UNIVERSIDAD DE SONORA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES

Programa de Posgrado en Matemáticas

Construcción del complejo simplicial de Čech generalizado y su filtración

T E S I S

Que para obtener el grado académico de:

Maestra en Ciencias Matemáticas

Presenta:

Beatriz Ramonetti Valencia

Directores de tesis:
Dr. Jesús Francisco Espinoza Fierro
Dr. Rafael Roberto Ramos Figueroa

Hermosillo, Sonora, México Septiembre de 2018

Sinodales

Dr. Jesús Francisco Espinoza Fierro
Departamento de Matemáticas,
Universidad de Sonora

Dr. Julio César Ávila Romero
Instituto Tecnológico de Estudios Superiores de Monterrey,
Campus Sonora Norte

Dr. Saúl Díaz Infante Velasco
Departamento de Matemáticas,
Cátedra CONACYT-UNISON

Dr. Rafael Roberto Ramos Figueroa
Departamento de Matemáticas,
Universidad de Sonora

Eustaquio, Martina y Orlo,
por y para ustedes.

Agradecimientos

Gracias a Dios por permitirme llegar a donde estoy hoy y por la familia que me has brindado, a la cual le agradezco infinitamente pues es lo más importante que tengo, por darme su apoyo incondicional, su comprensión y todo su amor en las buenas y en las malas, espero que se sientan orgullosos de mi. Má, pá, Goldish, sin ustedes ésto no hubiera sido posible, los AMO.

Quiero agradecerle también a mis directores, al Dr. Jesús Francisco Espinoza Fierro y al Dr. Rafael Roberto Ramos Figueroa, por sus enseñanzas, su paciencia y por haberme dedicado su tiempo, siempre con amabilidad y disposición. Es un gran honor trabajar por segunda vez con ustedes. De igual manera, quiero agradecer al Dr. Julio César Ávila Romero y al Dr. Saúl Díaz Infante Velasco, por el tiempo, apoyo y comentarios brindados para la corrección de esta tesis.

Quiero agradecer a todas las personas que me han apoyado a lo largo de estos últimos años, por su compañía y su amistad, no alcanzan las palabras para decir lo que significan para mi y lo feliz que soy de tenerlos en mi vida. Gracias por estar conmigo en todo momento.

Por último y no menos importante, le doy las gracias a mis profesores que durante la maestría me ayudaron a crecer personal y profesionalmente. En especial a las personas que conocí durante este tiempo que me enseñaron que viajar abre la mente y el corazón.

Índice general

Introducción

Dentro de las matemáticas, y otras ciencias, una de de las áreas de investigación que ha tomado bastante fuerza durante la última década es el análisis topológico de datos (ATD), la cual se dedica al análisis de conjuntos de datos utilizando técnicas de topología algebraica y fue inicialmente motivada por el estudio de la *forma* de los datos.

Además de utilizar herramientas de la topología algebraica, el ATD combina herramientas de otras áreas de las matemáticas para permitir un estudio matemáticamente riguroso de la estructura topológica de un conjunto de datos. Una manera de analizar la forma de un conjunto de datos es construir un complejo simplicial abstracto filtrado y así poder calcular los grupos de homología persistente de éste.

La información codificada en la homología persistente es comúnmente representada en forma gráfica, ya sea en un código de barras o en un diagrama de persistencia, e ilustra aquellas características topológicas más significativas en el contexto del problema a resolver.

Una herramienta muy utilizada en el ATD es la teoría de Morse discreta, desarrollada por Robin Forman en 1998 (ver [9]) y busca encontrar un método más eficiente para calcular la homología simplicial de un complejo simplicial abstracto. Para lo cual se necesita dar la estructura de complejo al conjunto de datos proveniente de nuestro problema, y las estructuras más conocidas son *Vietoris-Rips* y *Čech*.

En el 2010, Afra Zomorodian presentó un algoritmo capaz de generar de manera eficiente una estructura de Vietoris-Rips a partir de un conjunto finito de puntos en $\mathbb{R}^d$ (ver [26]). Además, en 2015, Le et al. presentaron en [16] un algoritmo capaz de generar una estructura de Čech generalizada también a partir de un conjunto finito de puntos en $\mathbb{R}^2$.

El objetivo de este trabajo es presentar un algoritmo que permite calcular la filtración de Čech generalizada de un conjunto finito de discos en $\mathbb{R}^2$. Para dicho algoritmo nos inspiramos en los dos trabajos anteriores y el texto se organizó de la siguiente manera:

El primer capítulo está destinado a exponer los dos tipos más comúnes de estructuras que se pueden dar a un conjunto finito de puntos, sus diferencias, ventajas y desventajas. Como mencionamos, las más comúnes son Vietoris-Rips y Čech. En este capítulo es donde se expone el algoritmo de Le et al. (ver [16]).

Después, en el segundo capítulo, introducimos la teoría de *sistemas de discos* y es donde exponemos el funcionamiento de nuestro algoritmo. La clave de nuestro algoritmo, y de la filtración de Čech generalizada, es encontrar la *escala de Čech* de los simplejos,

es decir, en qué momento *nace* el simplejo. Una aplicación de este valor es utilizarlo para encontrar el radio de la bola minimal (problema clásico para estructuras de Čech estándar).

Cada nivel de la filtración se define por cada escala de Čech encontrada y, de esta manera, todo se resume a resolver cómo encontrar estos valores. En este capítulo definimos una *función de peso* encargada de encontrar dichos valores y también mostramos el tiempo promedio que el algoritmo de la función de peso tarda en ejecutarse con respecto al tamaño del sistema de discos.

Desafortunadamente no es claro cómo generalizar el algoritmo para determinar la escala de Čech en un sistema de bolas en $\mathbb{R}^d$, con $d \geq 3$. Sin embargo, si es posible determinar la escala de Čech para tres bolas, la explicación se encuentra en este capítulo y todos los scripts utilizados se pueden encontrar en el Apéndice A.

Por último, en la literatura, y en nuestro tercer capítulo, se puede encontrar cómo calcular la homología simplicial de un complejo simplicial abstracto. Pero también exponemos cómo calcular la homología persistente de la filtración de Čech generalizada encontrada por nuestro algoritmo.

Estos cálculos son posibles de realizar con el software PERSEUS (ver [17]) desarrollado por Mischaikow y Nanda (ver [17]) y en este capítulo mostramos un ejemplo que explica las aplicaciones de nuestro algoritmo y del software. En el Apéndice B explicamos su funcionamiento con respecto a la teoría de Morse discreta.

Capítulo 1

Estructuras simpliciales

La topología combinatoria es una rama de la topología en la que se usan las herramientas del álgebra abstracta para estudiar los espacios topológicos a través de los complejos simpliciales y homología simplicial. Nosotros utilizaremos este tipo de complejos para estudiar la forma de un conjunto finito de puntos. También podemos estudiar los complejos simpliciales abstractos a través de sus representaciones o realizaciones geométricas, es decir con los complejos simpliciales geométricos.

En este primer capítulo presentaremos los conceptos y resultados básicos y principales para el estudio de complejos simpliciales. En las siguientes secciones expondremos definiciones y resultados necesarios para estudiar la teoría de homología, todo encontrado en [7], [9], [11] y [22]. La homología simplicial la veremos en los siguientes capítulos.

1.1. Complejos simpliciales

Como mencionamos, en esta sección hablaremos de complejos simpliciales, específicamente de definiciones y resultados sobre los complejos simpliciales abstractos y los complejos simpliciales geométricos. Estos tipos de complejos son más amigables que los complejos celulares al momento de hacer los cálculos (computacionales) de homología.

1.1.1. Complejos simpliciales abstractos

Utilizaremos los complejos simpliciales abstractos, pues existen equivalencias entre la categoría de éstos complejos y los complejos simpliciales. Para comenzar, la definición de este tipo de complejos es una descripción meramente combinatoria de la noción geométrica del complejo simplicial.

Definición 1.1.1. Sea V un conjunto finito. Un **complejo simplicial abstracto** K es una familia de subconjuntos de V, llamados *simplejos*, tal que

i) si $v \in V$, entonces $\{v\} \in K$,

ii) si $\sigma \in K$ y $\sigma' \subset \sigma$, entonces $\sigma' \in K$.

Llamaremos a V el conjunto de vértices de K, lo denotaremos por $\mathrm{Vert}(K)$.

En el siguiente ejemplo veremos varios complejos simpliciales abstractos generados por el mismo conjunto de vértices. Por lo cual, es muy importante darnos cuenta de que no tenemos definido el complejo simplicial abstracto sólo por tener su conjunto de vértices.

Ejemplo 1.1.2. Sea $V = \{1, 2, 3\}$. Entonces los siguientes son distintos complejos simpliciales abstractos, con el mismo conjunto de vértices:

a) $K_1 = 2^V = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}$.

b) $K_2 = \{\varnothing, \{1\}, \{2\}, \{1,2\}\}$.

c) $K_3 = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}\}$.

La Figura 1.1 muestra las representaciones geométricas de los complejos simpliciales anteriores:

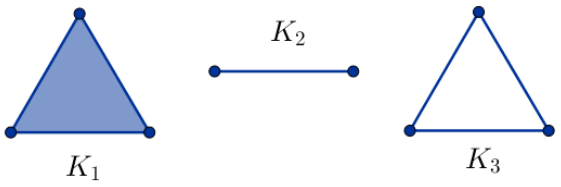


Figura 1.1: Ejemplos de complejos simpliciales abstractos.

♦

Si tenemos el conjunto finito $V = \{v_0, v_1, \ldots, v_d\}$, el conjunto 2^V es un complejo simplicial abstracto y lo llamaremos **simplejo estándar de dimensión** d, ya que equivale a tener al simplejo definido por todos los vértices. A este complejo lo denotamos por Δ^d. Si un simplejo $\sigma \in K$ tiene $d+1$ elementos, decimos que la dimensión de σ es d (o que σ es un d**-simplejo**) y lo denotamos por σ^d. Si no hay confusión con la dimensión, se puede escribir simplemente σ.

Como vimos, cada simplejo tiene su dimensión, de esta manera podemos hablar de la dimensión del complejo que los contiene. Esto da pie a la siguiente definición:

Definición 1.1.3. Si K es un complejo simplicial abstracto, entonces $\dim(K) = \max\{\dim(\sigma) \mid \sigma \in K\}$.

