \{T^k\}) \cup \{x_j T^k : j \in A \subseteq \{1, \ldots, n\}\}$
Si $X^\alpha \in L^{k-1} \setminus \{T^k\} \subseteq L^{k-1} \Rightarrow X^\alpha \succeq_\sigma T^k$ y $X^\alpha \neq T^k \Rightarrow X^\alpha \succ_\sigma T^k$
Si $X^\alpha = x_j T^k$ para algún $j, 1 \leq j \leq n$, como $x_j \succeq_\sigma 1 \Rightarrow X^\alpha = x_j T^k \succ_\sigma T^k$
$\Rightarrow minL^k \succ_\sigma T^k$ y $minL^k = T^{k+1}$
Ahora veamos que $Lt_\sigma(f^k) = T^k$
Si $k = 1$
$f^1 = 1 - 0$ y $1 = T^1$
Supongamos que $Lt_\sigma(f^j) = T^j \ \forall j \leq k$
$f^{k+1} = T^{k+1} - \sum \alpha_l S_l^k$ pero $Lt_\sigma(S_l^k) = Lt_\sigma(f^j)$ para algún $j \leq k$
$\Rightarrow Lt_\sigma(\sum \alpha_l S_l^k) \preceq_\sigma \max_l\{Lt_\sigma(S_l^k)\} = T^j \preceq_\sigma T^k \prec_\sigma T^{k+1}$
$\Rightarrow Lt_\sigma(f^{k+1}) = T^{k+1}$

□

Lema 3.19. *Sea* $T^k \preceq_\sigma T \prec_\sigma T^{k+1} = minL^k \Rightarrow T = T^k$ *o* $\exists T^j$ *con* $j \leq k$ *tal que* $f^j \in GB^k$ *y* $T^j \mid T$

Demostración. Por inducción en k.

Si $k = 1$, $T^1 = 1$ y $T^2 = min\{x_1, \ldots, x_n\}$. Supongamos $T^2 = x_i$.

Si $T \prec_\sigma x_i$, supongamos que $x_j \mid T \Rightarrow x_j \preceq_\sigma T \prec_\sigma x_i$ y esto es absurdo. Luego, $T = 1 = T^1$

Supongamos que la hipótesis es cierta para todo $j \leq k$, veamos que vale para $k+1$: Sea T tal que $T^{k+1} \preceq_\sigma T \prec_\sigma T^{k+2}$

- Si $T = T^{k+1}$, listo.
- Supongamos que no. Como $1 = T^1 \mid T$, definimos $j_0 = max\{j : T^j \mid T\}$. Tenemos $T = T^{j_0} x_i T'$ para algún i.
  - Si $T^{j_0} x_i \prec_\sigma T^{k+1}$, por hipótesis inductiva $T^{j_0} x_i = T^j$ con $j \leq k$ (y esto sería absurdo porque j_0 es un máximo), o $\exists T^j$ con $j \leq k$ tal que $f^j \in GB^k$ y $T^j \mid T^{j_0} x_i \mid T$, y $GB^k \subseteq GB^{k+1}$
  - Supongamos que $T^{k+1} \preceq_\sigma T^{j_0} x_i$, como j_0 es máximo,
    $T^{k+1} \prec_\sigma T^{j_0} x_i \preceq_\sigma T \prec_\sigma T^{k+2}$

    queremos ver que $\exists T^j$ con $j \leq k+1$ tal que $f^j \in GB^{k+1}$ y $T^j \mid T^{j_0} x_i$ (luego, $T^j \mid T$)
    - Si $f^{j_0}(P_i) = 0 \, \forall i \Rightarrow f^{j_0} \in GB^{j_0} \subseteq GB^{k+1}$ y $T^{j_0} \mid T^{j_0} x_i$
    - Si no, $T^{j_0} \in \mathcal{O}^{j_0}$ y $T^{j_0} x_i \notin L^{j_0}$ pues $L^{j_0} - \{T^{j_0+1}, \ldots, T^{k+1}\} \subseteq L^{k+1}$ (con $T^{j_0} x_i \notin \{T^{j_0+1}, \ldots, T^{k+1}\}$) y $T^{j_0} x_i \prec_\sigma T^{k+2} = minL^{k+1}$. Entonces $T^{j_0} x_i (\neq T^{j_0}) \notin L^{j_0-1}$ y
      1. $T^{j_0} x_i$ es múltiplo de un elemento de $L^{j_0-1} - \{T^{j_0}\}$ Entonces $T^{j_0} x_i = \tilde{T} P$ con $\tilde{T} \in L^{j_0-1} - \{T^{j_0}\}$. Luego, $\tilde{T} \succ_\sigma T^{j_0}$. Y como el algoritmo finaliza cuando $L = \emptyset$, y solo se eliminan de L los elementos mínimos en cada paso, debe ser $\tilde{T} = T^j$ para algún $j \in \mathbb{N}$. Por lo tanto se tiene $T^j \mid T^{j_0} x_i$ y $T^{j_0} x_i \mid T$, y esto implica que $T^j \mid T$ pero $j > j_0$ y j_0 era un máximo. Luego, esto no puede suceder.
      2. $T^{j_0} x_i$ es múltiplo de $Lt_\sigma(g)$ con $g \in GB^{j_0}$. Sea j tal que $g = f^j$, como $GB^{j_0} \subseteq GB^{k+1}$, llegamos a lo que queríamos probar.

□

Lema 3.20. $\mathcal{O}^k = \{T \in \mathcal{O}_\sigma(I) : T \prec_\sigma T^{k+1}\}$

Demostración. Sea $T \in \mathcal{O}_\sigma(I), T \prec_\sigma T^{k+1}$, por lema 3.19 se cumple:

- $\exists f^j \in GB^k$ tal que $T^j \mid T$. Pero entonces, como $f^j \in GB^k \subseteq I, \exists g \in B$ tal que $Lt_\sigma(g) \mid T^j \Rightarrow Lt_\sigma(g) \mid T \in \mathcal{O}_\sigma(I)$ y esto es absurdo.
- $T = T^j$ para algún $j \leq k$
  - si $f^j \in I \Rightarrow \exists g \in B$ tal que $Lt_\sigma(g) \mid T^j \Rightarrow Lt_\sigma(g) \mid T \in \mathcal{O}_\sigma(I)$. Absurdo.

- $f^j \notin I \Rightarrow T^j \in \mathcal{O}^j \subseteq \mathcal{O}^k \Rightarrow T \in \mathcal{O}^k$

Así queda probada una inclusión. Supongamos que la otra no es cierta. Entonces $\exists\, T^j \in \mathcal{O}^k$ tal que $T^j \notin \mathcal{O}_\sigma(I)$. Sea $T^{j_0} = min\{T^j \in \mathcal{O}^k : T^j \notin \mathcal{O}_\sigma(I)\} \Rightarrow \exists\, g \in B$ tal que $Lt_\sigma(g) \mid T^{j_0} \prec_\sigma T^{k+1}$. Entonces

- $\exists\, f^j \in GB^k$ tal que $T^j \mid Lt_\sigma(g) \mid T^{j_0}$. Y esto es absurdo por construcción de L, pues $T^j \prec_\sigma T^{j_0} (f^{j_0} \notin GB^k)$ y $T^j \mid T^{j_0}$.
- $Lt_\sigma(g) = T^j$
  - Si $f^j \in GB^k$, esto es un absurdo por construcción de L.
  - Si $T^j \in \mathcal{O}^j \Rightarrow T^j \prec_\sigma T^{j_0} \Rightarrow T^j \in \mathcal{O}_\sigma(I)$ pero $T^j = Lt_\sigma(g)$ y $g \in B$. Luego, esto no puede suceder.

Por lo tanto, valen las dos inclusiones. □

Lema 3.21. $GB^k = \{g \in B : Lt_\sigma(g) \prec_\sigma T^{k+1}\}$

Demostración. Sea $g \in B, Lt_\sigma(g) \prec_\sigma T^{k+1}$, por Lema 3.19:

- $\exists\, j$ tal que $Lt_\sigma(g) = T^j$
  - Si $f^j \notin I \Rightarrow T^j \in \mathcal{O}^j \subseteq \mathcal{O}_\sigma(I) \Rightarrow Lt_\sigma(g) \in \mathcal{O}_\sigma(I)$. Y esto es absurdo.
  - $f^j \in I \Rightarrow f^j - g \in I$
    1. Si $f^j - g \neq 0 \Rightarrow \exists\, \tilde{g} \in B$ tal que $Lt_\sigma(\tilde{g}) \mid Lt_\sigma(f^j - g)$, pero $Lt_\sigma(f^j - g)$ es monomio de f^j ó de g y es $\prec_\sigma T^j$. Luego, como B es reducida, $Lt_\sigma(f^j - g)$ es monomio de f^j y es distinto de T^j. Y por construcción, $Lt_\sigma(f^j - g) = k_j T^l$ con k_j una constante en k, $T^l \in \mathcal{O}^l \subseteq \mathcal{O}_\sigma(I)$. Luego, $Lt_\sigma(\tilde{g}) \mid T^l \in \mathcal{O}_\sigma(I)$. Y esto es absurdo.
    2. Si $f^j = g$ listo.
- $\exists\, f^j \in GB^k$ tal que $T^j \mid Lt_\sigma(g)$ pero $f^j \in GB^k \Rightarrow f^j \in I \Rightarrow \exists\, \tilde{g} \in B$ tal que $Lt_\sigma(\tilde{g}) \mid Lt_\sigma(g)$. Entonces, como B es reducida, $Lt_\sigma(\tilde{g}) = T^j = Lt_\sigma(g)$ y estamos en la situación anterior.

Sea ahora $f^j \in GB^k \subseteq I \Rightarrow \exists\, g \in B$ tal que $Lt_\sigma(g) \mid T^j \prec_\sigma T^{k+1}$. Entonces, por la otra inclusión, $\exists\, f^l \in GB^k$ tal que $g = f^l$ y $T^l \mid T^j$. Entonces, por construcción de L, $l = j$ y $f^j = g \in B$, y $Lt_\sigma(g) \prec_\sigma T^{k+1}$.

□

3.3.3 Teorema

Teorema 3.22. *Dados $\chi = \{P_1, \ldots, P_s\} \subseteq k^n, \sigma$ un orden monomial, el algoritmo de Buchberger-Möller termina y calcula la base de Gröbner reducida de $\mathbb{I}(\chi)$ para σ, $\mathcal{O} = \mathcal{O}_\sigma(\mathbb{I}(\chi))$ y una lista de separadores de P_i de χ.*

Demostración. Primero veamos que termina. En cada iteración se agrega un elemento a GB o a $\mathcal{O}$. Como GB y $\mathcal{O}$ están contenidos en conjuntos finitos (por lemas 3.20 y 3.21, respectivamente), si el algoritmo es correcto, sólo puede tener finitos pasos. Y cada paso claramente involucra un número finito de cálculos.

Para ver que es correcto, suponiendo que al comenzar cada iteración, los valores de GB y $\mathcal{O}$ son correctos, tenemos que ver que siguen siendo correctos al final de la iteración. Pero por lo visto en los lemas 3.20 y 3.21, como en cada iteración se agrega un elemento a GB o a $\mathcal{O}$, y éstas están contenidas en B y $\mathcal{O}_\sigma(\mathbb{I}(\chi))$ respectivamente, efectivamente se obtienen valores apropiados.

□

3.3.4 Interpoladores

Con el algoritmo de Buchberger-Möller descripto arriba es realmente sencillo encontrar $f \in k[x_1, \ldots, x_n]$ tal que $f(P_i) = a_i$, para $i = 1, \ldots, n$.

Una vez que tenemos la lista de separadores, $\{S_1, \ldots, S_s\}$, podemos construir un polinomio interpolador de la siguiente manera:

$$f = \sum_{i=1}^{s} a_i S_i$$

De este modo, $f(P_j) = \sum_{i=1}^{s} a_i S_i(P_j) = a_j \cdot 1 = a_j$.

3.3.5 Cuando la cantidad de puntos es mucho menor que la cantidad de variables

Al estudiar las redes reguladoras de genes, lo que suele suceder es que la información se limita a decenas de puntos, mientras que hay miles de genes o variables. En [J-S] se presenta un algoritmo diseñado especialmente para calcular bases de Gröbner de ideales cero dimensionales que se optimiza cuando la cantidad de puntos es mucho menor que la cantidad de indeterminadas.

Capítulo 4

Ejemplo de ingeniería reversa utilizando bases de Gröbner

En este capítulo haremos una breve descripción del algoritmo utilizado en [L-S] y en el capítulo 5 se desarrollará con más cuidado la mejora presentada en [J-L-S-S].

Asumimos que el conjunto de estados es un cuerpo finito, denotado por k. El modelo f buscado está determinado por sus funciones coordenadas $f_i : k^n \to k$. Y se le aplicará ingeniería reversa a cada coordenada independientemente para ir reconstruyendo la red de a un nodo por vez.

Para cada i, f_i es minimal en el sentido de que no existe un polinomio no nulo $g_i \in k[x_1, \ldots, x_n]$, que surja de reordenar y reagrupar los términos de f_i, tal que $f_i = h_i + g_i$ y g_i se anule en todos los puntos dados. Es decir, se excluyen todos los términos en los polinomios f_i que se anulan en los datos.

Entonces, el problema es el siguiente:

Dadas una colección de estados $\mathbf{s}_1 = (s_{11}, \ldots, s_{n1}), \ldots, \mathbf{s}_m = (s_{1m}, \ldots, s_{nm})$ (donde m suele ser mucho menor que la cantidad de elementos de k^n) tales que $\mathbf{s}_{j+1}$ es el estado siguiente a $\mathbf{s}_j$ para $j = 1, \ldots, m-1$ (es decir, $\mathbf{s}_1, \ldots, \mathbf{s}_m$ es una serie de tiempo), y una determinada coordenada $i \in \{1, \ldots, n\}$.