Si $\sigma, \tau \in K$ son tales que $\sigma \subseteq \tau$, decimos que σ es **cara** de τ (denotado por $\sigma \leq \tau$) y que τ es **supercara** de σ (denotado por $\tau \geq \sigma$). Además, si $\dim(\tau) = \dim(\sigma) + 1$, donde τ es maximal y σ es la única cara de τ, decimos que σ es una **cara libre** de τ.

Definición 1.1.4. Si $\tilde{K} \subset K$ es una subcolección que por sí misma es un complejo simplicial abstracto, entonces decimos que $\tilde{K}$ es un **subcomplejo** de K.

Al tener el complejo K, el subcomplejo que consiste de todos los l-simplejos con $l \leq d$, le llamamos el d**-esqueleto** de K y lo denotamos con $K^{(d)}$. Así, podemos hablar de contenciones de complejos.

Definición 1.1.5. Una **filtración** de un complejo simplicial K es una colección $\{K_1, K_2, \ldots, K_M\}$ de subcomplejos de K que satisface

$$\varnothing \subset K_1 \subset K_2 \subset \cdots \subset K_M = K.$$

Todo complejo simplicial abstracto de dimensión d tiene la filtración dada por los l−esqueletos con $l = 0, \ldots, d$, es decir, la filtración en K está dada por

$$\varnothing \subset K^{(0)} \subset K^{(1)} \subset \cdots \subset K^{(d)} = K.$$

Con el fin de analizar las relaciones entre complejos simpliciales abstractos introducimos un tipo de funciones que se comportan de manera adecuada con respecto a las estructuras simpliciales.

Definición 1.1.6. Si K y L son complejos simpliciales abstractos, entonces un **morfismo simplicial** $\phi : K \to L$ es una función $\phi : \mathrm{Vert}(K) \to \mathrm{Vert}(L)$ tal que para cada simplejo $\{v_0, \ldots, v_q\}$ en K, se tiene que $\{\phi(v_0), \ldots, \phi(v_q)\}$ es un simplejo en L (es posible tener repeticiones en este último).

1.1.2. Complejos simpliciales geométricos

Hasta aquí se había trabajado con los complejos y sus simplejos de modo algebraico (o combinatorio), pero ahora buscamos que la representación también sea geométrica: tanto una imagen que represente el complejo, como la construcción de complejos.

1.1.2.1. Realización geométrica

Buscamos que los complejos simpliciales tengan una realización geométrica, de esta manera se muestran ejemplos más visuales. Se ve con más claridad a la hora de utilizar filtraciones y complejos en general, lo cual se verá más adelante. Por lo pronto, vamos a introducir teoría que nos permite definir las realizaciones geométricas de los complejos.

Un subconjunto A de un espacio euclidiano es llamado **afín** si para todo par de puntos distintos $x, x' \in A$, la línea determinada por ellos está contenida en A. Además, se tiene que los espacios afines son convexos. Podemos pensar que afín es una línea y convexo es un segmento.

Estos conjuntos cumplen la propiedad de que la intersección de subconjuntos convexos (afines) de $\mathbb{R}^n$ también es convexa (afín). En efecto, si $\{X_j\}_{j\in J}$ una familia de conjuntos convexos tal que $\bigcap_{j\in J} X_j \neq \varnothing$. Tomemos $x, y \in \bigcap X_j$. Entonces $x, y \in X_j$ para cada $j \in J$, donde cada uno es convexo, por lo que $\{(1-t)x + ty\}$, con $0 \leq t \leq 1$, pertenece a X_j para cada $j \in J$. Así, el segmento $\{(1-t)x + ty\}$, con $0 \leq t \leq 1$, está en la intersección. Para el caso de subconjuntos afines, se omite la restricción $0 \leq t \leq 1$.

Una **combinación afín** de puntos $p_0, \ldots, p_m \in \mathbb{R}^n$ es una combinación lineal de la siguiente forma

$$x = t_0 p_0 + t_1 p_1 + \cdots + t_m p_m,$$

donde $\sum_{i=0}^{m} t_i = 1$. Una **combinación convexa** es una combinación afín con $t_i \geq 0$ para toda $i = 1, \ldots, m$.

Por ejemplo, sean x_0, x_1 dos puntos distintos en $\mathbb{R}^n$. Su combinación afín es

$$\{t_0x_0 + t_1x_1 \mid t_0, t_1 \in \mathbb{R} \text{ tal que } t_0 + t_1 = 1\} = \{t_0x_0 + (1 - t_0)x_1 \mid t_0 \in \mathbb{R}\},$$

que es la línea que pasa por x_0 y x_1. Mientras que la combinación convexa de los mismos puntos es:

$$\begin{aligned} \{t_0x_0 + t_1x_1 \mid t_0, t_1 \geq 0 \text{ tal que } t_0 + t_1 = 1\} &= \{t_0x_0 + (1 - t_0)x_1 \mid t_0 \geq 0, 1 - t_0 \geq 0\} \\ &= \{t_0x_0 + (1 - t_0)x_1 \mid t_0 \in [0, 1]\}, \end{aligned}$$

que es el segmento en $\mathbb{R}^n$ que une x_0 con x_1.

Teorema 1.1.7. Si $p_0, \ldots, p_m \in \mathbb{R}^n$ entonces el conjunto convexo generado por éstos, $[p_0, \ldots, p_m]$, es el conjunto de todas las combinaciones convexas de $p_0, \ldots, p_m$.

La prueba de este resultado, se puede ver en [22, p. 32]. Además, se tiene que el espacio afín generado por $p_0, \ldots, p_m \in \mathbb{R}^n$ consiste de todas las combinaciones afines de esos puntos. Entonces definimos la envolvente convexa de $p_0, \ldots, p_m \in \mathbb{R}^n$ como la intersección de todos los conjuntos convexos que los contengan.

Un conjunto ordenado de puntos $\{p_0, \ldots, p_m\} \subset \mathbb{R}^n$ es **afín independiente** si el conjunto $\{p_1 - p_0, p_2 - p_0, \ldots, p_m - p_0\}$ es un subconjunto linealmente independiente de $\mathbb{R}^n$. Por lo que cualquier subconjunto de $\mathbb{R}^n$ linealmente independiente es un conjunto afín independiente.

Por ejemplo, si tomamos $\{e_0, \ldots, e_n\}$, una base de $\mathbb{R}^n$, entonces $\{0, e_0, \ldots, e_n\}$ es afín independiente aunque no sea linealmente independiente.

En [22, p. 33], se muestra que con un conjunto ordenado de puntos $p_0, \ldots, p_m$ en $\mathbb{R}^n$, las siguientes condiciones son equivalentes:

1. $\{p_0, \ldots, p_m\}$ es afín independiente.
2. Si $\{s_0, \ldots, s_m\} \subset \mathbb{R}$ satisface $\sum_{i=0}^{m} s_ip_i = 0$, donde $\sum_{i=0}^{m} s_i = 0$, entonces $s_i = 0$ para toda $i = 0, \ldots, m$.
3. Para cada $x \in A$, el conjunto afín generado por $\{p_0, \ldots, p_m\}$, tiene una única expresión como combinación afín $x = \sum_{i=0}^{m} t_ip_i$ y $\sum_{i=0}^{m} t_i = 1$.

Decimos que un conjunto de puntos $\{a_1, \ldots, a_k\}$ en $\mathbb{R}^n$ está en **posición general** si para cada $n + 1$ de estos puntos forman un conjunto afín independiente.

Si suponemos que los elementos de $\mathrm{Vert}(K)$ son puntos que están en una posición general en algún espacio euclidiano, entonces el espacio topológico $|K|$ está definido por $\bigcup_{\sigma \in K} |\sigma|$, donde $|\sigma|$ denota la envolvente convexa de los puntos en σ. Entonces decimos que

$|K|$ es una realización geométrica de K.

En la sección anterior vimos lo que son los complejos, simplejos y relaciones entre ellos. Ahora les pondremos un orden a los vértices de dichos elementos y así poder empezar con los cálculos.

Definición 1.1.8. Sea K un complejo simplicial abstracto y $\sigma \in K$. Una **orientación** en σ es un orden total de sus vértices módulo una permutación par de ellos.

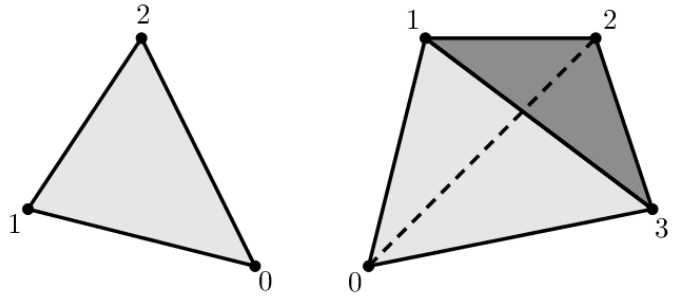


Figura 1.2: Simplejos orientados con un orden en sus vértices.

Una forma natural de orientar un simplejo es enumerar sus vértices. Si $\sigma^d = \{v_0, v_1, \ldots, v_d\}$ está orientado por el orden de sus subíndices denotaremos su orientación con $[v_0, v_1, \ldots, v_d]$. Si el simplejo tiene dimensión cero, entonces una orientación consiste en elegir un signo + o −.

Si $\sigma^d = \{v_0, v_1, \ldots, v_d\}$ está dotado con la orientación $[v_0, v_1, \ldots, v_d]$ y $\alpha_i^{d-1} = \{v_0, v_1, \ldots \hat{v_i} \ldots, v_d\}$ es la $(d-1)$-cara de σ dado por todos los vértices excepto v_i, entonces la orientación en α_i está determinada por $(-1)^i[v_0, v_1, \ldots \hat{v_i} \ldots, v_d]$. Esta orientación se llama la **orientación heredada** de σ^d.

Poniendo un poco de atención a la Figura 1.2, específicamente al 2-simplejo del lado izquierdo, podemos ver que la orientación que heredan sus aristas son $(01), (12)$ y $-(02)$.

1.1.2.2. Lema del nervio

En esta sección enunciamos el Lema del nervio y mostramos un ejemplo donde se ve la importancia de dicho lema, el cual nos da una relación entre espacios topológicos y complejos simpliciales. De este modo podremos caracterizar las propiedades topológicas de un espacio y buscar invariantes topológicos.

Sea M un espacio topológico y sea $\mathcal{U}$ una cubierta abierta finita de M. El **nervio** $N(\mathcal{U})$ de la cubierta, es el complejo simplicial abstracto donde cada $k-$simplejo corresponde a una intersección no vacía de $k+1$ elementos distintos de $\mathcal{U}$.

Lema 1.1.9. Sea $\mathcal{U}$ una cubierta abierta finita de M. Si $\mathcal{U}$ es una cubierta buena, i.e., tal que toda intersección no vacía de conjuntos de $\mathcal{U}$ es contraíble, entonces M es homotópicamente equivalente a la realización geométrica de $N(\mathcal{U})$.

La prueba de este lema no se expondrá en el texto, pero se puede encontrar en [15, p. 269]. Ahora veremos un ejemplo ilustrativo del resultado.

Ejemplo 1.1.10. Consideremos a S^1 con dos cubiertas abiertas finitas distintas $\mathcal{U}_1$ y $\mathcal{U}_2$, donde $\mathcal{U}_1 = \{U_1, U_2\}$ y $\mathcal{U}_2 = \{U_1, U_2, U_3\}$. Cada cubierta se comporta como se representa en la Figura 1.3.

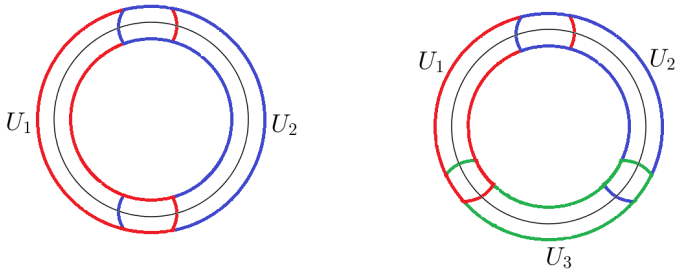


Figura 1.3: Cubiertas abiertas $\mathcal{U}_1$ (izquierda) y $\mathcal{U}_2$ (derecha).

Por definición, $\mathrm{Vert}(N(\mathcal{U}_1)) = \mathcal{U}_1$ y por las intersecciones que se ven en la Figura 1.3, el conjunto de aristas consta del único elemento $[U_1, U_2]$ y no contamos con simplejos de mayor dimensión. Decimos que $\mathcal{U}_1$ no es una cubierta buena porque la intersección $U_1 \cap U_2$ no es contraíble. Además, la representación geométrica de la arista no representa al espacio topológico S^1.

Por otro lado, para $\mathcal{U}_2$, contamos con el siguiente conjunto de aristas $\{[U_1, U_2], [U_1, U_3], [U_2, U_3]\}$ y seguimos sin tener simplejos de mayor dimensión. En este caso, llamamos a $\mathcal{U}_2$ una cubierta abierta buena porque las intersecciones no vacías entre los elementos de $\mathcal{U}_2$ son contraíbles. También tenemos que el complejo simplicial abstracto que se obtiene en el nervio es homotópicamente equivalente a S^1, como se puede observar en la Figura 1.4.

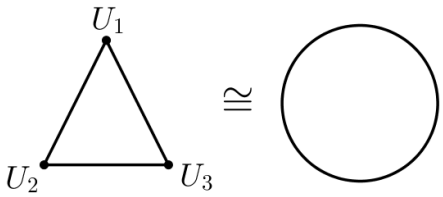


Figura 1.4:

⧫

1.1.3. Ejemplos. Los complejos de Čech y de Vietoris-Rips

En esta sección mostraremos dos métodos para construir complejos a partir de un conjunto finito de puntos en un espacio métrico, al cual denotaremos por Y. En la literatura podemos encontrar que estos métodos son los más utilizados para el estudio de la forma de dicho conjunto. Cabe mencionar que con estos complejos, las filtraciones se construyen utilizando la distancia entre los puntos del conjunto.

Definición 1.1.11. Si $N = \{c_0, c_1, \ldots, c_d\} \subset \mathbb{R}^d$ y $\epsilon \geq 0$, entonces el **complejo de Čech** de radio ϵ, denotado por $\mathscr{C}(N;\epsilon)$, es el complejo simplicial abstracto cuyos m-simplejos se corresponden con los subconjuntos $\sigma \subset N$ tales que la cardinalidad de σ es $m+1$ y que

$$\bigcap_{c_i \in \sigma} B\left(c_i;\epsilon\right) \neq \varnothing \text{ donde } B(c;r) = \left\{p \in \mathbb{R}^d \mid d(p,c) \leq r\right\}.$$

Especificamos que los complejos de Čech se definen en $\mathbb{R}^d$ ya que se utiliza la definición de *bolas cerradas*. En la Figura 1.5 podemos ver una filtración cuando N es un conjunto de 3 puntos y con diferentes valores del parámetro de proximidad ϵ.

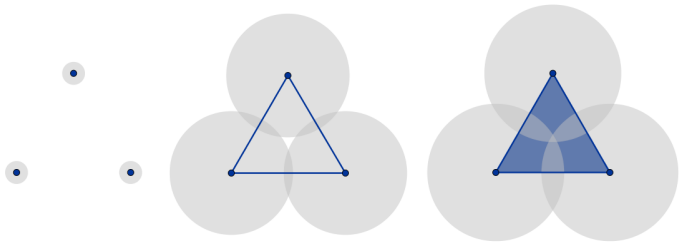


Figura 1.5: Filtración de Čech con los parámetros $\epsilon_1 = 0.5$, $\epsilon_2 = 2.7$, $\epsilon_3 = 3$.

Nótese que para ϵ suficientemente pequeño, $\mathscr{C}(N;\epsilon) = N$ y para ϵ suficientemente grande, $\mathscr{C}(N;\epsilon)$ es isomorfo al simplejo estándar $\Delta^{|N|}$. Además se cumple que si $\epsilon_1 < \epsilon_2$, entonces $\mathscr{C}(N;\epsilon_1)$ es un subcomplejo de $\mathscr{C}(N;\epsilon_2)$.

Como todo d-simplejo tiene 2^{d+1} caras, entonces el número de simplejos en el complejo de Čech crece de forma exponencial con la cardinalidad de N, por lo que se requiere de mucha memoria en una computadora para almacenarlo (para valores grandes de N y d).

Por otro lado, tenemos otra forma de generar complejos simpliciales abstractos a partir de un conjunto finito de puntos. Este método sólo depende del 1−esqueleto y por eso no se requiere tanta memoria como con el complejo de Čech.

Definición 1.1.12. Si $N = \{c_0, c_1, \ldots, c_d\} \subset Y$ y $\epsilon \geq 0$, entonces el **complejo de Vietoris-Rips** de radio ϵ, denotado por $\mathrm{VR}(N;\epsilon)$, es el complejo simplicial abstracto cuyos m-simplejos se corresponden con los subconjuntos $\sigma \subset N$ tales que la cardinalidad de σ es $m+1$ y para toda pareja $c_i, c_j \in \sigma$, $d(c_i, c_j) \leq \epsilon$.

En la Figura 1.6 podemos ver una filtración para este nuevo caso, donde N también es un conjunto de tres puntos y diferentes valores del parámetro de proximidad ϵ.

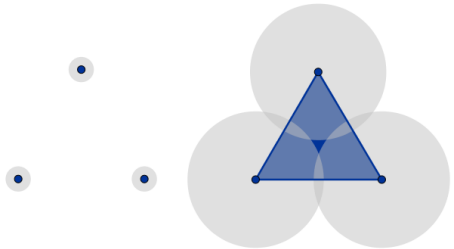


Figura 1.6: Filtración de Vietoris-Rips con los parámetros $\epsilon_1 = 0.5$ y $\epsilon_2 = 2.7$.

Nuevamente se cumple que $\mathrm{VR}(N;\epsilon) = N$ para ϵ suficientemente pequeño. Para ϵ suficientemente grande $\mathrm{VR}(N;\epsilon)$ es isomorfo al simplejo estándar $\Delta^{|N|}$, y si $\epsilon_1 < \epsilon_2$, entonces $\mathrm{VR}(N;\epsilon_1)$ es un subcomplejo de $\mathrm{VR}(N;\epsilon_2)$.

Nótese que si $\{c_0,\ldots,c_m\} \subset N$ es tal que cualquier pareja de puntos está a distancia menor que ϵ, entonces todo el m-simplejo generado por estos puntos pertence a $\mathrm{VR}(N;\epsilon)$. De las definiciones se puede verificar que para un $\epsilon > 0$ fijo, todo simplejo en $\mathscr{C}(N;\epsilon)$ es también simplejo de $\mathrm{VR}(N;\epsilon)$, por lo que existe un encaje natural de $\mathscr{C}(N;\epsilon)$ en $\mathrm{VR}(N;\epsilon)$.

1.2. Construcción del complejo de Čech generalizado

En esta sección presentaremos el algoritmo de [16], en el cual se calcula la estructura de Čech generalizada a partir de un conjunto finito de discos en el plano. Como vimos en la sección anterior, construimos los complejos con respecto a las bolas cerradas cuyos centros son los puntos del conjunto finito en cuestión. Un detalle importante del algoritmo es que los discos no tendrán el mismo radio y es una herramienta importante a la hora de trabajar con datos que tienen diferentes ponderaciones (según su contexto).

Es importante mencionar que trabajaremos en $\mathbb{R}^2$ y por tanto con *discos*. Llamaremos *discos vecinos* a los discos que se intersectan. El complejo, ya sea de Vietoris-Rips o de Čech, puede ser de dos tipos y eso depende de si los discos tienen el mismo tamaño o no. Cuando los discos tienen el mismo tamaño, el complejo es llamado *estándar* (como en la Definición 1.1.11 y la Definición 1.1.12); pero si son de diferentes tamaños, es llamado *generalizado*. En el resto de la sección se van a utilizar los complejos de Čech generalizados.

En el complejo de Čech, cada disco es representado por un vértice. El **índice** de un vértice c se define como el mayor entero k tal que para todo $i \leq k$ cada $(i-1)$–simplejo con c como elemento es cara de al menos un i–simplejo con c como elemento.