1. Encontrar todas las funciones polinomiales $f_i \in k[x_1, \ldots, x_n]$ tales que

$$f_i(\mathbf{s}_j) = s_{ij+1}$$

   Es decir, encontrar todos los polinomios que mandan $\mathbf{s}_j$ a la i-ésima coordenada del siguiente punto $\mathbf{s}_{j+1}$.

2. De ese conjunto de funciones, elegir una que no contenga términos que se anulen en todos los puntos.

Fijamos una coordenada/nodo en la red y le aplicamos ingeniería reversa a su función de transición. Para simplificar la notación, supongamos que tenemos una serie de tiempo $\mathbf{s}_1, \ldots, \mathbf{s}_m$, junto con elementos $a_1, \ldots, a_m \in k$, y buscamos todas las funciones polinomiales $f \in k[x_1, \ldots, x_n]$ tales que $f(\mathbf{s}_j) = a_j, \forall j = 1, \ldots, n$. Primero calculamos un polinomio particular f_0 que interpole (utilizando el algoritmo visto en el capítulo 3, por ejemplo).

Ahora, si consideramos dos polinomios $f, g \in k[x_1, \ldots, x_n]$ tales que

$$f(\mathbf{s}_j) = a_j = g(\mathbf{s}_j)$$

Luego, $(f-g)(\mathbf{s}_j) = 0$, $\forall j$. O sea, dos interpoladores difieren en un polinomio del ideal de $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}, I$.

El siguiente paso consiste en reducir al polinomio f_0 módulo el ideal I. Es decir, escribir f_0 como

$$f_0 = f + g$$

con $g \in I$. Más aún, f es minimal en el sentido de que no se puede seguir descomponiendo como $f' + h$ con $h \in I$. En otras palabras, g representa la parte de f_0 que está en I, y por lo tanto se anula en todos los puntos de la serie de tiempo. Luego, obtenemos todas las posibles funciones que interpolan a la serie de tiempo en la forma $f + g$, donde g puede ser cualquier elemento de I.

Algoritmo 2. *(Algoritmo de ingeniería reversa (Para un nodo de la red))*

ENTRADA: *Una serie de tiempo $\mathbf{s}_1, \ldots, \mathbf{s}_m \in k^n$ de estados de la red, junto con niveles de expresión $a_1, \ldots, a_m \in k$.*

SALIDA: *Una función polinomial $f \in k[x_1, \ldots, x_n]$ tal que $f(\mathbf{s}_j) = a_j$ y tal que f no contiene componentes polinomiales que se anulan en la serie de tiempo.*

PASO 1: *Calcular una solución particular f_0.*

PASO 2: *Calcular el ideal I de todas las funciones que se anulan en los datos.*

PASO 3: *Calcular la reducción f de f_0 con respecto a I.*

Como vimos en el capítulo anterior, los cálculos para encontrar I t f se basan en el cálculo de una base de Gröbner. En este proceso, se necesita la elección de algún orden monomial. El resultado final de los cálculos depende del orden que se haya elegido. Y, como dijimos en la introducción, no hay una elección de orden monomial que sea canónica.

4.1 Aplicación

Cuando se diseñan herramientas para modelar, es importante testearlas en sistemas que se conocen bien, así se puede medir la cantidad de información que identifica el modelo. Con este fin, el método descripto en este capítulo fue validado al aplicarlo a un conjunto de datos simulados de la red de los genes de la polaridad de los segmentos en la mosca de la fruta *Drosophila melanogaster*, que es una red bien estudiada. En el cuadro 4.1 se propone un modelo Booleano para la red de 5 genes y sus proteínas asociadas. La red consiste de un anillo de 12 células interconectadas, agrupadas en tres parasegmentos primigenios, en los cuales los genes se expresan en cada una de las cuartas células. El objetivo en [L-S] fue aplicarle ingeniería reversa a la red de interacciones entre especies moleculares, como también a la dinámica a través de la identificación de interacciones aditivas y no aditivas. Observemos que, para estos propósitos, es irrelevante si el modelo Booleano es efectivamente correcto, solo interesa verificar si el algoritmo lo aproxima bien (sin importar qué está aproximando y cuán correcto sea).

Los genes representados en el modelo Booleano son cinco (*wg, en, hh, ptc, ci*). También se incluyen las proteínas codificadas por estos genes, como también otros dos componentes (*SMO* y *SLP*), costituyendo 15 especies moleculares distintas (ver [A-O] o [Stigler]). La figura 4.1 muestra el "wiring diagram" de las conexiones en el modelo Booleano.

Cuadro 4.1: Las funciones Booleanas están basadas en las interacciones conocidas entre mRNAs y proteínas. Los superíndices denotan tiempo, y los subíndices ubicación relativa a la célula estudiada. En general, la regla de actualización da la expresión de un nodo en el tiempo $t+1$ como función de la expresión de sus nodos efectores en el tiempo t. Sin embargo, hay tres excepciones: se asume que la expresión de SLP no cambia, y que la activación de SMO y la unión de PTC a HH son instantáneas.

Nodo	Función Booleana de actualización
SLP_i	$SLP_i^{t+1} = SLP_i^t = \begin{cases} 0 & si \quad i \equiv 1(mod\,4) \qquad i \equiv 2(mod\,4) \\ 1 & si \quad i \equiv 3(mod\,4) \qquad i \equiv 0(mod\,4) \end{cases}$
wg_i	$wg_i^{t+1} = (CIA_i^t \wedge SLP_i^t \wedge \neg CIR_i^t) \vee [wg_i^t \wedge (CIA_i^t \vee SLP_i^t) \wedge \neg CIR_i^t]$
WG_i	$WG_i^{t+1} = wg_i^t$
en_i	$en_i^{t+1} = (WG_{i-1}^t \vee WG_{i+1}^t) \wedge \neg SLP_i^t$
EN_i	$EN_i^{t+1} = en_i^t$
hh_i	$hh_i^{t+1} = EN_i^t \wedge \neg CIR_i^t$
HH_i	$HH_i^{t+1} = hh_i^t$
ptc_i	$ptc_i^{t+1} = CIA_i^{t+1} \wedge \neg EN_i^{t+1} \wedge \neg CIR_i^{t+1}$
PTC_i	$PTC_i^{t+1} = ptc_i^t \vee (PTC_i^t \wedge \neg HH_{i-1}^t \wedge \neg HH_{i+1}^t)$
PH_i	$PH_i^t = PTC_i^t \wedge (HH_{i-1}^t \vee HH_{i+1}^t)$
SMO_i	$SMO_i^t = \neg PTC_i^t \vee HH_{i-1}^t \vee HH_{i+1}^t$
ci_i	$ci_i^{t+1} = \neg EN_i^t$
CI_i	$CI_i^{t+1} = ci_i^t$
CIA_i	$CIA_i^{t+1} = CI_i^t \wedge (SMO_i^t \vee hh_{i-1}^t \vee hh_{i+1}^t)$
CIR_i	$CIR_i^{t+1} = CI_i^t \wedge \neg SMO_i^t \wedge \neg hh_{i\pm1}$

En el grafo, los nodos representan mRNAs y proteínas. Una arista A $\rightarrow$ B entre los nodos de proteínas A y B implica que A regula la síntesis de B, mientras que una arista A $\rightarrow$ b de la proteína A al mRNA b implica que A regula la síntesis de b, es decir, la transcripción del gen b. Las aristas denotan la existencia de regulación, no el tipo, sea activación o inhibición.

Se concentra la atención en la reconstrucción de la red para una célula, teniendo en cuenta las conexiones intercelulares. Para más detalles, ver [A-O]. Para representar las dependencias intercelulares se incluyeron 6 nodos extras en el modelo. En el cuadro 4.2 se muestra la traducción de las funciones Booleanas del cuadro 4.1 en funciones polinomiales en las variables $x_1, \ldots, x_{21}$, con coeficientes en $\mathbb{Z}_2$. Y debajo aparecen las leyendas con los nombres de las variables.

En [L-S] trataron de reconstruir un modelo polinomial discreto usando solo los datos de expresión de tipo silvestre y su modelo encontró 20 enlaces en la red, de los cuales 14

Cuadro 4.2: Representación polinomial de las funciones Booleanas de la red, junto con la leyenda de los nombres de las variables.

$$
\begin{aligned}
F_1 &= x_1 \\
F_2 &= (x_{15}+1)(x_1x_{14}+x_2(x_1+x_{14}+x_1x_{14})+x_1x_2x_{14}(x_1+x_{14}+x_1x_{14})) \\
F_3 &= x_2 \\
F_4 &= (x_{16}+x_{17}+x_{16}x_{17})(x_1+1) \\
F_5 &= x_4 \\
F_6 &= x_5(x_{15}+1) \\
F_7 &= x_6 \\
F_8 &= x_{13}((x_{11}+x_{20}+x_{11}x_{20})+x_{21}+(x_{11}+x_{20}+x_{11}x_{20})x_{21})(x_4+1) \\
&\quad (x_{13}(x_{11}+1)(x_{20}+1)(x_{21}+1)+1) \\
F_9 &= x_8+x_9(x_{18}+1)(x_{19}+1)+x_8x_9(x_{18}+1)(x_{19}+1) \\
F_{10} &= (x_8+x_9(x_{18}+1)(x_{19}+1)+x_8x_9(x_{18}+1)(x_{19}+1))(x_{20}+x_{21}+x_{20}x_{21}) \\
F_{11} &= x_8+x_9Y+x_8x_9Y+1+x_{20}+((x_8+x_9Y+x_8x_9Y+1)x_{20})+x_{21}+ \\
&\quad +(x_8+x_9Y+x_8x_9Y+1+x_{20}+(x_8+x_9Y+x_8x_9Y+1)x_{20})x_{21} \\
F_{12} &= x_5+1 \\
F_{13} &= x_{12} \\
F_{14} &= x_{13}((x_{11}+x_{20}+x_{11}x_{20})+x_{21}+(x_{11}+x_{20}+x_{11}x_{20})x_{21}) \\
F_{15} &= x_{13}(x_{11}+1)(x_{20}+1)(x_{21}+1)
\end{aligned}
$$

SLP_i	wg_i	WG_i	en_i	EN_i	hh_i	HH_i	ptc_i
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8

PTC_i	PH_i	SMO_i	ci_i	CI_i	CIA_i	CIR_i
x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

WG_{i-1}	WG_{i+1}	HH_{i-1}	HH_{i+1}	hh_{i-1}	hh_{i+1}	Y
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	$(x_{18}+1)(x_{19}+1)$

realmente están (la red tiene 44 enlaces). El rendimiento del algoritmo mejoró notablemente cuando incorporaron series de tiempo "noqueadas"(*knock-out*) para los cinco genes (ver sección 5.3).

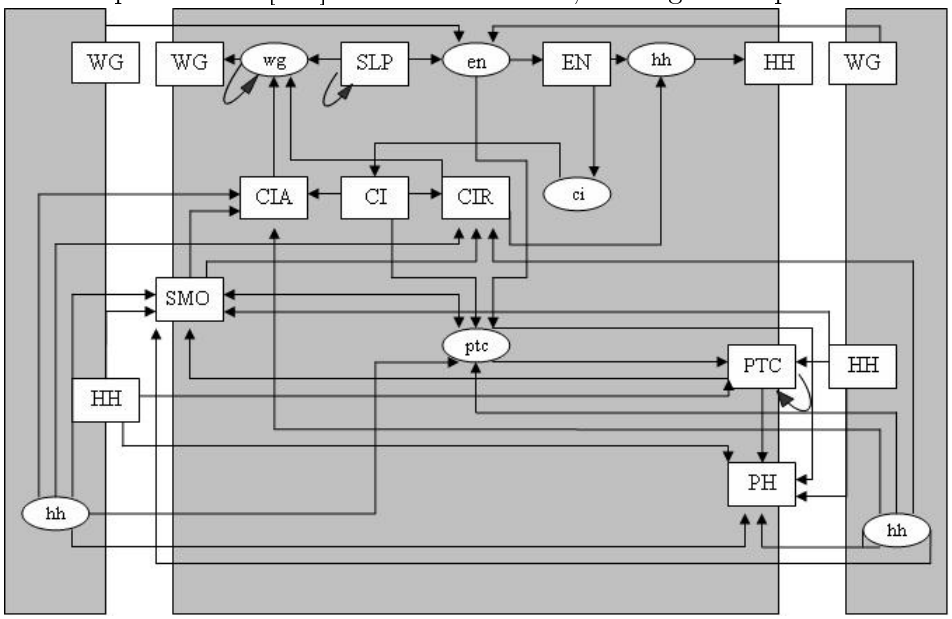
Para simular un experimento en el cual el nodo x_i que representa un gen está noqueado, se considera nula su correspondiente función de actualización F_i (y las otras funciones permanecen iguales). También se anulan las correspondientes funciones en las células vecinas. Se genera luego una nueva serie de tiempo iterando las inicializaciones dadas. Con esta información, en [L-S], construyeron el modelo de la figura 4.2.

Primero se construyen todos los modelos polinomiales que se acomodan a una serie de tiempo discreta. Luego, se elige uno que es minimal con respecto a los sumandos en cada una de las funciones que se anulan en los puntos de tiempo. El algoritmo se basa en la elección de un orden monomial, que involucra un orden entre las mismas variables. El efecto de tal orden en las variables es que aquellas menores para este orden se usan con preferencia en el algoritmo. Para contrarrestar esta dependencia, en [L-S] usaron un modelo consensual extraído de cuatro elecciones diferentes de orden para las variables. Usando cada orden, corrieron el algoritmo e intersecaron los resultados para obtener un modelo consensual.

El modelo consensual que construyeron, que incorpora información de los mutantes silvestres y noqueados, predice 37 de los 44 enlaces en la red Booleana.

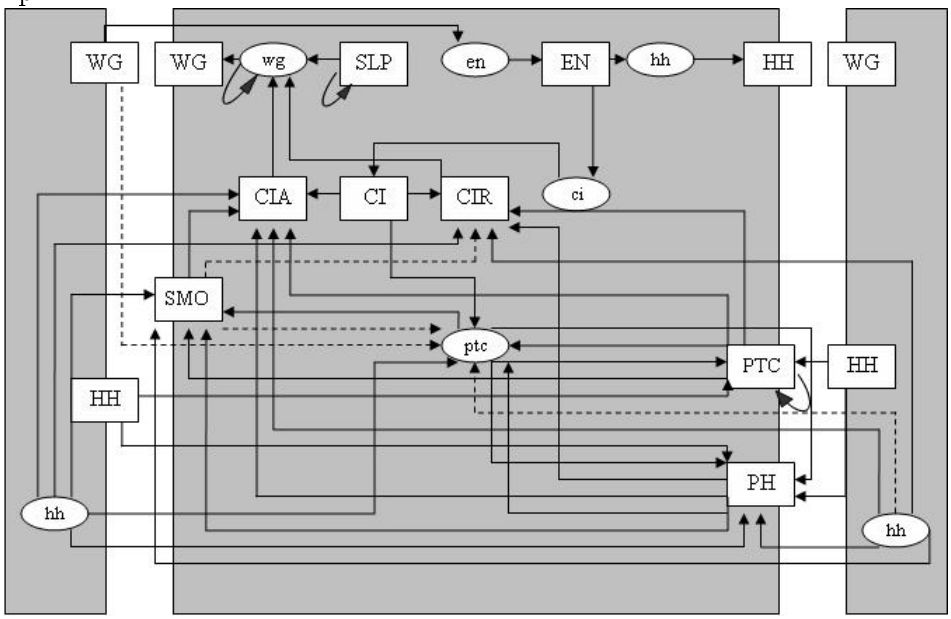
Centrando la atención en el proceso de ingeniería reversa de la dinámica del modelo Booleano, las funciones en este modelo contienen términos que se anulan en todos los puntos de tiempo, así que resulta imposible detectar las correspondientes relaciones. Para poder comparar con el modelo Booleano a la dinámica predicha por el método, redujeron los polinomios en el cuadro 4.2. Esta reducción, por supuesto, depende de la elección del

Figura 4.1: El grafo de interacciones en una célula de la red con interacciones celulares, como se presenta en [L-S]. Óvalos = mRNAs, rectángulos = proteínas.



orden monomial, al igual que la salida del algoritmo. Para cada elección de orden monomial, las funciones reducidas del modelo Booleano y las funciones obtenidas con ingeniería reversa a partir de los datos dados coincidieron. Esto muestra que se puede predecir completamente la dinámica del modelo Booleano a partir de la información de los tipos silvestres y noqueados. Sin embargo, dada la sensibilidad del método al orden monomial elegido, la forma particular de las funciones obtenidas aplicando ingeniería reversa no puede ser directamente interpretada en términos de relaciones regulatorias.

Figura 4.2: El grafo del modelo consensual, construido a partir de las series temporales de los tipos silvestre y noqueado. El grafo es la intersección de cuatro modelos polinomiales, uno para cada orden monomial usado. Las líneas punteadas representan enlaces que aparecen en tres de los cuatro modelos.



Capítulo 5

Ejemplo de ingeniería reversa utilizando descomposición primaria de ideales monomiales

En este capítulo desarrollaremos el método propuesto en [J-L-S-S], que surgió como mejora del método visto en el capítulo anterior. Y en la última sección describiremos abreviadamente un algoritmo propuesto en [Krupa] para resolver el mismo problema pero en tiempo polinomial.

Supongamos que consideramos n variables, $x_1, \ldots, x_n$. Supongamos que hemos medido m pares de transición de estado con valores reales, $(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m)$ con

$$\mathbf{s}_i = (s_{i1}, \ldots, s_{in}), \mathbf{t}_i = (t_{i1}, \ldots, t_{in}), i = 1, \ldots, m,$$

donde $s_{ij}, t_{ij} \in \mathbb{R}$. Esto es, si el sistema se encuentra en estado $\mathbf{s}_i$, en el siguiente paso temporal se encontrará en el estado $\mathbf{t}_i$. De aquí, se discretiza la información en un cuerpo finito k y se obtienen pares asociados discretos $\mathbf{s}_i, \mathbf{t}_i$ en k^n. Un modelo admisible es una función $f : k^n \to k^n$ tal que

$$f(\mathbf{s}_i) = \mathbf{t}_i \tag{5.1}$$

Como cualquier función $k^n \to k$ se puede representar como una función polinomial, el espacio modelo a considerar es la colección de todos los sistemas polinomiales dinámicos $f = (f_1, \ldots, f_n)$ que satisfacen la ecuación 5.1.

Se puede observar que, si f y g son dos modelos, luego $f - g$ es una función polinomial que se anula en los puntos dados. Por lo tanto, podemos calcular el espacio modelo a partir de un modelo particular $f^0 = (f_1^0, \ldots, f_n^0)$, donde la función coordenada $f_i^0 \in k[x_1, \ldots, x_n]$, y el ideal I de la variedad dada por los puntos $\mathbf{s}_i$. El espacio modelo está dado por las n-uplas de coclases $(f_1^0 + I, \ldots, f_n^0 + I)$. Y es claro ver que basta trabajar coordenada a coordenada.

Trabajando sobre una sola coordenada y sin hallar un modelo particular del espacio $h + I$, donde $h \in k[x_1, \ldots, x_n]$, se intentará encontrar aquellos conjuntos de variables minimales $\{x_{i_1}, \ldots, x_{i_r}\}$ para los cuales se verifica

$$k[x_{i_1}, \ldots, x_{i_r}] \cap (h + I) \neq \emptyset.$$

Entonces tenemos el siguiente conjunto de datos

$$(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m)$$

donde $\mathbf{s}_i \in k^n$ y $t_i \in k$. Se buscan funciones $f \in k[x_1, \ldots, x_n]$ tales que $f(\mathbf{s}_i) = t_i$. Para $a \in k$, sea

$$X_a = \{\mathbf{s}_i : t_i = a\}$$

y $X = \{X_a : a \in k\}$. Se denota al espacio modelo $h + I$ para el conjunto de datos dado como

$$Y = \{f \in k[x_1, \ldots, x_n] : f(\mathbf{p}) = a, \forall \mathbf{p} \in X_a, a \in k\}$$

Sea $F \subset \{1, \ldots, n\}$ y sea $R_F = k[x_i : i \notin F]$. Se define

$$\Delta_X := \{F \subset \{1, \ldots, n\} : Y \cap R_F \neq \emptyset\}$$

Sea ahora $M_X \subseteq k[x_1, \ldots, x_n]$ el ideal monomial libre de cuadrados generado por

$$W = \{m(\mathbf{p}, \mathbf{q}) : \mathbf{p} \in X_a, \mathbf{q} \in X_b, \, y \, a \neq b \in k\}$$

donde

$$m(\mathbf{p}, \mathbf{q}) := \prod_{p_i \neq q_i} x_i$$

Proposición 5.1. *Para un subconjunto dado $F \subset \{1, \ldots, n\}$, $F \in \Delta_X$ si y solo si el ideal $\langle x_i : i \notin F \rangle$ contiene al ideal monomial M_X*

Demostración. Supongamos primero que $F \in \Delta_X$. Luego, $Y \cap R_F \neq \emptyset$. Sean $\mathbf{p} \in X_a, \mathbf{q} \in X_b$ con $a \neq b$, y sea $f \in k[x_i : i \notin F]$ tal que $f(\mathbf{p}) = a$ y $f(\mathbf{q}) = b$. Luego, $\mathbf{p}, \mathbf{q}$ difieren en alguna coordenada $j \notin F$. Y esto implica que $m(\mathbf{p}, \mathbf{q})$ tiene a x_j como factor y por lo tanto $m(\mathbf{p}, \mathbf{q}) \in \langle x_i : i \notin F \rangle$.

Supongamos ahora que M_X está contenido en $\langle x_i : i \notin F \rangle$.

Sea $h \in Y$. Sea $\varphi := \pi_{i_1} \times \cdots \times \pi_{i_m} : k^n \to k^m$ la proyección en las $i_1, \ldots, i_m$-ésimas coordenadas, donde $F^C = \{i_1, \ldots, i_m\}, m \leq n$. Vale que si $\varphi(\mathbf{p}) = \varphi(\mathbf{q})$ y $\mathbf{p}, \mathbf{q} \in X$, entonces $h(\mathbf{p}) = h(\mathbf{q})$. En efecto, si $h(\mathbf{p}) \neq h(\mathbf{q})$, podemos considerar $m(\mathbf{p}, \mathbf{q})$, y $m(\mathbf{p}, \mathbf{q}) \in M_X \subset \langle x_i : i \notin F \rangle$. Por lo tanto, existe $i \notin F$ tal que $p_i \neq q_i$. Y entonces existe $j, 1 \leq j \leq m$, tal que $p_{i_j} \neq q_{i_j}$. Pero $p_{i_j} = \pi_{i_j}(\mathbf{p})$ y $q_{i_j} = \pi_{i_j}(\mathbf{q})$ y por lo tanto $\varphi(\mathbf{p}) \neq \varphi(\mathbf{q})$.

Sea g tal que $g(\varphi(\mathbf{p})) = a \, \forall \mathbf{p} \in X_a, \, \forall a \in k$. g está bien definida pues si $\varphi(\mathbf{p}) = \varphi(\mathbf{q})$ entonces $h(\mathbf{p}) = h(\mathbf{q})$ (i.e. $\mathbf{p}, \mathbf{q} \in X_a$)y $g \in k[x_{i_1}, \ldots, x_{i_m}]$.

Por lo tanto, podemos tomar $f \in Y \cap k[x_{i_1}, \ldots, x_{i_m}], f(\mathbf{p}) := g(\varphi(\mathbf{p}))$. Luego, $f \in Y \cap R_F$.

□

Lema 5.2.

$$M_X = \bigcap_{F \in \Delta_X} \langle x_i, i \notin F \rangle$$

Demostración. Probemos las dos inclusiones.

$\subseteq$) Es inmediata de la Proposición 5.1.

$\supseteq$) Sea $M_X = \bigcap q_i$ la descomposición primaria minimal de M_X. Por construcción, como M_X es monomial y libre de cuadrados, $q_i = \langle x_{i_1}, \ldots, x_{i_r} \rangle$. Como $M_X \subseteq q_i$, sea $F_i \subseteq \{1, \ldots, n\}$ tal que $q_i = \langle x_j : j \notin F_i \rangle$, por la Proposición 5.1 podemos afirmar que $F_i \in \Delta_X (\forall i)$. Y entonces

$$M_X = \bigcap_{F_i \in \Delta_X} \langle x_j : j \notin F_i \rangle \supseteq \bigcap_{F \in \Delta_X} \langle x_j : j \notin F \rangle$$

□

Observación 5.3. $\langle x_i, i \notin F \rangle \subseteq \langle x_i, i \notin F' \rangle \Leftrightarrow F^C \subseteq F'^C \Leftrightarrow F' \subseteq F$

Sean $\{F_1, \ldots, F_N\} \subseteq \Delta_X$ tal que no existe $F' \in \Delta_X$ con $F' \subsetneq F_j (j = 1, \ldots, N)$

$\Rightarrow M_X = \bigcap_{j=1}^{N} \langle x_i, i \notin F_j \rangle$ es la descomposición primaria minimal.

Corolario 5.4. *Los subconjuntos F maximales (para $\subseteq$) tales que $Y \cap R_F \neq \emptyset$ son precisamente el complemento de los conjuntos de índices de los generadores para los primos minimales en la descomposición primaria del ideal M_X.*

5.1 Algoritmo

Algoritmo 3. *(para obtener los conjuntos maximales)*

ENTRADA: $\{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m)\}$, *con* $\mathbf{s}_i \in k^n, t_i \in k$

SALIDA: *Todos los conjuntos maximales* $F \subset \{1, \ldots, n\}$ *tales que existe* $f \in k[x_i, i \notin F]$ *función polinomial con* $f(\mathbf{s}_i) = t_i$

PASO 1: *Calcular el ideal* M_X

PASO 2: *Calcular la descomposición primaria de* M_X

PASO 3: *Calcular los conjuntos de generadores de todos los primos minimales de* M_X

Ejemplo 5.5. Sea $k = \mathbb{F}_5$

$$\begin{array}{ll} \mathbf{s}_1 = (3,0,0,0,0) & t_1 = 3 \\ \mathbf{s}_2 = (0,1,2,1,4) & t_2 = 1 \\ \mathbf{s}_3 = (0,1,2,1,0) & t_3 = 0 \\ \mathbf{s}_4 = (0,1,2,1,1) & t_4 = 0 \\ \mathbf{s}_5 = (1,1,1,1,3) & t_5 = 4 \end{array}$$

Luego, $X_0 = \{\mathbf{s}_3, \mathbf{s}_4\}; X_1 = \{\mathbf{s}_2\}; X_2 = \emptyset; X_3 = \{\mathbf{s}_1\}; X_4 = \{\mathbf{s}_5\}$

$M_X = \langle x_1x_2x_3x_4x_5, x_1x_2x_3x_4, x_5, x_1x_3x_5 \rangle = \langle x_1x_2x_3x_4, x_5 \rangle = \langle x_1, x_5 \rangle \cap \langle x_2, x_5 \rangle \cap \langle x_3, x_5 \rangle \cap \langle x_4, x_5 \rangle$

$\Rightarrow \{1,5\}^C, \{2,5\}^C, \{3,5\}^C, \{4,5\}^C \in \Delta_X$

Luego,

$\{2,3,4\}, \{1,3,4\}, \{1,2,4\}, \{1,2,3\}$ son los conjuntos maximales F

Por lo tanto, $Y \cap k[x_1, x_5] \neq \emptyset$, $Y \cap k[x_2, x_5] \neq \emptyset$, $Y \cap k[x_3, x_5] \neq \emptyset$, $Y \cap k[x_4, x_5] \neq \emptyset$.

Este algoritmo se ha utilizado en un ejemplo biológico con poco más de 80 variables, y dio como resultado 250.000 posibles conjuntos minimales. Y observemos que, aunque puede resultar lento, da como resultado *todos* los posibles conjuntos minimales.