Por una parte, el índice de un vértice nos dice cuántas veces el disco correspondiente de este vértice se traslapa con sus vecinos. Por lo que un índice cero indica que la celda está separada de las demás y un índice k indica que se conecta con otros por k–simplejos.

Ejemplo 1.2.1. En la Figura 1.7 se ven tres complejos distintos, y a cada uno de ellos se les va a calcular los índices de sus vértices.

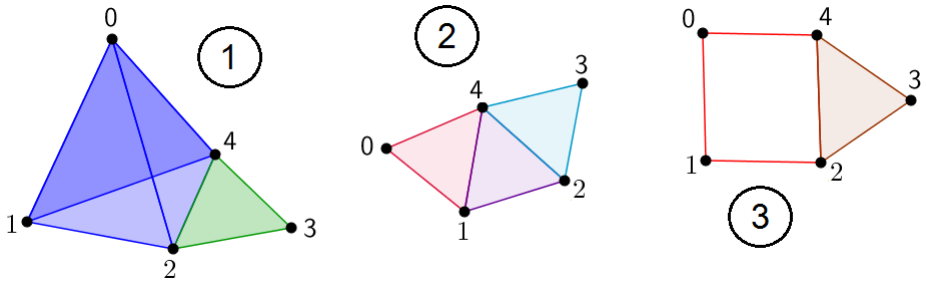


Figura 1.7: Ejemplos de realizaciones geométricas de complejos simpliciales.

1) $i_0 = i_1 = 3$, $i_2 = i_3 = i_4 = 2$.

2) $i_0 = i_1 = i_2 = i_3 = i_4 = 2$.

3) $i_0 = i_1 = i_2 = i_4 = 1$, $i_3 = 2$.

⧫

A continuación vamos a presentar el algoritmo que genera una estructura de complejo de Čech a partir de un conjunto finito de puntos $N = \{c_1, c_2, \ldots, c_n\}$. Recordemos que el sistema de Čech $\mathscr{C}(N;\epsilon)$ se define con los discos cuyos centros son los puntos de N, por lo que consideraremos lo siguiente:

- Cada 0–simplejo $\sigma = (i)$ se representa con el disco $D_i(c_i;\epsilon)$.
- Cada k–simplejo $\sigma = (0, 1, \ldots, k)$ se representa con $k+1$ discos de intersección no vacía, i.e., $\bigcap_{i=0}^{k} D_i \neq \varnothing$.
- Los conjuntos S_k se compone de los k–simplejos del complejo de Čech, i.e., $S_k = \{\sigma \in \mathscr{C}(N;\epsilon) \mid \dim(\sigma) = k\}$.

El primer pseudocódigo que presentamos es la secuencia que genera el conjunto S_1 (aristas) a partir de los discos dados (0–simplejos).

Algoritmo 1: 1-Esqueleto

Entrada: S_0.
Salida: S_1.

```
1 S_1 := ∅
2 N ← |S_0|
3 para i = 1, ..., N − 1 hacer
4     para j = i + 1, ..., N hacer
5         si los discos i y j se intersectan entonces
6             Añadir s = (i, j) a S_1
```

Ahora, veremos un ejemplo para mostrar cómo funciona el pseudocódigo con un conjunto pequeño de puntos.

Ejemplo 1.2.2. Consideremos el conjunto $N = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ de centros de los discos ubicados como se muestra en la Figura 1.8.

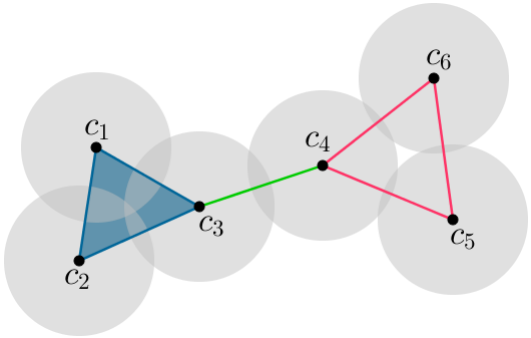


Figura 1.8: Sistema de discos y la realización geométrica del complejo de Čech.

$i = 1$	$j = 2$	$(1,2) \in S_1$	$i = 3$	$j = 4$	$(3,4) \in S_1$
	$j = 3$	$(1,3) \in S_1$		$j = 5, 6$	
	$j = 4, 5, 6$		$i = 4$	$j = 5$	$(4,5) \in S_1$
$i = 2$	$j = 3$	$(2,3) \in S_1$		$j = 6$	$(4,6) \in S_1$
	$j = 4, 5, 6$		$i = 5$	$j = 6$	$(5,6) \in S_1$

Por tanto, el resultado de ejecutar el algoritmo es el siguiente conjunto de 1−simplejos:

$$S_1 = \{(1,2), (1,3), (2,3), (3,4), (4,5), (4,6), (5,6)\}.$$

⧫

A la hora de construir k−simplejos, con $k \geq 2$, el proceso es más complicado. En el resto de la sección nos dedicaremos a los detalles para construirlos, pues se debe tener cuidado ya que las combinaciones de $k+1$ discos en todos los N discos es enorme. Entonces debemos considerar sólo a un grupo de *candidatos* que tienen la oportunidad de ser k−simplejos.

Por lo que si asumimos que $u = \{D_0, D_1, \ldots, D_k\}$ define un k−simplejo, entonces podemos deducir que cada par $\{D_i, D_j\}$, donde $0 \leq i \neq j \leq k$, son vecinos. Esto sugiere que $\hat{u} = \{\hat{D}_0, \hat{D}_1, \ldots, \hat{D}_k\}$, donde $\hat{D}_1, \ldots, \hat{D}_k$ son vecinos de $\hat{D}_0$, es candidato del disco $\hat{D}_0$ a definir un k−simplejo. Si uno de estos $\hat{D}_1, \ldots, \hat{D}_k$ no es vecino de $\hat{D}_0$, $\hat{D}$ ya no define un k−simplejo.

De esta manera, nada más tenemos que verificar que los candidatos sean k−simplejos. Si denotamos por d_{ij}, d_{ji} los dos puntos de intersección de los discos D_i, D_j, consideremos a $\mathcal{D} = \{d_{ij}, d_{ji} \mid 0 \leq i < j \leq k\}$ como el conjunto de puntos de intersección para el candidato $\hat{u}$.

Denotamos por D_* al disco de radio más pequeño de los discos $\hat{D}_0, \hat{D}_1, \ldots, \hat{D}_k$ de $\hat{u}$. Entonces tenemos los siguientes tres casos:

1. Si D_* está dentro de los otros, entonces $\hat{u}$ es un k−simplejo.

2. Si D_* no está dentro de los otros discos del candidato, pero si existe un punto de intersección $d_{ij} \in \mathcal{D}$ que está dentro de los discos D_t, donde $0 \leq t \leq k$ con $t \neq i, j$, entonces tenemos que $\hat{u}$ es un k−simplejo.

3. Si no se cumplen los casos anteriores, entonces el candidato $\hat{u}$ no es un k−simplejo.

De modo que el siguiente pseudocódigo es el encargado de verificar si un candidato es efectivamente un k−simplejo de Čech, donde $k \geq 2$. La clave para utilizar este pseudocódigo es que se debe tener previamente el conjunto de candidatos, y de él se hará un *filtro*.

Algoritmo 2: Verificación de candidatos

Entrada: El candidato $\hat{u}$.
Salida: verdadero/falso dependiendo si $\hat{u}$ es un simplejo.

D_* es el disco más pequeño de $\hat{u}$
si *D_* está dentro del disco $\hat{D}_i$ para todos los discos distintos de D_** **entonces**
    `verification` ← verdadero
en otro caso
    $\mathcal{D} := \varnothing$
    para $i = 0, \ldots, k-1$ **hacer**
        para $j = i+1, \ldots, k$ **hacer**
            si *existen d_{ij}, d_{ji} los puntos de intersección de los discos i, j* **entonces**
                Añadir $\{d_{ij}, d_{ji}\}$ a $\mathcal{D}$
            en otro caso
                `verification` ← falso
                regresar(`verification`)
    si *existe d_{ij} dentro de todos los discos $\hat{c}_t$ distintas de i, j* **entonces**
        `verification` ← verdadero
    en otro caso
        `verification` ← falso
regresar(`verification`)

Por último, mostramos el pseudocódigo principal. Este algoritmo llama a los dos que se presentaron anteriormente. Al terminar de presentar el pseudocódigo, se van a explicar sus detalles con un ejemplo.

Algoritmo 3: Simplejos de dimensión mayor o igual a 2

Entrada: S_0.
Salida: S_k para $k \geq 1$.

Cacular S_1 con Algoritmo 1
$k = 2$
mientras $k < N$ **hacer**
    $S_k := \varnothing$
    para cada $\hat{c} \in S_0$ **hacer**
        $S^* = \{\text{candidatos de } \hat{D}_0\}$
        para cada $\hat{u} \in S^*$ **hacer**
            Verificación($\hat{u}$)
            si `verification` = verdadero **entonces**
                Añade $\hat{u}$ a S_k
    si $S_k \neq \varnothing$ **entonces**
        $k = k + 1$
    en otro caso
        terminar mientras.

De esta manera, podemos obtener la estructura de Čech generalizada de un conjunto finito de puntos en $\mathbb{R}^2$. Ahora, veremos un ejemplo donde consideramos 15 puntos en $\mathbb{R}^2$ y sus radios no son iguales.

Ejemplo 1.2.3. Consideremos los siguientes puntos (x, y) en $\mathbb{R}^2$ con sus respectivos radios r (tabla de la derecha). Como se ve en la Figura 1.9, hay intersección entre algunos de los discos, con los algoritmos veremos exactamente cuáles son esas intersecciones.