5.2 Elección del modelo

Este algoritmo puede producir una gran cantidad de posibles modelos para un conjunto dado de datos. Si se dispone de información adicional sobre la red, como por ejemplo la existencia o ausencia de ciertas interacciones, ésta se puede utilizar para elegir el modelo. Si no se posee información extra, en [J-L-S-S] se describe una pequeña colección de medidas estadísticas [1] para la selección del modelo. Se consideran los complementos de los conjuntos de salida del algoritmo. Algunas de las medidas prefieren los conjuntos pequeños, y otras prefieren conjuntos grandes que contienen variables que aparecen en muchos de los conjuntos minimales. Dependiendo del tipo de red que se considera, una medida será más apropiada que otra.

Sea $\{x_1, \ldots, x_n\}$ el conjunto de variables y sea

$$\mathcal{F} = \{F_1, \ldots, F_t\} \subseteq \mathcal{P}(\{x_1, \ldots, x_n\}),$$

los complementos de la salida del algoritmo para un cierto conjunto de datos. Se construyen algunas medidas estadísticas sobre esta salida que permite la elección de uno o más subconjuntos/modelos con mayor probabilidad.

Para $1 \leq s \leq n$, sea Z_s la cantidad de conjuntos F_j en $\mathcal{F}$ de longitud s, y para cada $x_i \in \{x_1, \ldots, x_n\}$, sea $W_i(s)$ la cantidad de conjuntos F_j en $\mathcal{F}$ de longitud s tales que $x_i \in F_j$. Esto es,

$$Z_s = |\{j : F_j \in \mathcal{F} \; y \; |F_j| = s\}|$$

$$W_i(s) = |\{j : F_j \in \mathcal{F}, x_i \in F_j \; y \; |F_j| = s\}|$$

Para darle un puntaje a cada variable $x_i \in \{x_1, \ldots, x_n\}$ se proponen tres métodos diferentes. Sean

$$S_1(x_i) = \sum_{s=1}^{n} \frac{W_i(s)}{s.Z_s}$$

$$S_2(x_i) = \sum_{s=1}^{n} \frac{W_i(s)}{s}$$

$$S_3(x_i) = \sum_{s=1}^{n} W_i(s)$$

Y para darle un puntaje a cada conjunto $F_j \in \mathcal{F}$ se proponen dos métodos diferentes. Se definen

$$T_1(F_j) = \prod_{x_i \in F_j} S(x_i)$$

[1] para más información, ver [D-P]

$$T_2(F_j) = \frac{\sum\limits_{x_i \in F_j} S(x_i)}{|F_j|}$$

donde $S(x_i)$ es el puntaje de x_i usando cualquiera de los tres métodos descriptos anteriormente.

Si normalizamos los conjuntos de puntajes dividiendo por $D = \sum_j T_i(F_j), i = 1, 2$ obtenemos entonces una distribución de probabilidad en el conjunto $\mathcal{F}$.

Ejemplo 5.6. Sea $n = 6$, y sea

$$\mathcal{F} = \{F_1 = \{x_1\}; F_2 = \{x_2, x_3\}; F_3 = \{x_2, x_4\}; F_4 = \{x_3, x_5, x_6\}\}$$

Entonces

	S_1	S_2	S_3
x_1	1	1	1
x_2	1/2	1	2
x_3	7/12	5/6	2
x_4	1/4	1/2	1
x_5	1/3	1/3	1
x_6	1/3	1/3	1

El siguiente cuadro muestra el puntaje de los conjuntos usando diferentes combinaciones de los métodos para asignar puntajes S_i y T_j.

	S_1, T_1	S_1, T_2	S_2, T_1	S_2, T_2	S_3, T_1	S_3, T_2
F_1	1	1	1	1	1	1
F_2	7/24	13/24	20/24	22/24	4	2
F_3	1/8	3/8	4/8	6/8	2	12/8
F_4	7/108	45/108	10/108	54/108	2	4/3

Tanto con T_1 como con T_2 se elegiría F_1 si se usara S_1 ó S_2, y F_2 si se usara S_3.

Vale notar que usando S_1 ó S_2 para darle puntaje a las variables no siempre se escogen los conjuntos de un solo elemento, como uno podría imaginarse a partir del ejemplo anterior. Supongamos que tenemos la colección de conjuntos

$$\{x_1\}, \{x_{13}\}, \{x_2, x_3\}\ \{x_2, x_4, x_5\}, \{x_2, x_6, x_7\}, \{x_2, x_8, x_9\}, \{x_2, x_{10}, x_{11}, x_{12}\}$$

Entonces $T(\{x_2, x_3\}) > T(\{x_1\})$, donde T es T_1 ó T_2, usando cualquiera de los métodos para darle puntaje a las variables anteriormente descriptos.

En [J-L-S-S] se describe un posible algoritmo para elegir el modelo.

Algoritmo 4. *(Elección del modelo)*

ENTRADA: *Una colección de subconjuntos $F_1, \ldots, F_t$ de $\{x_1, \ldots, x_n\}$.*

SALIDA: *El(los) conjunto(s) con mayor puntaje.*

PASO 1: *Para cada x_i, calcular $S(x_i)$*

PASO 2: *Para cada F_j, calcular $T(F_j)/D$.*

PASO 3*: Devolver el(los) conjunto(s) F_j con el mayor puntaje $T(F_j)/D$.*

Para obtener un solo modelo, se puede querer hacer una nueva selección basada en información adicional sobre la red a modelar o en otros criterios, como querer elegir el

modelo más simple. Generalmente, este es un proceso heurístico. Pero también se puede usar esta información para obtener puntos de datos adicionales para una siguiente elección del modelo. Para aplicaciones a sistemas biológicos, en particular a redes bioquímicas, algunas moléculas suelen tener propiedades bien conocidas. Este tipo de información se puede usar en el proceso de selección del modelo, y es generado por la siguiente modificación del algoritmo 4

Algoritmo 5. *(Elección del modelo-Información adicional)*

ENTRADA: *Una colección de subconjuntos $F_1, \ldots, F_t$ de $\{x_1, \ldots, x_n\}$*

SALIDA: *Devuelve todos los conjuntos de un solo elemento y la(s) variable(s) y el(los) conjunto(s) con mayor puntaje*

PASO 1: *Para cada x_i, calcular $S(x_i)$.*

PASO 2: *Para cada F_j, calcular $T(F_j)/D$.*

PASO 3: *Devolver el(los) conjunto(s) F_j con la mayor probabilidad $T(F_j)/D$ y todas las variables que tienen puntajes iguales o mayores que el mínimo puntaje de variables que aparece en el F_j elegido en el algoritmo 4.*

El algoritmo 3 y las medidas estadísticas están implementadas en [Macaulay 2]. Además, en el apéndice mostramos una implementación del algoritmo 3 escrita por Enrique A. Tobis que permite correr dicho algoritmo en [CoCoA] o en Singular [GPS05].

5.3 Aplicación

Este método también fue validado al aplicarlo a un conjunto de datos simulados de la red de los genes de la polaridad de los segmentos en la mosca de la fruta *Drosophila melanogaster*, y se comparó con los resultados del algoritmo del capítulo 4.

El conjunto de datos consistía en 24 conjuntos de 7 pares de estados de transición, cada uno generado al aplicar la función F descripta en el cuadro 4.2 a 24 inicializaciones S_0, tomadas de [L-S]. Luego, cada conjunto de datos está formado por pares

$$(S_0, F(S_0))$$

$$(F(S_i), F(S_{i+1})), 1 \leq i \leq 5$$

donde $S_{i+1} = F(S_i)$, $i = 0, \ldots, 6$.

Se representan seis condiciones experimentales diferentes: $WT = F$ y $KO_i = F^{(i)} := (F_1, \ldots, F_{i-1}, 0, F_{i+1}, \ldots, F_n)$, para $i = 2, 4, 6, 8, 12$. La condición WT representa la información del tipo silvestre, es decir, las observaciones de un estado biológico en su estado natural. Y $F^{(i)}$ es el noqueado para el nodo i, que representa el haber silenciado un gen.

Para cada condición experimental, hay cuatro inicializaciones, en las cuales un número pequeño de entradas son unos, y el resto son ceros.

Se aplicó el algoritmo 3 a los datos generados y se calcularon los conjuntos minimales para cada nodo. Estos conjuntos no siempre son únicos, de hecho solo para 9 de las 15 funciones hay un único conjunto minimal. Restringimos ahora la atención a las siguientes funciones coordenadas para las cuales hay más de una opción:

$$
\begin{aligned}
F_8 &= x_{13}((x_{11}+x_{20}+x_{11}x_{20})+x_{21}+(x_{11}+x_{20}+x_{11}x_{20})x_{21})(x_4+1)\\
&\quad (x_{13}(x_{11}+1)(x_{20}+1)(x_{21}+1)+1)\\
F_9 &= x_8+x_9(x_{18}+1)(x_{19}+1)+x_8x_9(x_{18}+1)(x_{19}+1)\\
F_{10} &= (x_8+x_9(x_{18}+1)(x_{19}+1)+x_8x_9(x_{18}+1)(x_{19}+1))(x_{20}+x_{21}+x_{20}x_{21})\\
F_{11} &= x_8+x_9Y+x_8x_9Y+1+x_{20}+((x_8+x_9Y+x_8x_9Y+1)x_{20})+x_{21}+\\
&\quad +(x_8+x_9Y+x_8x_9Y+1+x_{20}+(x_8+x_9Y+x_8x_9Y+1)x_{20})x_{21}
\end{aligned}
$$

Hay 30, 19, 2 y 5 opciones de conjuntos minimales, respectivamente. En cada caso se eligieron los conjuntos que coincidían con alguna propiedad biológica básica de la red. Las funciones F_8, F_{10} y F_{11} están asociadas a bioquímicos conocidos por no regular directamente su propia síntesis, así que se eligieron aquellos conjuntos que no contenían x_8, x_{10} y x_{11}, respectivamente. Se hizo una selección similar para la reconstrucción de F_9. Esto resultó en la identificación de 19 de las 21 aristas esperadas en el grafo de dependencia generado por el soporte de $F_8, \ldots, F_{11}$ (con el soporte de una función nos referimos al conjunto de variables que aparecen en dicha función) [2]. Las dos aristas que no se descubrieron corresponden a variables que están en el soporte de F_{10}, más precisamente x_{18} y x_{19}. Inspeccionando, se encontró que la parte de F_{10} que involucraba estas dos variables es idénticamante 0 en el conjunto de datos dado. Por lo tanto, las correspondientes interacciones en la red no son identificables usando este conjunto de datos.

Para la red completa, encontraron 39 aristas, todas correctas, usando los algoritmos 3 y 5. Las cinco aristas faltantes fueron identificadas como correspondientes a elementos del ideal de los puntos de la entrada.

En la sección 4.1 vimos que el modelo consensual al que habían llegado predecía 37 de las 44 aristas, y pareciera no tener mucha desventaja con el algoritmo propuesto en este capítulo. Sin embargo, al repetir el algoritmo del capítulo 4 con otros órdenes monomiales, se puede llegar a resultados no tan acertados.

De todos modos, vale la pena destacar que en [Just] se realiza un análisis que ayuda a decidir qué órdenes monomiales conviene elegir cuando se tiene un conjunto de datos con determinadas características.

5.4 Otra forma de encontrar la estructura minimal

En [Krupa] se muestra otra forma de encontrar lo que B. Krupa denomina la *estructura minimal* y muestra también un algoritmo menos preciso que permite encontrar una solución en tiempo polinomial. Básicamente, el proceso es el mismo: en lugar de armar monomios que son productos de las variables en donde difieren las coordenadas, se arman conjuntos con los subíndices de las mismas; y en cambio de buscar la descomposición primaria minimal del ideal generado por esos monomios, se buscan conjuntos minimales que intersequen a todos los conjuntos armados anteriormente (ver sección 2.2.1).

En esta sección presentaremos la notación utilizada en [Krupa] y describiremos el algoritmo mencionado.

Definición 5.7. Una *red causal* se define como un par ordenado $\mathcal{N} = (E, \rho)$, donde $E = \{1, \ldots, N\}$ es el conjunto de elementos en el sistema complejo, tales que a cada uno se

[2] El soporte de $F_8, \ldots, F_{11}$ está bien definido, pues estas están definidas en todo k^n.

le puede asignar un valor en un conjunto discreto Λ ($|\Lambda| = \xi$), y $\rho : \Lambda^N \to \Lambda^N$ es la función de transición entre los estados del sistema. La función de transición ρ también puede considerarse como N funciones de transición independientes $(\rho_1, \ldots, \rho_N), \rho_i : \Lambda^N \to \Lambda$, tales que $\rho_i(x)$ repesenta la i-ésima coordenada de $\rho(x)$.

Definición 5.8. El elemento i *no depende* del elemento j en la red causal si la función de transición ρ_i no depende de la j-ésima coordenada (es decir, $\forall\ s_j, s'_j \in \Lambda, \rho_i(s_1, \ldots, s_j, \ldots, s_N) = \rho_i(s_1, \ldots, s'_j, \ldots, s_N)$). Si no sucede esto, se dice que el elemento i *depende* del elemento j.

Definición 5.9. Se asocia un grafo dirigido a una red causal, con N nodos y un arco que sale de un nodo i hacia un nodo j si y solo si el elemento j depende del elemento i. El grafo se denomina *estructura* de la red y los arcos, *enlaces causales*.

Definición 5.10. El conjunto de todos los elementos de los cuales depende i se denomina el *conjunto de dependencia de i*. Cualquier conjunto que contenga al conjunto de dependencia *determina i*.

Definición 5.11. Sea s_L la proyección del estado s sobre las coordenadas de L, donde $L \subseteq E$. Se define $\rho|_L : \Lambda^L \to \Lambda^N$ como $\rho|_L(s_L) = \rho(s_{\{L,0\}})$, donde $s_{\{L,0\}}$ denota el estado s con todos los elementos fuera de L fijados como 0. Análogamente se define $\rho_i|_L$.

Lema 5.12. *$\rho_i|_L(s_L) = \rho_i(s) \forall\ s \in \Lambda^N$, si y solo si L determina i.*

Demostración. Primero observemos que L determina i si y solo si $L \supseteq \{j : i \text{ depende de } j\}$. Y también, i depende de j si y solo si existen $s \in \Lambda^N$ y $s'_j \in \Lambda$ tales que $\rho_i(s_1, \ldots, s_j, \ldots, s_N) \neq \rho_i(s_1, \ldots, s'_j, \ldots, s_N)$.

Probemos ahora las dos implicaciones:

$\Leftarrow$) Supongamos que existe $s \in \Lambda^N$ tal que $\rho_i|_L(s_L) \neq \rho_i(s)$. Como $\rho_i(s_L) = \rho_i(s_{\{L,0\}})$ por definición, tiene que ser $s_{\{L,0\}} \neq s$. Sea j tal que $(s_{\{L,0\}})_j \neq s_j$ (luego $j \notin L$), entonces i depende de j, y por hipótesis $j \in L$. Y esto es una contradicción.

$\Rightarrow$) Sea j tal que existen $s \in \Lambda^N, s'_j \in \Lambda$ con $\rho_i(s_1, \ldots, s_j, \ldots, s_N) \neq \rho_i(s_1, \ldots, s'_j, \ldots, s_N)$. Queremos ver que $j \in L$. Supongamos que no. Sea $s' = (s_1, \ldots, s'_j, \ldots, s_N)$. Luego, $s_L = s'_L$ y por lo tanto $\rho_i|_L(s_L) = \rho_i|_L(s'_L)$. Pero, por hipótesis, $\rho_i|_L(s_L) = \rho_i(s)$ y $\rho_i|_L(s'_L) = \rho_i(s')$. Y aquí se llega a una contradicción.

□

Por lo tanto, una red causal está unívocamente determinada por la colección de sus funciones de transición ρ_i restringidas a sus conjuntos de dependencia L_i.

Definición 5.13. Una *observación* en una red causal es un par ordenado de estados (s, t) tales que $t = \rho(s)$. Luego, cualquier observación consiste implícitamente en N observaciones independientes (s, t_i) en las funciones de transición individuales de la red, tales que $t_i = \rho_i(s)$.

Definición 5.14. Una red causal es *consistente* con el conjunto de observaciones $\mathcal{S} = \{(s_\alpha, t_\alpha) : \alpha = 1, \ldots, M\}$ si $\rho(s_\alpha) = t_\alpha$ para todo α.

Definición 5.15. *L determina i bajo $\mathcal{S}$* si para cualesquiera dos observaciones $(s, t_i), (s', t'_i)$ en $\mathcal{S}, t_i = t'_i$ siempre que $s_L = s'_L$. Si L no tiene subconjuntos propios que también determinan i bajo $\mathcal{S}, L$ se llamará el *conjunto de dependencia de i bajo $\mathcal{S}$*.

Problema: Dado un conjunto $\mathcal{S}_i = \{(s, t_i)_\alpha\}$ de observaciones en un elemento i, encontrar el menor conjunto de dependencia de i bajo $\mathcal{S}_i$.

Según la definición 5.15, solo se puede establecer que un conjunto L *no* es un conjunto de dependencia de i en la presencia de un contraejemplo, es decir si existen dos estados en $\mathcal{S}_i$ con los mismos valores en L que se aplican a diferentes estados. Por ejemplo, si tenemos (0110,0) y (0011,1), entonces $\{1, 3\}$ no es un conjunto de dependencia. Por lo tanto, la construcción del conjunto de dependencia es negativa, por eliminación.

Proposición 5.16. *Para dos estados s y t, sea $s \cap s'$ el conjunto más grande L en el cual $s_L = s'_L$. Sean dos observaciones (s, t_i) y (s', t'_i) con $t_i \neq t'_i$, entonces ni $s \cap s'$ ni cualquiera de sus subconjuntos son conjuntos de dependencia de i.*

Por lo tanto, el conjunto de observaciones $\mathcal{S}_i = \{(s, t_i)_\alpha\}$ da un conjunto de restricciones $\mathbb{C}^C = \{s \cap s' : \forall\, (s, t_i), (s', t'_i) \in \mathcal{S}_i, t_i \neq t'_i\}$ sobre los conjuntos de dependencia del elemento i, puesto que los conjuntos de dependencia de i no pueden ser subconjuntos de cualquiera de los conjuntos que pertenecen a $\mathbb{C}^C$. Si $\mathbb{C}$ contiene los complementos de los conjuntos en $\mathbb{C}^C$, entonces la condición se convierte en que el conjunto de dependencia debe tener intersección no vacía con todos los conjuntos en $\mathbb{C}$.

Teorema 5.17. *Sea $\mathbb{C}_i = \{(s \cap s')^C : \forall\, (s, t_i), (s', t'_i) \in \mathcal{S}_i$ tales que $t_i \neq t'_i\}$ el conjunto de las restricciones sobre el conjunto de dependencia del elemento i bajo el conjunto de observaciones $\mathcal{S}_i$. Luego L es un conjunto de dependencia de i bajo $\mathcal{S}_i$ si y solo si $L \cap \mathcal{X} \neq \emptyset$ para todo $\mathcal{X} \in \mathbb{C}_i$ y si L no contiene subconjuntos propios que también satisfagan esta condición.*

Demostración. Recordemos primero que un conjunto que determina i bajo $\mathcal{S}_i$ es considerado un conjunto de dependencia de i bajo $\mathcal{S}_i$ si es minimal respecto de la inclusión. Probemos ahora las dos implicaciones:

$\Rightarrow$) Sea $\mathcal{X} \in \mathbb{C}_i$. Entonces existen $(s, t_i), (s', t'_i) \in \mathcal{S}_i$ tales que $\mathcal{X} = (s \cap s')^C$ y $t_i \neq t'_i$. Si $L \cap \mathcal{X} = \emptyset$, esto quiere decir que $L \subseteq s \cap s'$ y por lo tanto $s_L = s'_L$. Pero $t_i \neq t'_i$ y esto implica que L no determina i bajo $\mathcal{S}_i$. Luego, si L determina i bajo $\mathcal{S}_i$, necesariamente $L \cap \mathcal{X} \neq \emptyset\ \forall\, \mathcal{X} \in \mathbb{C}$.

$\Leftarrow$) Sean $(s, t_i), (s', t'_i) \in \mathcal{S}_i$ tales que $t_i \neq t'_i$, entonces $s \cap s' \in \mathbb{C}_i$ y por hipótesis, $L \cap (s \cap s')^C \neq \emptyset$. Por lo tanto $L \nsubseteq s \cap s'$ y entonces $s_L \neq s'_L$. Luego, para dos observaciones tales que $s_L = s'_L$, necesariamente $t_i = t'_i$, y L determina i bajo $\mathcal{S}_i$. □

Este teorema sugiere el siguiente algoritmo de búsqueda exhaustiva que resuelve el problema de hallar una estructura minimal:

1. Para cada i hacer lo siguiente:
2. Construir $\mathbb{C}_i$ según el teorema
3. Empezar por los conjuntos más pequeños $L \subseteq E$ y hacer:
4. Si $L \cap \mathcal{X} \neq \emptyset$ para todo $\mathcal{X} \in \mathbb{C}_i$ devolver L.

El tiempo de ejecución de este algoritmo depende del tamaño del conjunto $\mathbb{C}_i$. Si $\mathcal{S}_i$ contiene M observaciones, el tamaño de $\mathbb{C}_i$ es a lo sumo M^2. Por lo tanto, el tiempo de ejecución es a lo sumo $M^2\xi^N$ (con $\xi = |\Lambda|$) para un solo elemento, y $NM^2\xi^N$ para toda la red.

Luego, la cantidad de operaciones en el peor de los casos para resolver el problema de la estructura minimal a través de la búsqueda exhaustiva es $O(NM^2\xi^N)$.

El tiempo de ejecución del algoritmo crece exponencialmente con el número de elementos y es por lo tanto imposible de llevar a cabo incluso para una red de tamaño moderado.

Para remediar este problema, a veces se puede perder un poco en la precisión de la solución y ganar en el tiempo de ejecución del algoritmo. Para esto, se puede ejecutar el siguiente algoritmo que intenta resolver el problema de la estructura minimal, aunque con menos precisión:

1. Para cada i hacer lo siguiente:
2. Construir $\mathbb{C}_i$ según el teorema
3. $V := \emptyset$
4. Mientras $\mathbb{C}_i \neq \emptyset$
5. $x \leftarrow$ elemento contenido en la mayoría de los conjuntos de $\mathbb{C}_i$
6. Remover de $\mathbb{C}_i$ todos los conjuntos que contengan a x
7. Poner x en V
8. Fin
9. Devolver V

Este algoritmo requiere $O(M^2N)$ operaciones para encontrar el candidato a conjunto de dependencia de un solo elemento (pues $|\mathbb{C}_i| \leq M^2, |E| = N$). Encontrar la estructura minimal para toda la red requiere $O(M^2N^2)$ operaciones.

Si nos referimos a sus resultados, por construcción se ve que $V \cap \mathcal{X} \neq \emptyset$ para todo $\mathcal{X} \in \mathbb{C}_i$ y además contiene un conjunto de dependencia. De hecho, si V no contiene subconjuntos propios que también satisfagan tener intersección no vacía con todos los elementos de $\mathbb{C}_i$, por el teorema 5.17, V es un conjunto de dependencia de i bajo $\mathcal{S}_i$. Si no, sea $V' \subsetneq V$ tal que $V' \cap \mathcal{X} \neq \emptyset$ para todo $\mathcal{X} \in \mathbb{C}_i$. Si no contiene subjconjuntos propios con esa condición, entonces V' es un conjunto de dependencia. Si no, continúo con este razonamiento hasta eventualmente llegar a un conjunto de la forma $\{x_j\}$ tal que $x_j \in \mathcal{X}$ para todo $\mathcal{X} \in \mathbb{C}_i$, y como éste no contiene subconjuntos propios, debe ser un conjunto de dependencia de i bajo $\mathcal{S}_i$.

Lo que puede suceder es que V contenga propiamente un conjunto de dependencia de i bajo $\mathcal{S}_i$ o que solo muestre algunos de los posibles, como mostramos en los siguientes ejemplos.

Ejemplo 5.18. Supongamos que $\mathcal{S}_i = \{((0,1,1,0),1);((1,0,1,1),0);((1,1,0,1),0);((0,0,1,1),0);((0,1,0,1),0);((0,0,1,0),0);((0,1,0,0),0)\}$
Entonces $\mathbb{C}_i = \{\{1,2,4\};\{1,3,4\};\{2,4\};\{3,4\};\{2\};\{3\}\}$
Luego, para el algoritmo exhaustivo el conjunto de dependencia es: $L = \{2,3\}$
Mientras que el último algoritmo devuelve: $V = \{4,2,3\}$
Y $L \subsetneq V$

Ejemplo 5.19. Si ahora $\mathcal{S}_i = \{((0,0,1),0);((1,1,1),1);((2,1,2),2);((3,3,1),3);((4,1,4),4)\}$
Entonces $\mathbb{C}_i = \{\{1,2\};\{1,2,3\};\{1,3\}\}$.
Para el primer algoritmo, los conjuntos de dependencia son: $\{1\};\{2,3\}$
Para el segundo algoritmo, $V = \{1\}$

Capítulo 6

Discretización y trabajo futuro

6.1 Discretización

Como los datos de la expresión de los genes son números reales, el primer paso en cualquier algoritmo de ingeniería reversa que usa modelos discretos debe ser el de discretizar dichos números reales en un conjunto finito (usualmente pequeño) de posibles estados. Para las redes Booleanas esto simplemente se reduce a la elección de un solo umbral para el nivel de expresión de cada gen, debajo del cual un gen se considera inactivo. Una desventaja obvia de la discretización binaria es que etiquetar información a valores reales de acuerdo a un criterio activo/inactivo generalmente produce la pérdida de una gran cantidad de información. Por supuesto, la forma de discretizar los datos juega un rol importante en el modelo que uno obtiene. La primera elección importante es la cantidad de estados discretos permitidos.

"Juguemos" un poco con algunas discretizaciones diferentes, y veamos qué sucede al aplicar el algoritmo 3.

Supongamos que el sistema dinámico se rige por la función

$$g : \mathbb{R}^5 \to \mathbb{R}^5, g(x_1, \ldots, x_5) = (x_1 e^{x_2 x_3} + x_4 x_5, x_2, x_3, x_5, x_4)$$

Su grafo de dependencia se muestra en la figura 6.1.

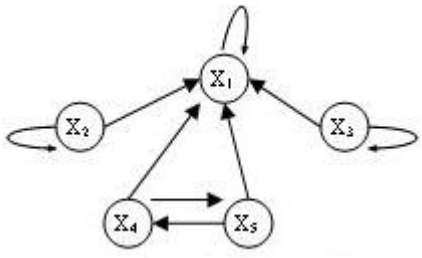


Figura 6.1: Grafo de dependencia de g

Fijemos S_0 y tomemos S_{i+1} como $g(S_i)$. Consideremos primero $S_0 = (0, 1; 0; 0; 3, 5; 4)$. Entonces tenemos la siguiente serie de tiempo:

$S_0 = (0,1;0;0;3,5;4), S_1 = (14,1;0;0;4;3,5), S_2 = (28,1;0;0;3,5;4), S_3 = (42,1;0;0;4;3,5)$

Discretizaremos en un cuerpo finito k y buscaremos $f : k^5 \to k^5$, $f = (f_1, \ldots, f_5)$ con $f_i \in k[x_1, \ldots, x_5]$ y $f(\mathbf{s}_j) = s_{j+1}$, $j = 0, 1, 2$ (y por lo tanto, $f_i(\mathbf{s}_j) = s_{ij+1}, i = 1, \ldots, 5$, $j = 0, 1, 2$).