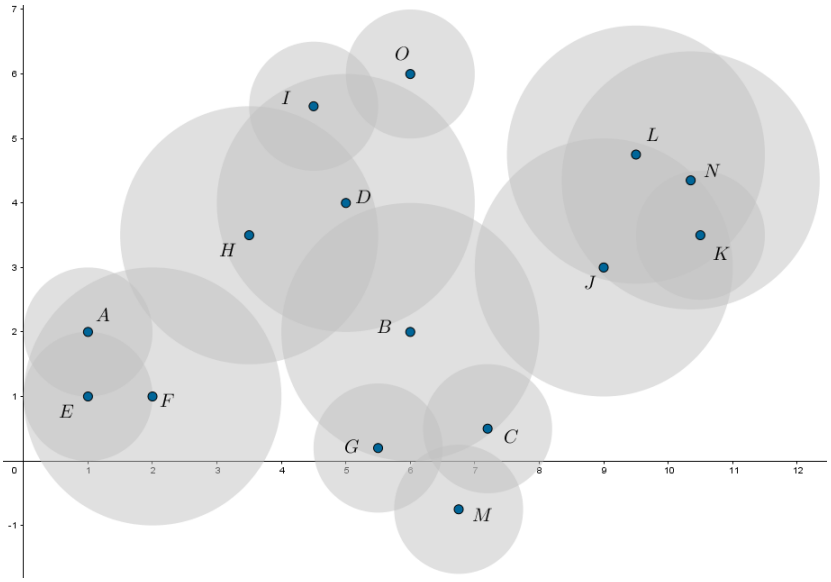


Figura 1.9: Sistema de discos en $\mathbb{R}^2$.

v	(x, y)	r
A	$(1, 2)$	1
B	$(6, 2)$	2
C	$(7.2, 0.5)$	1
D	$(5, 4)$	2
E	$(1, 1)$	1
F	$(2, 1)$	2
G	$(5.5, 0.2)$	1
H	$(3.5, 3.5)$	2
I	$(4.5, 5.5)$	1
J	$(9, 3)$	2
K	$(10.5, 3.5)$	1
L	$(9.5, 4.75)$	2
M	$(6.75, -0.75)$	1
N	$(10.35, 4.35)$	2
O	$(6, 6)$	1

Utilizando el Algoritmo 1, obtenemos S_1 como sigue:

$$\begin{array}{ccccc} (A,E) & (A,F) & (A,H) & (B,C) & (B,D) \\ (B,G) & (B,H) & (B,J) & (B,M) & (C,G) \\ (C,M) & (D,H) & (D,I) & (D,O) & (E,F) \\ (F,H) & (G,M) & (H,I) & (I,O) & (J,K) \\ (J,L) & (J,N) & (K,L) & (K,N) & (L,N) \end{array}$$

Al tener los conjuntos S_0 y S_1 (no vacíos), podemos buscar si el complejo de Čech generalizado resultante tiene simplejos de dimensión mayor. Empecemos con $k = 2$, los 2−simplejos son:

$$\begin{array}{ccccc} (A,E,F) & (A,F,H) & (B,C,G) & (B,C,M) & (B,D,H) \\ (B,G,M) & (C,G,M) & (D,H,I) & (D,I,O) & (J,K,L) \\ & (J,K,N) & (J,L,N) & (K,L,N) & \end{array}$$

Ahora veremos los simplejos de dimensión 3, i.e., 3-simplejos:

$$(B,C,G,M) \qquad (J,K,L,N)$$

Para dimensiones más grandes, también hay conjunto de candidatos S^* (no vacío), pero como no llegan a ser simplejos, entonces $S_k = \varnothing$ para $k \geq 4$. En la Figura 1.10 mostramos la estructura simplicial del sistema de discos de la Figura 1.9:

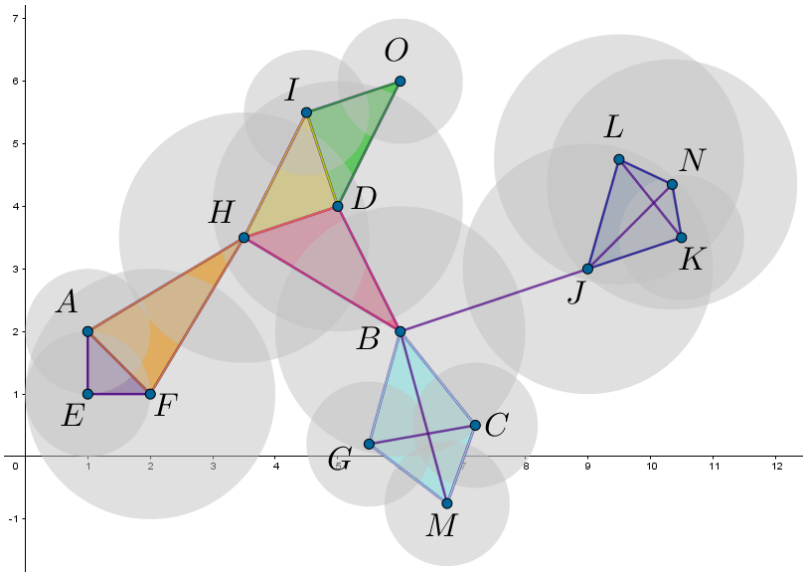


Figura 1.10: Estructura de Čech del sistema de discos.

♦

Este ejemplo se calculó gracias al algoritmo programado en R, que se presenta en los anexos. Por tanto, los detalles se expondrán ahí. Es importante hacer la aclaración de que los algoritmos presentados en este capítulo solamente generan la estructura de Čech del sistema fijo de discos, no da información sobre ninguna filtración; eso se verá más adelante.

Capítulo 2

Filtración de Čech en sistemas de discos

En este capítulo se muestra el algoritmo principal de este proyecto de tesis que fue implementado en el software R (Apéndice A). Como se mencionó anteriormente, a partir de un conjunto finito de discos en $\mathbb{R}^2$, buscamos calcular la filtración de Čech de este conjunto y hacemos uso de la estructura de Čech generalizada.

2.1. Sistemas de discos

Ahora, en lugar de referirnos a los *puntos* de nuestro *conjunto finito*, tomaremos la siguiente definición como notación para el resto del escrito. Consideremos al conjunto finito $N = \{c_1, \ldots, c_n\}$ en $\mathbb{R}^2$. Al utilizar la estructura de Čech generalizada, cada centro c_i viene con su respectivo radio r_i. Así, tenemos lo siguiente:

Definición 2.1.1. Sea $N \subset \mathbb{R}^2$ un conjunto finito, definimos el **sistema de discos** asociado a N como

$$M = \{D_i := D_i(c_i; r_i) \mid c_i \in N,\ r_i > 0\}.$$

Donde D_i son discos cerrados y denotamos por ∂D_i la frontera del disco D_i.

Las características principales que aplicamos en nuestro algoritmo, y que podemos enlistar, son:

- El algoritmo del capítulo anterior trabaja con un radio fijo para todos los centros, lo que hacemos nosotros es utilizar dicho algoritmo como una *función* y poder así crear una filtración del sistema. De esta manera, también calculamos una lista de *parámetros de proximidad*, de forma que cada nivel de la filtración es un complejo de Čech generalizado asociado al sistema de discos donde los radios son *reescalados*.

- Necesitamos determinar un límite en los parámetros y en la dimensión. Si no tomáramos un límite no hay manera de decidir hasta qué momento dejamos de aumentar los radios y la lista de simplejos.

- Otro detalle fue modificar el modo de calcular los S_k. Cada vez que se calcula un $k-$simplejo, se obtiene el *parámetro* que ocupa para ser un simplejo de Čech. Se busca que los parámetros no pasen del límite mencionado en el punto anterior.

Cada punto de la lista anterior se va a exponer y al final se explicará el algoritmo completo mediante un ejemplo que utiliza el conjunto de discos del Ejemplo 1.2.3. La intención de hacer estos cambios es poder calcular una filtración de Čech y obtener la homología persistente (por medio del software PERSEUS: [17]) de dicho conjunto.

Siendo así, para un valor fijo en el parámetro de proximidad ϵ, podemos conocer la estructura de Čech correspondiente $\mathscr{C}(N;\epsilon)$, sin embargo no conocemos los distintos valores de ϵ que definen la filtración de Čech. Por lo que también mostramos un algoritmo para calcular, dado un sistema de discos, el factor de reescalamiento para los radios que induce a una intersección no trivial. Dicho factor de reescalamiento es llamado la *escala de Čech* del sistema de discos.

Esta información permite construir una función de pesos

$$\begin{aligned} \omega : \mathscr{C}(N;\epsilon) &\to \mathbb{R} \\ \sigma &\mapsto \omega(\sigma) := \lambda^*(\sigma) \end{aligned}$$

y en consecuencia la correspondiente filtración de Čech. Obteniendo la lista de parámetros (con valores menores o iguales a ϵ), podemos considerar la siguiente definición:

Definición 2.1.2. Sea $\lambda \geq 0$ y sea M el sistema de discos asociado al conjunto N, definimos el **sistema reescalado por** λ asociado a N como

$$M_\lambda = \{D_i^\lambda := D_i\,(c_i; \lambda \cdot r_i) \mid D_i \in M\}.$$

Geométricamente, el conjunto M_λ consiste de tantos discos como M, con los mismos centros que los discos de M pero con los radios reescalados por λ. De modo que llamaremos *escala* a λ. Podemos observar que cuando $\lambda > 0$, M_λ será un sistema de discos otra vez. Tenemos dos casos inmediatos con respecto a λ, $M_1 = M$ y $M_0 = N$.

El paso principal para calcular la filtración de Čech consiste en el cálculo de la escala de Čech de cada subcolección de discos que tiene la posibilidad de formar un simplejo de Čech. La técnica para calcular dicha escala es mediante una función $\rho_M(\lambda)$ que depende continuamente del parámetro λ y, la escala de Čech de M corresponde a una raíz de ρ_M. Dicha raíz la encontramos con un método numérico.

En las siguientes secciones introducimos la escala de Vietoris-Rips, la escala de Čech de un sistema de discos y la relación que existe entre ellas (Lema 2.2.4). También una forma de cuantificar la intersección de discos, este concepto es clave para la continuidad de la función ρ_M mencionada anteriormente y caracteriza la escala de Čech, como en [16].

Un teorema importante utilizado en varias situaciones a lo largo del capítulo, es el Teorema de Helly (ver [5]):

Teorema 2.1.3. Para una colección finita de subconjuntos convexos $X_1, X_2, \ldots, X_n$ de $\mathbb{R}^d$, donde $n > d$, si la intersección de toda subcolección de $d+1$ de esos conjuntos es no vacía, entonces $\bigcap_{j=1}^{n} X_j \neq \varnothing$.

En el contexto de este capítulo, el Teorema de Helly implica que para un sistema de discos en el plano $M = \{D_i\}_{i=1}^n$, se cumple que $\bigcap_{i=1}^{n} D_i \neq \varnothing$ si y sólo si, toda terna de discos de M tiene intersección no vacía.

Demostramos en el Teorema 2.2.9 que el algoritmo para calcular las escalas de Čech es consistente y en efecto, calcula lo establecido. Además, mostramos dos aplicaciones: el cálculo del radio de la bola minimal y un algoritmo para la construcción de la estructura simplicial filtrada de Čech.

2.2. Sistemas de Vietoris-Rips y de Čech

Sea M un sistema de discos de un conjunto finito N y sean D_i, D_j discos de M tales que $D_i \cap D_j \neq \varnothing$. Definimos $D_i \sqcap D_j$ como el conjunto unitario $\{d_{ij}\}$ caracterizado como sigue:

1. Si $\partial D_i \cap \partial D_j \neq \varnothing$, entonces $d_{ij} \in \partial D_i \cap \partial D_j$ es el único punto con la propiedad de $\langle d_{ij} - c_i, \mathbf{n}_{ij} \rangle \geq 0$, y $\mathbf{n}_{ij} = (-b, a)$ es el vector normal a $c_j - c_i = (a, b)$,
2. Si $\partial D_i \cap \partial D_j = \varnothing$, define d_{ij} como el único punto de intersección en $\partial D_i^\lambda \cap \partial D_j^\lambda$, para λ dado como la mínima escala tal que $D_i^\lambda \subset D_j^\lambda$, i.e., $\lambda = \dfrac{\|c_i - c_j\|}{|r_i - r_j|}$.

De hecho, cuando D_i y D_j son concéntricos, tenemos que $D_i \sqcap D_j = \{c_i\} = \{c_j\}$. Por otro lado, si los discos D_i y D_j son interna o externamente tangentes, entonces $d_{ij} = d_{ji}$. Podemos pensar a d_{ij} como el punto de intersección de las fronteras (cuando $\partial D_i \cap \partial D_j$ es no vacía) que está a la izquierda del vector de c_i a c_j. La Figura 2.1 muestra la construcción mencionada:

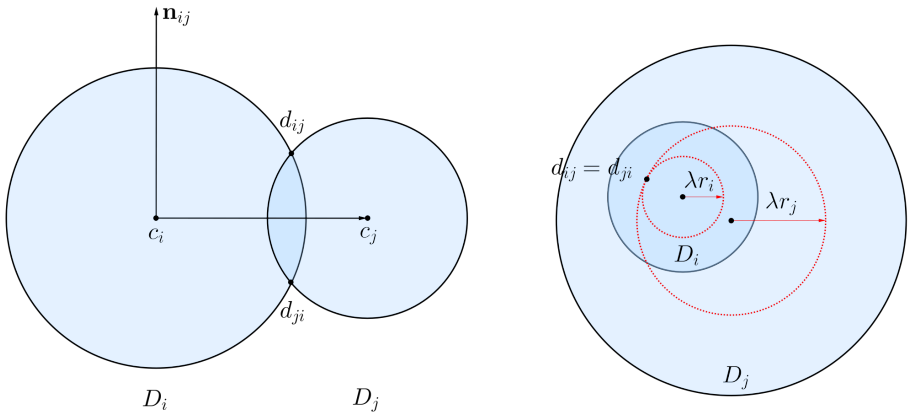


Figura 2.1: Discos con intersección no vacía.

Definición 2.2.1. Sea $M = \{D_1, D_2, \ldots, D_m\}$ un sistema de discos. Llamaremos a M un **sistema de Vietoris-Rips** si $D_i \cap D_j \neq \varnothing$ para cada par $i, j \in \{1, 2, \ldots, m\}$. La **escala de Vietoris-Rips** se define como

$$\lambda_M := \text{ínf}\{\lambda \in \mathbb{R} \mid M_\lambda \text{ es un sistema de Vietoris-Rips}\}.$$

Más aún, si el sistema de discos M tiene la propiedad de intersección no vacía $\bigcap_{D_i \in M} D_i \neq \varnothing$ entonces M es llamado un **sistema de Čech**. La **escala de Čech** se define como

$$\lambda^* := \text{ínf}\{\lambda \in \mathbb{R} \mid M_\lambda \text{ es un sistema de Čech}\}.$$

Se sigue que el sistema de discos M es un sistema de Vietoris-Rips si y sólo si $\lambda_M \leq 1$. En particular, $\lambda_{M_{\lambda_M}} = 1$. En la Figura 2.2 se muestra un sistema de discos con la propiedad de intersección no vacía, i.e., M es un sistema tanto de Vietoris-Rips como de Čech.

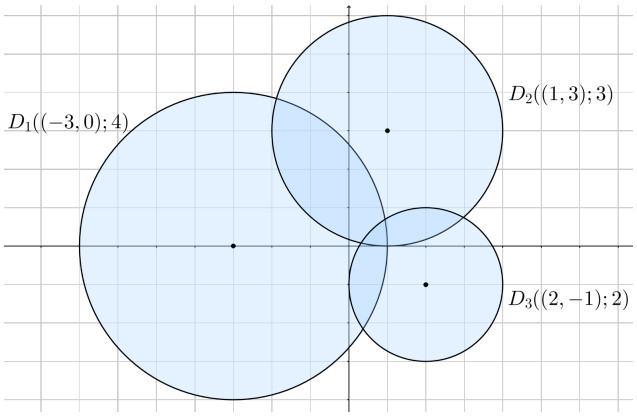


Figura 2.2: $M = \{D_1((-3,4);4), D_2((1,3);3), D_3((2,-1);2)\}$.

No es difícil calcular la escala de Vietoris-Rips λ_M para un sistema de discos $M = \{D_1, D_2, \ldots, D_m\}$. Si r_i es el radio de D_i y $\|c_i - c_j\|$ es la distancia entre los centros de D_i y D_j, entonces

$$\lambda_M = \max_{i<j} \left\{ \frac{\|c_i - c_j\|}{r_i + r_j} \right\}.$$

Por ejemplo, para el sistema de discos M de la Figura 2.2, se tiene que $\lambda_M = \dfrac{\sqrt{26}}{6} \approx 0.8498366$. De esta manera, comprobamos que M era un sistema de Vietoris-Rips originalmente.

Por otro lado, un sistema de discos M no siempre tiene intersección no vacía en la escala λ_M, i.e., $\bigcap_{D_i \in M} D_i^{\lambda_M} \neq \varnothing$. No es razonable esperar esta propiedad, especialmente porque el sistema de Vietoris-Rips M_{λ_M} tiene la intersección mínima por pares.

La Figura 2.3 muestra el sistema de Vietoris-Rips reescalado M_{λ_M}:

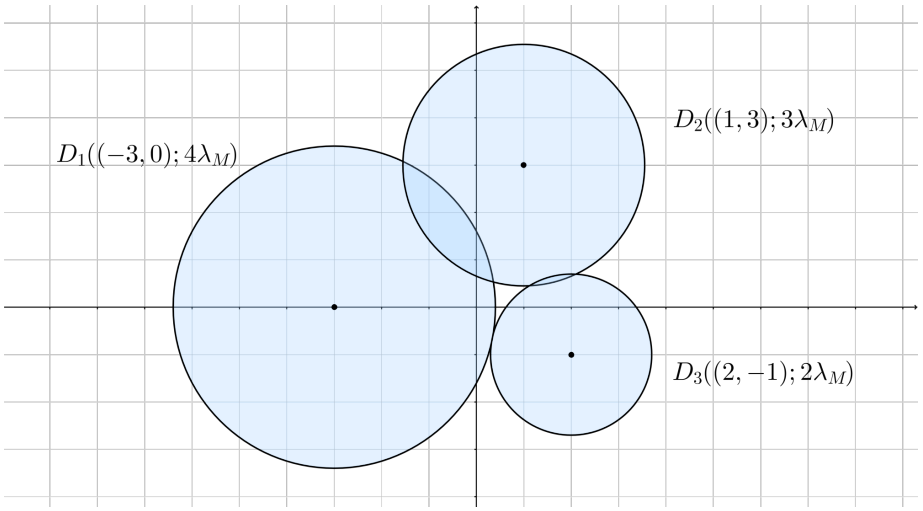


Figura 2.3: $M_{\lambda_M} = \left\{D_1^{\lambda_M}, D_2^{\lambda_M}, D_3^{\lambda_M}\right\}$.

Es importante darnos cuenta que si λ^* es la escala de Čech de un sistema M, entonces el sistema M_{λ^*} tiene un único punto de intersección

$$c_M = \bigcap_{D_i \in M} D_i^{\lambda^*}.$$

Porque si suponemos que existen $p_1, p_2 \in \bigcap_{D_i \in M} D_i^{\lambda^*}$ diferentes, y como los discos son compactos, tenemos que el punto medio $\hat{p} = \frac{1}{2}(p_1 + p_2)$ también está en todos los discos $D_i^{\lambda^*}$. Además, $\|\hat{p} - c_i\| < \max\{\|p_1 - c_i\|, \|p_2 - c_i\|\}$ para todos los centros c_i del sistema M.

Sea $\lambda_i < \lambda^*$ una escala tal que $\hat{p} \in \bigcap_{D_i \in M} D_i^{\lambda_i}$. Se sigue que $\hat{\lambda} = \max\{\lambda_i\} < \lambda^*$ y $\hat{p} \in \bigcap_{D_i \in M} D_i^{\hat{\lambda}}$, lo que contradice la minimalidad de la escala de Čech λ^*.

Para estudiar los sistemas de Čech, damos la siguiente caracterización de acuerdo a los puntos de intersección d_{ij}.

Lema 2.2.2. Sea $M = \{D_1, D_2, \ldots, D_m\}$ un sistema de discos. Entonces M es un sistema de Čech si y sólo si, existe $d_{ij} \in D_i \sqcap D_j$ tal que $d_{ij} \in D_k$ para toda $k \neq i, j$.

Demostración. Definimos $A = \bigcap_{1 \leq i \leq m} D_i \neq \varnothing$. Entonces A tiene únicamente una de las siguientes geometrías:

1. $A = \{c^*\}$,
2. A es la región acotada por más de un arco de circunferencia,
3. $A = D_i$ para algún $i \in \{1, \ldots, m\}$.