- Discreticemos primero en $\mathbb{Z}_2$ con el siguiente criterio:
  - 0 si es menor que 1
  - 1 si es mayor o igual a 1

  Luego tenemos:

  $S_0 = (0;0;0;1;1), S_1 = (1;0;0;1;1), S_2 = (1;0;0;1;1), S_3 = (1;0;0;1;1)$

  Para f_1 tenemos los pares de transición:

  $((0;0;0;1;1),1), ((1;0;0;1;1),1)$

  Como las imágenes son las mismas, no podemos implementar el algoritmo.

  Lo mismo sucede para f_2, f_3, f_4 y f_5.

  Podríamos interpretar $f_1 \equiv f_4 \equiv f_5 \equiv 1$ y $f_2 \equiv f_3 \equiv 0$ y el grafo de dependencia no tendría aristas.

- Discreticemos ahora en $\mathbb{Z}_2$ con el siguiente criterio:
  - 0 si es menor que 20
  - 1 si es mayor o igual que 20

  Luego tenemos:

  $S_0 = (0;0;0;0;0), S_1 = (0;0;0;0;0), S_2 = (1;0;0;0;0), S_3 = (1;0;0;0;0)$

  Hay una inconsistencia: $(0;0;0;0;0)$ tiene dos imágenes.

- Discreticemos ahora en $\mathbb{F}_4$, el cuerpo de cuatro elementos $(\mathbb{Z}_2[X]/(X^2 + X + 1) = \{\bar{0}, \bar{1}, \overline{X}, \overline{X^2}\}$, identificando $0 = \bar{0}, 1 = \bar{1}, 2 = \overline{X}$ y $3 = \overline{X^2}$), con el siguiente criterio:
  - 0 si es menor que 1
  - 1 si es mayor o igual que 1 y menor que 4
  - 2 si es mayor o igual que 4 y menor que 30
  - 3 si es mayor o igual que 30

  Luego tenemos:

  $S_0 = (0;0;0;1;2), S_1 = (2;0;0;2;1), S_2 = (2;0;0;1;2), S_3 = (3;0;0;2;1)$

  Para f_1 tenemos los siguientes pares:

  $((0;0;0;1;2),2), ((2;0;0;2;1),2), ((2;0;0;1;2),3)$

  Aplicamos el algoritmo 3 "a mano", basándonos en el primer algoritmo para descomposición primaria de ideales monomiales libres de cuadrados:

$M_X = \langle x_1, x_4x_5\rangle = \langle x_1, x_4\rangle \cap \langle x_1, x_5\rangle$

f_2 y f_3 son nulos.

Para f_4 tenemos los siguientes pares:

$((0;0;0;1;2),2),((2;0;0;2;1),1),((2;0;0;1;2),2)$

Aplicamos el algoritmo 3 "a mano":

$M_X = \langle x_1x_4x_5, x_4x_5\rangle = \langle x_4x_5\rangle = \langle x_4\rangle \cap \langle x_5\rangle$

Para f_5:

$((0;0;0;1;2),1),((2;0;0;2;1),2),((2;0;0;1;2),1)$

$M_X = \langle x_1x_4x_5, x_4x_5\rangle = \langle x_4x_5\rangle = \langle x_4\rangle \cap \langle x_5\rangle$

Los posibles grafos de dependencia se ven en la figura 6.2

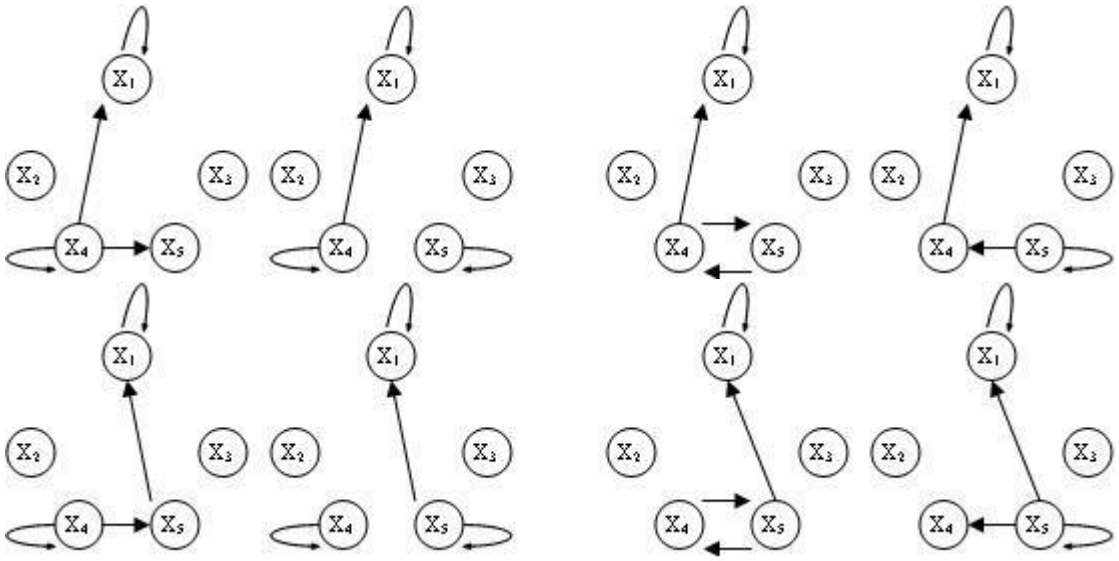


Figura 6.2: Posibles grafos de dependencia para la primera discretización en $\mathbb{F}_4$

- Ahora discreticemos en $\mathbb{F}_4$ según el siguiente criterio:
  - 0 si es menor que 1
  - 1 si es mayor o igual que 1 y menor que 4
  - 2 si es mayor o igual que 4 y menor que 20
  - 3 si es mayor o igual que 20

  Luego tenemos:

  $S_0 = (0;0;0;1;2), S_1 = (2;0;0;2;1), S_2 = (3;0;0;1;2), S_3 = (3;0;0;2;1)$

  Para f_1 tenemos los siguientes pares:

  $((0;0;0;1;2),2),((2;0;0;2;1),3),((3;0;0;1;2),3)$

Aplicamos el algoritmo 3 "a mano":

$M_X = \langle x_1 x_4 x_5 \rangle = \langle x_1 \rangle \cap \langle x_4 \rangle \cap \langle x_5 \rangle$

f_2 y f_3 son nulos.

Para f_4 tenemos:

$((0;0;0;1;2),2),((2;0;0;2;1),1),((3;0;0;1;2),2)$

$M_X = \langle x_1 x_4 x_5 \rangle = \langle x_1 \rangle \cap \langle x_4 \rangle \cap \langle x_5 \rangle$

Para f_5:

$((0;0;0;1;2),1),((2;0;0;2;1),2),((3;0;0;1;2),1)$

$M_X = \langle x_1 x_4 x_5 \rangle = \langle x_1 \rangle \cap \langle x_4 \rangle \cap \langle x_5 \rangle$

Los posibles grafos de dependencia se ven en las figuras 6.3 y 6.4

Si ahora empezamos con $S_0 = (1;2;0,001;5,39;200)$, entonces:
$S_0 = (1;2;0,001;5,39;200), S_1 = (1079,002\ldots;2;0,001;200;5,39)$,
$S_2 = (2159,1622\ldots;2;0,001;5,39;200), S_3 = (3241,4848\ldots;2;0,001;200;5,39)$
Aquí podemos observar que los valores de $S_{i1}\,(i = 1,2,3)$ que se obtienen (al medir, por ejemplo) pueden no ser exactos.

- Discreticemos en $\mathbb{F}_4$ con el siguiente criterio:
  - 0 si es menor que 1
  - 1 si es mayor o igual que 1 y menor que 4
  - 2 si es mayor o igual que 4 y menor que 30
  - 3 si es mayor o igual que 30

  Luego, tenemos:

  $S_0 = (1;1;0;2;3), S_1 = (3;1;0;3;2), S_2 = (3;1;0;2;3); S_3 = (3;1;0;3;2)$

  Para f_1 tenemos los siguientes pares:

  $((1;1;0;2;3),3);((3;1;0;3;2),3);((3;1;0;2;3),3)$

  Y no podemos aplicar el algoritmo pues todos tienen la misma imagen.

- Discreticemos ahora en $\mathbb{F}_4$ con el siguiente criterio:
  - 0 si es menor que 1
  - 1 si es mayor o igual que 1 y menor que 100
  - 2 si es mayor o igual que 100 y menor que 2000
  - 3 si es mayor o igual que 2000

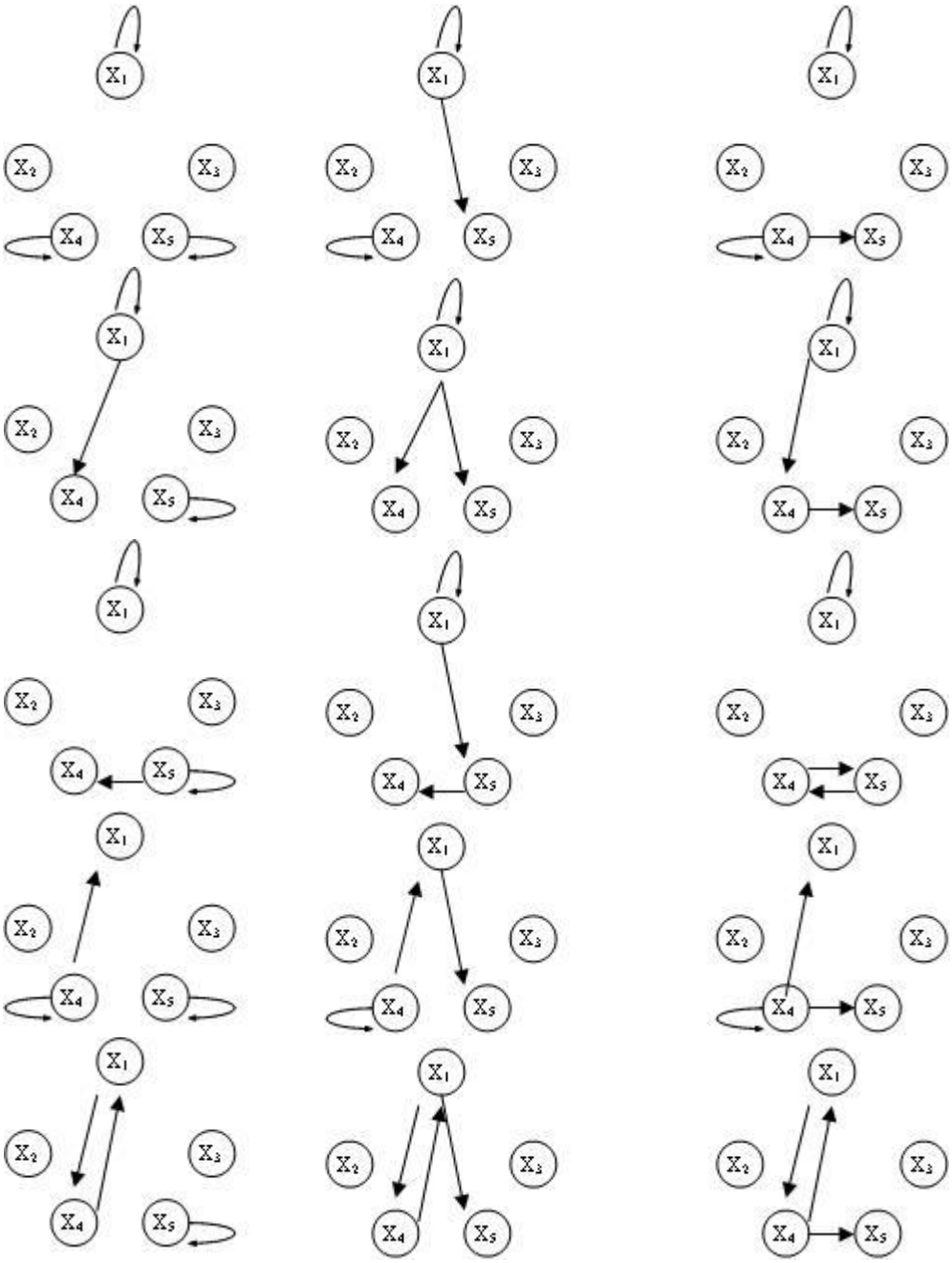


Figura 6.3: Posibles grafos de dependencia para la segunda discretización en $\mathbb{F}_4$

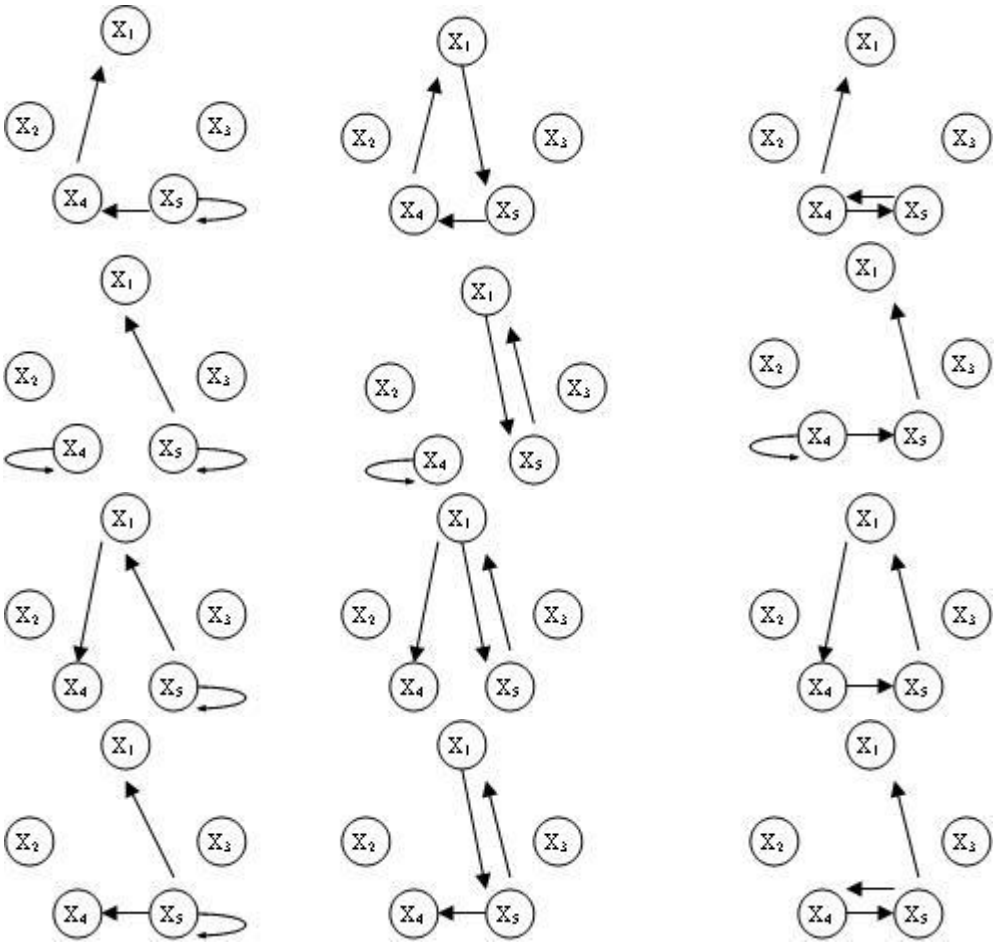


Figura 6.4: Más posibles grafos de dependencia para la segunda discretización en $\mathbb{F}_4$

Luego tenemos:

$S_0 = (1;1;0;1;2), S_1 = (2;1;0;2;1), S_2 = (3;1;0;1;2), S_3 = (3;1;0;2;1)$

Para f_1 tenemos los siguientes pares:

$((1;1;0;1;2),2),((2;1;0;2;1),3),((3;1;0;1;2),3)$

$M_X = \langle x_1x_4x_5, x_1\rangle = \langle x_1\rangle$

f_2 y f_3 son nulas.