En el primer caso, necesariamente se tiene que $c^* \in \partial D_i \cap \partial D_j$ para algunos i, j y se sigue que $d_{ij} = c^* \in D_k$ para toda k. En el segundo caso, si $a \in \partial A \subset A$, está en la intersección de dos arcos, digamos ∂D_i y ∂D_j, entonces $a = d_{ij}$ y satisface que $d_{ij} \in D_k$ para toda $k \neq i, j$.
Para el último caso, si $A = D_i$ para algún i, entonces para cada $j \neq i$ se tiene que $d_{ij} \in A \subset D_k$ para toda $k \neq i, j$. El recíproco es cierto también. ■

Este criterio fue presentado en [16] para un sistema de discos en el cual uno de los discos está dentro de los otros discos, o cuando todos los discos tienen intersección no vacía en sus fronteras. Nuestra definición de intersección $D_i \sqcap D_j$ para todos los discos tal que $D_i \cap D_j \neq \varnothing$, incluso para $\partial D_i \cap \partial D_j = \varnothing$, nos permite extender el criterio para todas las posibles configuraciones de sistemas de discos.

El siguiente lema es una consecuencia inmediata de la definición de la escala de Čech, y nos da otra caracterización de los sistemas de Čech.

Lema 2.2.3. Sea M un sistema de discos y sea λ^* su escala de Čech. Entonces, $\bigcap_{D_i \in M} D_i \neq \varnothing$ si y sólo si $\lambda^* \leq 1$.

2.2.1. La escala de Čech

En esta sección presentamos un algoritmo para calcular la escala mínima λ^* en la cual el sistema de discos M tiene intersección no vacía, i.e., M_{λ^*} es un sistema de Čech.

En el caso de tener radios iguales $r_i = r$, el Lema de Vietoris-Rips (ver [6]) nos dice que para cada sistema de Vietoris-Rips M, el sistema reescalado por $\sqrt{4/3}$ es un sistema de Čech. De hecho, el lema es verdad para un sistema de Vietoris-Rips en cualquier dimensión, i.e., dado $M = \{D_i(c_i; r) \subset \mathbb{R}^d \mid r \in \mathbb{R}^+\}$ tal que $D_i \cap D_j \neq \varnothing$ para todo i, j, entonces $\bigcap_{D_i \in M} D_i^{\sqrt{2d/(d+1)}} \neq \varnothing$.

En el siguiente lema extendemos el Lema de Vietoris-Rips para cualquier sistema $\{D(c_i; r_i) \subset \mathbb{R}^2 \mid i = 1, \ldots, m\}$ de Vietoris-Rips en el plano. Gracias al Teorema de Helly (Teorema 2.1.3), solamente nos fijaremos en las ternas del sistema de discos.

Lema 2.2.4. Sea $M = \{D(c_1; r_1), D(c_2; r_2), D(c_3; r_3)\}$ un sistema de Vietoris-Rips. Entonces el sistema de discos $M_{\sqrt{4/3}}$ es un sistema de Čech, i.e., $\bigcap_{i=1}^{3} D_i^{\sqrt{4/3}} \neq \varnothing$.

Demostración. El resultado es trivial si M es un sistema de Čech, entonces sea $\lambda > \lambda_M$ la escala de Čech del sistema de discos M. Demostraremos que $\lambda^2 \leq \dfrac{4}{3}$.

Sea $\{c^*\} = \bigcap_i D(c_i; \lambda r_i)$. Del hecho de que M no es un sistema de Čech, se sigue que $\|c_i - c^*\| = \lambda r_i$.

Demostremos primero que c^* pertenece a la envolvente convexa del conjunto $\{c_1, c_2, c_3\}$. Procedemos suponiendo lo opuesto, a saber, que existe un vector v tal que $\langle v, c_i - c^* \rangle > 0$

para cada $i = 1, 2, 3$. Se sigue que

$$\begin{aligned}
\|c_i - c^*\|^2 &= \|c_i - (c^* + tv) + tv\|^2 \\
&= \|c_i - (c^* + tv)\|^2 + 2\langle c_i - (c^* + tv), tv\rangle + \|tv\|^2 \\
&= \|c_i - (c^* + tv)\|^2 + 2\langle c_i - c^*, tv\rangle - 2\langle tv, tv\rangle + \langle tv, tv\rangle \\
&= \|c_i - (c^* + tv)\|^2 + 2t\langle c_i - c^*, v\rangle - \|tv\|^2 \\
&= \|c_i - (c^* + tv)\|^2 + 2t\langle c_i - c^*, v\rangle - t^2\|v\|^2 \\
&> \|c_i - (c^* + tv)\|^2
\end{aligned}$$

para $t \in I_v := \left(0, 2\langle v, c_i - c^*\rangle / \|v\|^2\right)$. Lo cual implica que $c^* + tv \in D(c_i; \lambda r_i)$ para $i = 1, 2, 3$ y $t \in I_v$. Esto contradice el hecho de que $\{c^*\} = \bigcap_i D(c_i; \lambda r_i)$. Por lo tanto, c^* pertenece a la envolvente convexa.

Definimos $\hat{c}_i := c_i - c^*$ y sea θ_{ij} el ángulo formado por los vectores $\hat{c}_i$ y $\hat{c}_j$. De la relación $\theta_{12} + \theta_{13} + \theta_{23} = 2\pi$, podemos suponer sin pérdida de generalidad que $\pi > \theta_{12} \geq \dfrac{2\pi}{3}$ y, por tanto $-1 < \cos(\theta_{12}) \leq -\dfrac{1}{2}$. Así,

$$\sqrt{2(1 + \cos(\theta_{12}))} \leq 1. \tag{2.1}$$

Luego, de (2.1) y de la desigualdad media geométrica-media aritmética, tenemos la siguiente desigualdad:

$$\sqrt{2(1 + \cos(\theta_{12}))} \cdot \sqrt{r_1 r_2} \leq \sqrt{r_1 r_2} \leq \frac{r_1 + r_2}{2}. \tag{2.2}$$

A partir de la desigualdad (2.2), podemos observar lo siguiente:

$$\begin{aligned}
\sqrt{2(1 + \cos(\theta_{12}))} \cdot \sqrt{r_1 r_2} &\leq \frac{r_1 + r_2}{2} \\
2\sqrt{2r_1 r_2(1 + \cos(\theta_{12}))} &\leq r_1 + r_2 \\
8r_1 r_2(1 + \cos(\theta_{12})) &\leq (r_1 + r_2)^2 \\
8r_1 r_2 - 2r_1 r_2 &\leq r_1^2 + r_2^2 - 8r_1 r_2 \cos(\theta_{12}) \\
3(2r_1 r_2) &\leq r_1^2 + r_2^2 - 8r_1 r_2 \cos(\theta_{12}) \\
3(r_1^2 + 2r_1 r_2 + r_2^2) &\leq 4r_1^2 + 4r_2^2 - 8r_1 r_2 \cos(\theta_{12}) \\
3(r_1 + r_2)^2 &\leq 4(r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta_{12}))
\end{aligned}$$

Equivalentemente,

$$\begin{aligned}
3\lambda^2(r_1 + r_2)^2 &\leq 4(\lambda^2 r_1^2 + \lambda^2 r_2^2 - 2\lambda^2 r_1 r_2 \cos(\theta_{12})) \\
&= 4(\|\hat{c}_1\|^2 + \|\hat{c}_2\|^2 - 2\|\hat{c}_1\|\|\hat{c}_2\| \cos(\theta_{12})) \\
&= 4(\|\hat{c}_1\|^2 + \|\hat{c}_2\|^2 - 2\langle \hat{c_1}, \hat{c_2}\rangle) \\
&= 4\|\hat{c}_1 - \hat{c}_2\|^2
\end{aligned}$$

Así,

$$\lambda^2 \leq \frac{4}{3} \cdot \frac{\|\hat{c}_1 - \hat{c}_2\|^2}{(r_1 + r_2)^2} = \frac{4}{3} \cdot \frac{\|c_1 - c_2\|^2}{(r_1 + r_2)^2} \leq \frac{4}{3}.$$

Por tanto, se probó lo que se quería. ■

Más aún, demostramos en [8] que si M es un sistema de Vietoris-Rips en $\mathbb{R}^d$, entonces $M_{\sqrt{2d/(1+d)}}$ es un sistema de Čech. En [6] se tiene que ésto es cierto para el caso de

sistemas con radios iguales.

Para un sistema de discos particular M, se puede aplicar el lema anterior usando solamente la escala de Vietoris-Rips. Siendo así, tenemos el siguiente corolario:

Corolario 2.2.5. Si M es un sistema de discos y λ_M es su escala de Vietoris-Rips, entonces su escala de Čech λ^* satisface que $\lambda^* \in [\lambda_M, \sqrt{4/3}\lambda_M]$. En particular, si $\sqrt{4/3}\lambda_M \leq 1$ entonces M_{λ_M} es un sistema de Čech. También se tiene que $M_{\sqrt{4/3}\lambda_M}$ es siempre un sistema de Čech.

Vamos a definir la función que nos permitirá calcular las escalas de los subsistemas de discos, recordemos que es una función continua que nos indica la escala de Čech de un sistema dado.

Definición 2.2.6. Sea $M = \{D_1, D_2, \ldots, D_m\}$ un sistema de Vietoris-Rips, con $m \geq 3$. Se define la función $\rho(M) := \max\limits_{1\leq i,j\leq m}\left\{\min\limits_{k\neq i,j}\{r_k - \|d_{ij} - c_k\|\}\right\}$. Si λ_M es la escala de Vietoris-Rips de M, definimos

$$\begin{aligned} \rho_M : [\lambda_M, \infty) &\to \mathbb{R} \\ \lambda &\mapsto \rho_M(\lambda) = \rho(M_\lambda) \end{aligned}$$

Se sigue por construcción que ρ_M es continua en todo el intervalo $[\lambda_M, \infty)$. En efecto, dados tres discos D_i, D_j, D_k denotemos por $\Lambda^k_{i,j}$ la función

$$\lambda \mapsto \lambda r_k - \|d_{ij}(\lambda) - c_k\|,$$

donde $d_{ij}(\lambda)$ es el elemento en $D_i^\lambda \sqcap D_j^\lambda$. Si $r_i \neq r_j$, entonces la función $\Lambda^k_{i,j}$ es continua en el intervalo $\left[\dfrac{\|c_i - c_j\|}{r_i + r_j}, \dfrac{\|c_i - c_j\|}{|r_i - r_j|}\right]$ y varía linealmente en el intervalo $\left[\dfrac{\|c_i - c_j\|}{|r_i - r_j|}, \infty\right)$, pues a partir de $\lambda = \dfrac{\|c_i - c_j\|}{|r_i - r_j|}$ el término $\|d_{ij}(\lambda) - c_k\|$ permanece constante.