Para f_4 tenemos:

$((1;1;0;1;2),2),((2;1;0;2;1),1),((3;1;0;1;2),2)$

$M_X = \langle x_1x_4x_5\rangle = \langle x_1\rangle \cap \langle x_4\rangle \cap \langle x_5\rangle$

Y análogo para f_5.

Los posibles grafos de dependencia se muestran en la figura 6.5

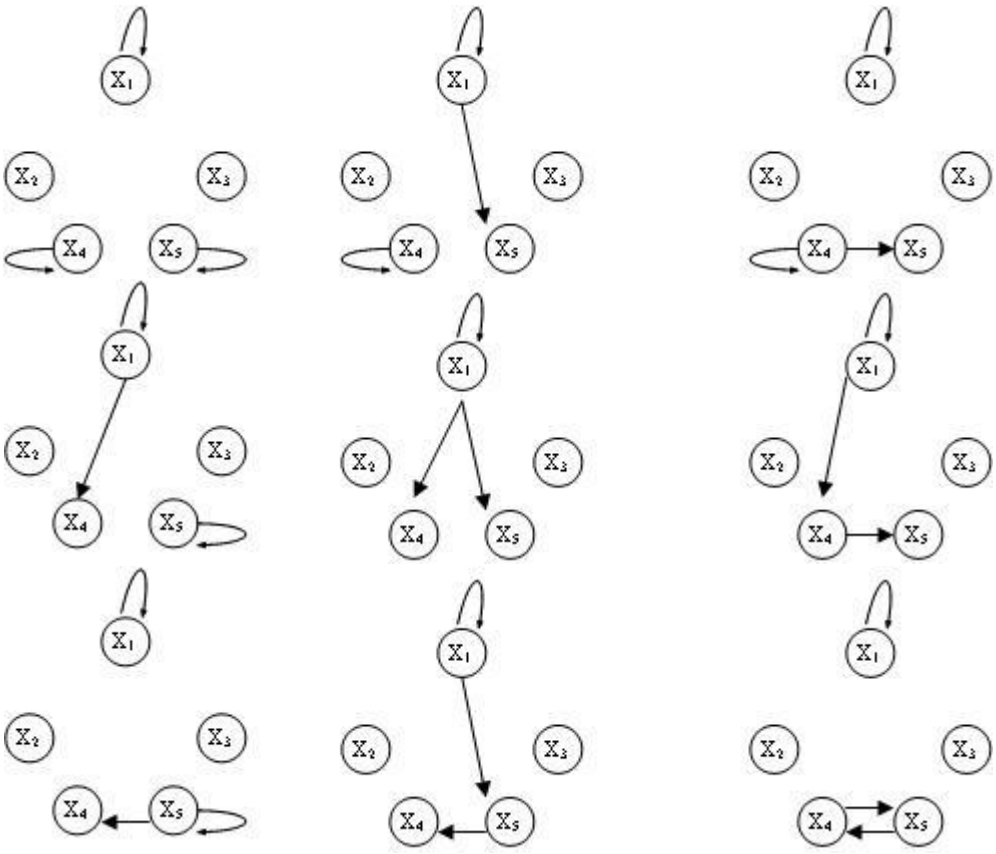


Figura 6.5: Posibles grafos de dependencia para la discretización en $\mathbb{F}_4$ de la segunda serie

En estos ejemplos pudimos ver que si $S_{ji} = c$ constante para $j \geq 1$, se pierden todas las aristas que terminan en x_i. Y si $S_{ji} = c$ constante para $j \geq 0$, también se pierden todas las aristas que comienzan en x_i, pues el algoritmo no podrá detectar diferencias entre las

i-ésimas coordenadas de los puntos. Esta situación se tiene, por ejemplo, cuando $f_i = x_i$, y sin embargo en este caso debería haber una arista que comience y termine en x_i.

En [D-MG-L] presentan un método para la discretización de información experimental en un número finito de estados. Este método fue diseñado específicamente para la discretización de cursos temporales polivalentes, como los utilizados para la construcción de modelos discretos de redes bioquímicas creados a partir de cursos temporales de información experimental. Una característica importante de dichos cursos temporales, que tuvieron presente al construir el método de discretización, es la relativamente pequeña cantidad de puntos de información. Utilizan un método de agrupamiento basado en teoría de grafos para realizar la discretización, y una técnica basada en teoría de la información para minimizar la pérdida de la información. Para los casos en los que la información discretizada se debe acomodar a un sistema dinámico en tiempo discreto, desarrollaron un algoritmo que garantiza eliminar cualquier inconsistencia causada por el proceso de discretización (es decir, si al discretizar dos vectores distintos éstos quedan iguales, procuran que sus imágenes también tengan la misma discretización).

Este método presenta algunas desventajas para los algoritmos descriptos en los capítulos 4 y 5, por lo que solo haremos una pequeña descripción.

Definición 6.1. Una *discretización* de un vector $\mathbf{v} = (v_1, \ldots, v_N)$ a valores reales es un vector $\mathbf{d} = (d_1, \ldots, d_N)$ a valores discretos, con las siguientes propiedades:

1. Cada elemento de $\mathbf{d}$ está en el conjunto $\{0, 1, \ldots, D-1\}$ para un (usualmente pequeño) entero positivo D, llamado el *grado* de la discretización.

2. Para todo $1 \leq i, j \leq N$, tenemos $d_i \leq d_j$ si $v_i \leq v_j$.

En [D-MG-L] se considera la discretización de expansión (*spanning*) de grado D, que es un caso particular definida como aquella discretización que verifica la propiedad adicional de que el menor elemento de $\mathbf{d}$ es igual a 0 y el mayor es igual a $D-1$. Además se asume que para cada entero a con $0 \leq a \leq D-1$, hay una entrada d_i de $\mathbf{d}$ tal que $a = d_i$. Es decir, si se ordenaran, las entradas de $\mathbf{d}$ son enteros consecutivos.

Si hay más de un vector a discretizar, el método discretiza cada vector por separado. Así que supongamos un vector con m entradas distintas. Se construye un grafo con pesos de m vértices, donde cada vértice representa una entrada y el peso de cada arista es la distancia euclídea entre sus extremos. El proceso de discretización comienza eliminando la(s) arista(s) con mayor peso hasta que el grafo se desconecta. Si hay más de una arista con el mayor peso actual, entonces todas las aristas con ese peso se eliminan. El orden en el cual las aristas son eliminadas lleva a componentes, en las cuales la distancia entre dos vértices cualesquiera es menor que la distancia entre dos componentes cualesquiera. Se define la distancia entre dos componentes G y H como $min\{|g-h| : g \in G, h \in H\}$. La salida del algoritmo es una discretización del vector, en la cual cada agrupación (cluster) corresponde a un estado discreto y las entradas del vector que corresponden a una misma componente se discretizan en el mismo estado.

En los ejemplos de los capítulos anteriores se necesita que todos los vectores del conjunto de datos se discreticen en la misma cantidad de estados. La manera de abordar este asunto en [D-MG-L] es primero discretizando cada vector por separado. Supongamos que para N

vectores, el método discretizó a cada uno en $m_1, m_2, \ldots, m_N$ estados, respectivamente. Sea $m = max\{m_i : i = 1, \ldots, N\}$.

Supongamos que un vector fue discretizado en m_i estados, $m_i < m$. Como el algoritmo de discretización dio m_i agrupaciones, los $m - m_i$ restantes se pueden construir ordenando las entradas en cada agrupación y partiendo aquel que contiene las dos entradas más distantes con respecto a la distancia Euclídea. Y esto se repite hasta obtener m entradas.

Este abordaje tiene un posible problema. De hecho, si un vector se discretiza en 4 estados y el número total de entradas distintas del vector es 5, luego m=6 no se puede alcanzar. En este caso, m debería tomarse como 5, juntando algunos estados donde sea necesario. En general, esto puede no ser deseable debido a que resulta en pérdida de información. Simpre es preferible incrementar la cantidad de estados, a menos que esto sea imposible, como en el ejemplo de arriba.

6.2 Errores en la medición

Los datos que se obtienen de procesos experimentales generalmente tienen errores introducidos por las deficiencias en la metodología o las imperfecciones en la instrumentación, como es el caso de los datos de microarreglos. Cuando se utiliza un método que requiere discretizar la información, se espera que en el proceso de discretización se suavice un poco el ruido producido al medir.

Sobre este tema también se trabajó en [D-MG-L].

6.3 Trabajo futuro

Al analizar los métodos propuestos en el capítulo 5, se nota que la discretización solo juega un papel importante al momento de diferenciar las coordenadas de los puntos de entrada. La teoría utilizada se basa en resultados sobre el anillo $k[x_1, \ldots, x_n]$, con k cuerpo, pero realmente no es necesario que k sea finito. Salvo quizás para garantizar que las funciones de transición sean efectivamente polinomios.

Por eso surgen dos interrogantes:

1. Si solo buscamos las relaciones causales, ¿es necesario que nos restrinjamos al anillo de polinomios $k[x_1, \ldots, x_n]$ para buscar las funciones de transición?

2. En los algoritmos vistos se parte de una determinada discretización. ¿Cómo afecta esta discretización a los resultados? ¿Cómo se pueden utilizar distintas discretizaciones para finalmente predecir el modelo correcto?

Para responder esta última pregunta, se propuso la siguiente teoría:

Definición 6.2. Una función $\mathcal{D} : \mathbb{R} \to \mathcal{C}$, con $\mathcal{C}$ un conjunto finito (no necesariamente un cuerpo), se llamará una ***función de discretización***.

Dado

$$A = \{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m)\} \tag{6.1}$$

con $\mathbf{s}_i \in \mathbb{R}^n, t_i \in \mathbb{R}$, decimos que $\mathcal{D}$ es ***consistente*** con A si

$$\mathcal{D}(\mathbf{s}_i) := (\mathcal{D}(s_{i1}), \ldots, \mathcal{D}(s_{in})) = \mathcal{D}(\mathbf{s}_j) \Rightarrow \mathcal{D}(t_i) = \mathcal{D}(t_j)\ \forall i, j$$

Un gran problema que surge es: Dado A, elegir $\mathcal{C}$ y $\mathcal{D}$ tal que $\mathcal{D}$ sea consistente con A. En particular, cuál es el menor cardinal de $\mathcal{C}$ para que esto sea posible. Esto es algo que habría que analizar.

Con respecto a la primera pregunta, pudimos ver en la sección 3.3.4 que siempre existe un polinomio interpolador que le asigna a cada $\mathbf{s}_i, t_i$. Y esta es la propiedad de $k[x_1, \ldots, x_n]$ más útil a nuestros propósitos.

Supongamos dados A, $\mathcal{C}$ y $\mathcal{D} : \mathbb{R} \to \mathcal{C}$ consistente con A. Sea

$$\mathcal{D}(A) = \{(\mathcal{D}(\mathbf{s}_1), \mathcal{D}(t_1)), \ldots, (\mathcal{D}(\mathbf{s}_m), \mathcal{D}(t_m))\} \subseteq \mathcal{C}^{n+1}$$

Sea entonces $\mathcal{F}_n \subseteq \mathcal{C}^{\mathcal{C}^n}$ tal que para todo $\mathcal{D}(A)$ finito existe $f \in \mathcal{F}_n$ tal que $f(\mathcal{D}(\mathbf{s}_i)) = \mathcal{D}(t_i),\ \forall i = 1, \ldots, m$.

Redefinimos ahora lo visto en el capítulo 5:

Definimos que $f : \mathcal{C}^n \to \mathcal{C}$ ***no depende*** de la coordenada j como en 5.8: si $\forall (c_1, \ldots, c_n) \in \mathcal{C}^n$ y $c'_j \in \mathcal{C}, f(c_1, \ldots, c_j, \ldots, c_n) = f(c_1, \ldots, c'_j, \ldots, c_n)$. Si esto no sucede, decimos que f ***depende*** de j.

$Y = \{f \in \mathcal{F}_n : f(\mathcal{D}(\mathbf{s}_i)) = \mathcal{D}(t_i)\, \forall\, i = 1, \ldots, m\}$

$M \subseteq k[x_1, \ldots, x_n]$ el ideal monomial libre de cuadrados generado por

$$W = \{m(\mathbf{s}_i, \mathbf{s}_j) : \mathcal{D}(t_i) \neq \mathcal{D}(t_j)\}$$

donde

$$m(\mathbf{s}_i, \mathbf{s}_j) = \prod_{\{k : \mathcal{D}(s_{ik}) \neq \mathcal{D}(s_{jk})\}} x_k$$

O quizás simplemente se pueda multiplicar sobre el conjunto de índices caracterizados por cierta propiedad que conozcamos termine produciendo que la asignación sea distinta.

Y formulamos el siguiente teorema:

Teorema 6.3. *$F = \{i_1, \ldots, i_l\} \subseteq \{1, \ldots, n\}$ es minimal tal que existe $f \in Y$ que depende sólo de F si y solo si $\langle x_{i_1}, \ldots, x_{i_l} \rangle$ es un primo minimal de M*

Adaptemos la proposición 5.1:

Proposición 6.4. *Existe $f \in Y$ que depende sólo de $\{i_1, \ldots, i_l\}$ si y solo si $M \subseteq \langle x_{i_1}, \ldots, x_{i_l} \rangle$*

Demostración. Probemos las dos implicaciones:

$\Rightarrow$) Sean $(\mathbf{s}_u, t_u), (\mathbf{s}_v, t_v)$ tales que $\mathcal{D}(t_u) \neq \mathcal{D}(t_v)$, y sea $f \in Y$ que depende sólo de $\{i_1, \ldots, i_l\}$. Luego, existe $j, 1 \leq j \leq l$, tal que $\mathcal{D}(\mathbf{s}_u)$ y $\mathcal{D}(\mathbf{s}_v)$ difieren en la i_j-ésima coordenada. Entonces $m(\mathbf{s}_u, \mathbf{s}_v)$ tiene a x_{i_j} como factor. Y por lo tanto, $m(\mathbf{s}_u, \mathbf{s}_v) \in \langle x_{i_1}, \ldots, x_{i_l} \rangle$.

$\Leftarrow$) Sea $h \in Y$. Sea $\phi : \pi_{i_1} \times \cdots \times \pi_{i_l} : \mathcal{C}^n \to \mathcal{C}^l$ la proyección a las $i_1, \ldots, i_l$-ésimas coordenadas. Como probamos en la proposición 5.1, si $\phi(\mathcal{D}(\mathbf{s}_i)) = \phi(\mathcal{D}(\mathbf{s}_j))$ esto implica que $h(\mathcal{D}(\mathbf{s}_i)) = h(\mathcal{D}(\mathbf{s}_j))$. Entonces sea $g \in \mathcal{F}_l$ tal que $g(\phi(\mathcal{D}(\mathbf{s}_i))) = \mathcal{D}(t_i)$ para $i = 1, \ldots, m$, g está bien definida y sólo depende de $\{i_1, \ldots, i_l\}$. Por lo tanto, puedo tomar $f \in Y$ que dependa sólo de $\{i_1, \ldots, i_l\}$, $f(\mathbf{p}) := g(\phi(\mathbf{p})), \forall\, \mathbf{p} \in \mathcal{C}^n$.

□

Observación 6.5. Para poder afirmar, al final de la demostración, que $f = g \circ \phi \in \mathcal{F}_n$, habría que pedir en la definición de $\mathcal{F}_n$ que este conjunto de funciones sea en algún sentido "cerrado" al componer con proyecciones (es decir, si $g \in \mathcal{F}_l$ y $\phi : \mathcal{C}^n \to \mathcal{C}^l$ proyección, entonces $g \circ \phi \in \mathcal{F}_n$, para todo l, n con $l \leq n$) . El conjunto de las funciones polinomiales en n indeterminadas a coeficientes en un cuerpo k satisface esta condición.

De aquí, con un razonamiento muy similar al que se usó en el capítulo 5, se llega a probar el teorema 6.3.

Seguiremos trabajando por este camino, viendo cómo discretizar de la mejor manera posible y analizando conjuntos de funciones que cumplan con la definición de $\mathcal{F}_n$ y puedan ser útiles para resolver problemas de ingeniería reversa de redes reguladoras de genes.

Bibliografía

[A-K-R] J. Abbott, M. Kreuzer, L. Robbiano. Computing Zero-Dimensional Schemes. Disponible en `http://cocoa.dima.unige.it/research/publications.html`, 2000

[A-M] M.F. Atiyah, I.G. Macdonald. Introduction to commutative algebra. Series in Mathematics. Addison-Wesley, 1969

[A-O] R. Albert, H.G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster. J. Theor. Biol. **223** 1-18*, 2003

[C-L-O] D. Cox, J. Little, D. O'Shea. Ideals, Varieties, and Algorithms. Undergraduate Texts in Mathematics, Springer-Verlag, 1992.

[CoCoA] CoCoA: a system for doing Computations in Commutative Algebra, disponible en: `http://cocoa.dima.unige.it`.

[D-MG-L] E.S. Dimitrova, J.J. McGee, R.C. Laubenbacher. Discretization of Time Course Data. arXiv:q-bio.OT/0505028, 2005

[D-P] J. Deegan, E. Packel. A new index for simple n-person games. *Int. J. Game Theory, 7:113-123*, 1978.

[Eriksson] Nicholas Eriksson. Biology and systems of polynomial equations. Disponible en `http://stanford.edu/~nke/slides.html`.

[GPS05] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2005) `http://www.singular.uni-kl.de`.

[H-S] S. Hoşten, G.G. Smith. Monomial Ideals. *Computations in algebraic geometry with Macaulay 2, Algorithms Comput. Math., vol. 8, Springer, Berlin pp.73-100*, 2002

[J-L-S-S] A.S. Jarrah, R. Laubenbacher, B. Stigler, M. Stillman. Reverse-Engineering of Polynomial Dynamical Systems. *Advances in Applied Mathematics (in press)*, 2006

[J-S] W. Just, B.Stigler. Computing Gröbner Basis of Ideals on Few Points in High Dimensions. arXiv:math.AC/0606189, 2006

[Just] W. Just. Reverse engineering discrete dynamical systems from data sets with random input vectors. *Journal of ComputationalBiology, Vol. 13, No. 8 : 1435 -1456* , 2006

[Krupa] B. Krupa. On the Number of Experiments Required to Find the Causal Structure of Complex Systems. *Journal of Theoretical Biology **219**, 257-267*, 2002

[L-S] R. Laubenbacher, B. Stigler. A computational Algebra Approach to the Reverse Engineering of Gene Regulatory Networks. *Journal of Theoretical Biology **229**, 523-537*, 2004

[Macaulay 2] D. Grayson, M. Stillman. Macaulay 2, a software system for reaserch in algebraic geometry. Disponible en `http://www.math.uiuc.edu/Macaulay2/`

[Miller] E. Miller. Alexander Duality for Monomial Ideals and Their Resolutions. arXiv:math.AG/9812095, 1998

[Milowski] R.A. Milowski. Computing irredundant irreducible decompositions of large scale monomial ideals, *ISAAC, pp. 235-242*, 2004

[Robbiano] L. Robbiano. Gröbner Bases and Statistics. *Gröbner bases and applications(Linz, 1998), London Math. Soc. Lecture Note Ser., vol. 251, Cambridge University Press, Cabridge, pp.: 179-189*, 1998

[Stigler] B. S. Stigler. An Algebraic Approach to Reverse Engineerirg with an Application to Biochemical Networks. *Virginia Tech Digital Library and Archives. PhD Dissertation*, 2005

Apéndice A

Programa

A continuación mostramos el programa creado por Enrique A. Tobis para correr el algoritmo 3 en CoCoA ([CoCoA]) o en Sigular ([GPS05]). Este algoritmo fue aplicado utilizando los datos de los ejemplos 5.5, 5.18 y 5.19, y dio los mismos resultados que se obtuvieron en las secciones correspondientes. Para ejemplificar, al final mostramos cómo se corrió utilizando los datos del ejemplo 5.19.

```
#include <fstream>
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <cctype>
#include <list>

enum Programa {cocoa,singular};
struct Opciones {
  Programa programa;
  std::string cuerpo;
  int n;
};


void leerEntrada(std::istream & input,Opciones & opciones,
 std::vector<std::vector<int> > & matriz)
{
  // Primero leemos el programa que hay que usar
  std::string programa,cuerpo;
  int n,m;
  input >> programa >> cuerpo >> n >> m;
  opciones.cuerpo = cuerpo;
  std::transform(programa.begin(),programa.end(),programa.begin(),toupper);
  if (programa == "COCOA") {
```

```
    opciones.programa = cocoa;
  } else if (programa == "SINGULAR") {
    opciones.programa = singular;
  }
  opciones.n = n;
  matriz.resize(m);
  for (int i = 0;i < m;i++) {
    matriz[i].resize(n+1);
    for (int j = 0;j < n+1;j++) {
      input >> matriz[i][j];
    }
  }
}

void generarMonomios(std::list<std::list<int> > & monomios,
     std::vector<std::vector<int> > & matriz)
{
  for (int i = 0;i < matriz.size();i++)
    for (int j = i+1;j < matriz.size();j++) {
      if (matriz[i][matriz[i].size() - 1] != matriz[j][matriz[j].size() - 1]) {
std::list<int> monomio;
for (int k = 0;k < matriz[i].size() - 1;k++) {
  if (matriz[i][k] != matriz[j][k])
    monomio.push_back(k);
}
monomios.push_back(monomio);
      }
    }
}

void generarSalida(std::ostream & output,
   const Opciones & opciones,
   const std::list<std::list<int> > & monomios)
{
  std::list<std::list<int> >::const_iterator it;
  switch(opciones.programa) {
  case singular:
    output << "ring r = " << opciones.cuerpo << ",(";
    for (int i = 0;i < opciones.n - 1;i++)
      output << "x" << i << ",";
    if (opciones.n > 0)
      output << "x" << opciones.n - 1 << "),dp;" << std::endl;
    output << "ideal i = ";
    for (it = monomios.begin();it != monomios.end();it++) {
      std::list<int>::const_iterator jt = it->begin();
```

```
      while (jt != it->end()) {
output << "x" << *jt;
jt++;
if (jt == it->end()) {
  it++;
  if (it != monomios.end())
    output << ",";
  it--;
} else {
  output << "*";
}
      }
    }
    output << ";" << std::endl;
    output << "LIB \"primdec.lib\";" << std::endl;
    output << "list l = primdecGTZ(i);list l2;" << std::endl;
    output << "for (int indiceQueNoVanAUsar = 1;"
   << "indiceQueNoVanAUsar <= size(l);indiceQueNoVanAUsar++)"
   << "{\n l2[indiceQueNoVanAUsar] = l[indiceQueNoVanAUsar][1];\n"
   << "};l2;"
   << std::endl;
    break;
  case cocoa:
    output << "Use R::= " << opciones.cuerpo << "[";
      output << "x[0.." << opciones.n-1 << "]];\n";
    output << "PrimaryDecomposition(Ideal(";
    for (it = monomios.begin();it != monomios.end();it++) {
      std::list<int>::const_iterator jt = it->begin();
      while (jt != it->end()) {
output << "x[" << *jt << "]";
jt++;
if (jt == it->end()) {
  it++;
  if (it != monomios.end())
    output << ",";
  it--;
} else {
  output << "*";
}
      }
    }
    output << "));" << std::endl;
    break;
  }
}
```

```
int main(int argc,char * argv[])
{
  std::istream * inputPtr = & std::cin; // Por default usamos la consola
  std::ostream * outputPtr = & std::cout;
  std::ifstream inputFile;
  std::ofstream outputFile;
  switch (argc) {
  case 3: // archivo de salida
    outputFile.open(argv[2]);
    if (!outputFile.good()) {
      std::cerr << "Hubo lío con el archivo de salida" << std::endl;
      exit(-1);
    } else
      outputPtr = & outputFile;
  case 2: // capacidad y archivo de entrada
   inputFile.open(argv[1]);
    if (!inputFile.good()) {
      std::cerr << "Hubo lío con el archivo de entrada" << std::endl;
      exit(-1);
    } else
      inputPtr = & inputFile;
  case 1:
    break;
  default:
    std::cout << "Por favor, revise los parametros que me pasó. Yo espero"
      << std::endl
      << "ConvertirInput "
      << "ARCHIVO_ENTRADA ARCHIVO_SALIDA"
      << std::endl;
  }
  std::ostream & output = *outputPtr;
  std::istream & input = *inputPtr;

  Opciones opciones;
  std::list<std::list<int> > monomios;
  std::vector<std::vector<int> > matriz;
  leerEntrada(input,opciones, matriz);
  generarMonomios(monomios,matriz);
  generarSalida(output,opciones,monomios);
  return 0;
}
```

Para correr el ejemplo 5.19 en CoCoA ([CoCoA]), primero creamos un archivo con la entrada:

```
cocoa
Q  %cuerpo sobre el cual se trabaja para hacer la descomposición
3 5 % n  m con n=cantidad de variables y m=cantidad de datos
0 0 1 0  %s t_i
1 1 1 1
2 1 2 2
3 3 1 3
4 1 4 4
```

Luego hacemos correr el programa en CoCoA ([CoCoA]) y nos devuelve:

```
[Ideal(x[0]), Ideal(x[1],x[2])]
```

Es decir, los conjuntos minimales son $\{1\}$ y $\{2,3\}$. Y estos son los mismos que habíamos calculado anteriormente.

Para correrlo en Singular ([GPS05]), el archivo con la entrada debe ser:

```
singular
0  %característica del cuerpo sobre el cual se trabaja para hacer la descomposición
3 5 % n  m con n=cantidad de variables y m=cantidad de datos
0 0 1 0  %s t_i
1 1 1 1
2 1 2 2
3 3 1 3
4 1 4 4
```

Luego hacemos correr el programa en Singular ([GPS05]) y nos devuelve:

```
[1]:
_[1]=x0
[2]:
_[1]=x2
_[2]=x1
```

Que se debe interpretar de la siguiente manera: el primer ideal tiene un solo generador, y es la primera variable; y el segundo ideal tiene dos generadores, que son la tercera y la segunda variables. Y, por lo tanto, nos da el mismo resultado.