La función ρ_M juega un papel fundamental en el resto del trabajo. El siguiente pseudocódigo está diseñado para evaluarse en un sistema de Vietoris-Rips M y una escala λ.

Algoritmo 4: Rho (ρ).

Entrada: M, λ.
Salida: $\rho(M_\lambda)$.

```
1  rho ← −∞
2  para i = 1, ..., m − 1 hacer
3      para j = i + 1, ..., m hacer
4          Punto(s) de intersección d_ij (y d_ji)
5          rho.ij.es.mayor ← verdadero
6          mientras rho.ij.es.mayor = verdadero y k ∈ {1, ..., î, ..., ĵ, ..., m} hacer
7              rho.ijk ← Λ^k_{i,j}(λ)
8              si rho.ijk < rho para algún k entonces
9                  rho.ij.es.mayor ← falso
10         si rho.ij.es.mayor = verdadero entonces
11             rho ← mín(rho.ijk)
12         si d_ji existe entonces
13             mientras rho.ij.es.mayor = verdadero y k ∈ {1, ..., î, ..., ĵ, ..., m}
                hacer
14                 rho.jik ← Λ^k_{j,i}(λ)
15                 si rho.jik < rho para algún k entonces
16                     rho.ij.es.mayor ← falso
17             si rho.ij.es.mayor = verdadero entonces
18                 rho ← mín(rho.jik,rho)
19 regresar(rho).
```

Otra alternativa es la siguiente: como para nuestro propósito solamente necesitamos saber si $\rho(M) \geq 0$, podemos utilizar también el siguiente algoritmo. De tal manera que nos indica si la escala arroja un valor negativo o no-negativo:

Algoritmo 5: $\rho(M_\lambda) < 0$ ó $\rho(M_\lambda) \geq 0$.

Entrada: M, λ.
Salida: $-1/1$ dependiendo si $\rho(M_\lambda)$ es negativo o no-negativo.

```
1  para i = 1, ..., m − 1 hacer
2      para j = i + 1, ..., m hacer
3          Punto(s) de intersección d_ij (y d_ji)
4          rho.ij.noneg ← verdadero
5          mientras rho.ij.noneg = verdadero y k ∈ {1, ..., î, ..., ĵ, ..., m} hacer
6              si Λ^k_{i,j}(λ) < 0 para algún k entonces
7                  rho.ij.noneg ← falso
8              si d_ji existe y Λ^k_{j,i}(λ) < 0 para algún k entonces
9                  rho.ij.noneg ← falso
10         si rho.ij.noneg = verdadero entonces
11             regresar(1).
12 regresar(−1).
```

Lema 2.2.7. Sea M un sistema de discos. Entonces M_λ es un sistema de Čech si y sólo si, $\rho_M(\lambda) \geq 0$. En particular $\rho_M(\sqrt{4/3}\lambda_M) \geq 0$.

Demostración. Por el Lema 2.2.2, M_λ es un sistema de Čech si y sólo si, existe $d_{ij}(\lambda)$ tal que $d_{ij}(\lambda) \in D_k^\lambda$ para cada $k \neq i, j$, i.e., $\Lambda_{i,j}^k(\lambda) \geq 0$ para cada $k \neq i, j$, que es equivalente a $\rho_M(\lambda) \geq 0$.

Del Lema 2.2.4, $M_{\sqrt{4/3}\lambda_M}$ es un sistema de Čech y se sigue de lo anterior que $\rho_M(\sqrt{4/3}\lambda_M) \geq 0$. ■

Si M es un sistema de discos y λ^* es la escala de Čech, tenemos por el lema anterior que $\rho_M(\lambda) \geq 0$ por cada $\lambda \geq \lambda^*$. Es claro que si $\rho_M(\lambda_M) \geq 0$, entonces $\lambda^* = \lambda_M$ por la minimalidad de la escala de Čech.

Por otro lado, si $\rho_M(\lambda_M) < 0$ entonces la escala de Čech satisface $\lambda^* \in (\lambda_M, \sqrt{4/3}\lambda_M]$ y además $\rho_M(\lambda^*) = 0$. Ésto es una consecuencia de la continuidad de ρ_M y el hecho de que $\rho_M(\lambda) \leq 0$ para cada $\lambda_M < \lambda \leq \lambda^*$ y $\rho_M(\lambda) \geq 0$ para $\lambda^* \leq \lambda$.

Para encontrar la escala de Čech del sistema de discos necesitamos resolver la ecuación $\rho_M(\lambda) = 0$. Sin embargo, para un sistema de discos M, es posible que haya varias soluciones a esta ecuación. Por ejemplo, la escala de Vietoris-Rips del sistema de discos en la Figura 2.4 es $\lambda_M = 0.9151271$, su escala de Čech (calculada usando el algoritmo `CechScale` de la siguiente sección) es $\lambda^* = 0.9181754$, pero aún así para la escala $\lambda' = 1$ tenemos $\rho_M(\lambda') = \rho_M(\lambda^*) = 0$, y $\lambda', \lambda^* \in [\lambda_M, \sqrt{4/3} \cdot \lambda_M] = [0.9151271, 1.294185]$.

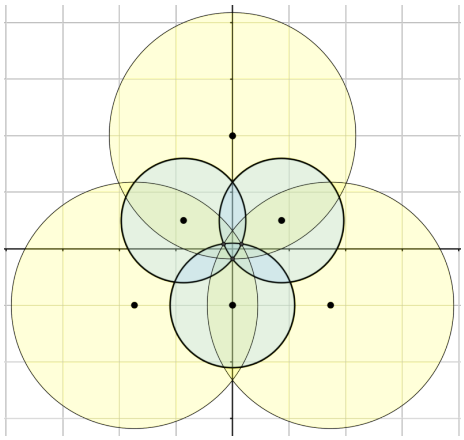


Figura 2.4: Sistema de discos M con más de una solución para ρ_M.

El cálculo de esta escala es fundamental para los objetivos del algoritmo que se propone en este trabajo. Desafortunadamente, el intentar resolver algebraicamente la ecuación $\rho_M(\lambda) = 0$ conlleva esfuerzos nada prácticos ni tangibles. De modo que un acercamiento numérico es más viable.

Por construcción, la función ρ_M sólo puede tener una cantidad finita de raíces en el intervalo $[\lambda_M, \sqrt{4/3}\lambda_M]$. La herramienta que utilizamos para encontrar una raíz de ρ_M en un intervalo $[a, b]$ con una precisión dada, es el método numérico de bisección, de modo

que una condición para poder implementarlo es el cambio de signos de la función en los extremos del intervalo. Si $\rho_M(\lambda_M) \leq 0$, entonces el Lema 2.2.4 garantiza la convergencia del algoritmo en el intervalo $[\lambda_M, \sqrt{4/3}\lambda_M]$.

Algoritmo 6: CechScale

Entrada: M.
Salida: λ^*.

```
λ* ← λ_M
si ρ_M(λ*) ≥ 0 entonces
    regresar(λ*)
en otro caso
    λ* ← √(4/3) λ_M
λ* ← Biseccion(λ_M, λ*)
regresar(λ*)
```

Para el método numérico se consideró utilizar *bisección* ya que es un método estable y respeta el intervalo definido en el cual se encuentra la raíz de nuestra función ρ_M. En la Figura 2.5 se hace la comparación entre los métodos *bisección* y *falsa posición* con $1,000$ repeticiones en cada $10n$ puntos, con $1 \leq n \leq 30$. Se puede observar que el método de falsa posición no es más eficiente que el de bisección en este caso.

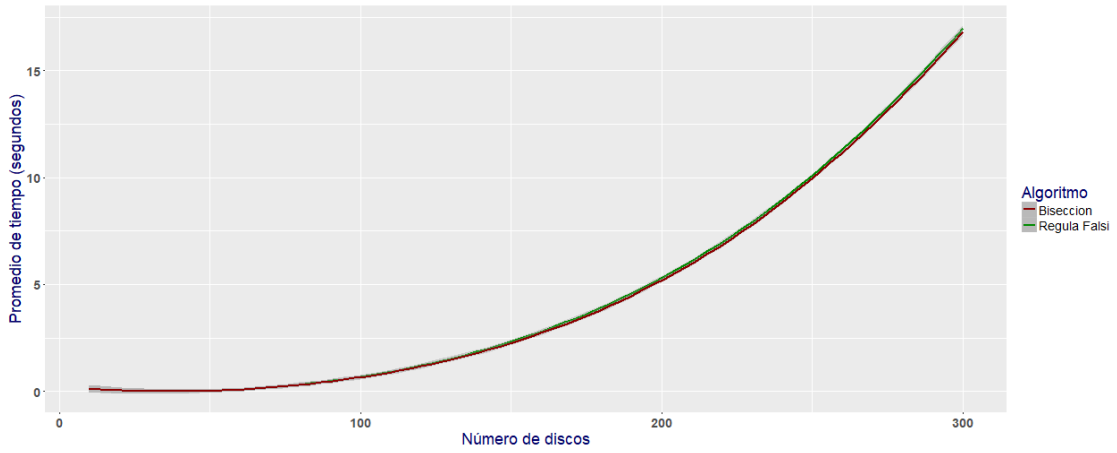


Figura 2.5: Comparación entre métodos numéricos.

Es importante destacar que la raíz λ' encontrada por el pseudocódigo `Biseccion` para $a = \lambda_M$ y $b = \sqrt{4/3}\lambda_M$ no necesariamente es la escala de Čech. Pero iterando el algoritmo para $a = \lambda_M$ y $b = \lambda'$, suficientes veces, obtendremos λ^*. Este proceso es implementado en el script `CechScale` de la siguiente sección. Un resultado que si podemos asegurar es el siguiente:

Lema 2.2.8. Sea $M = \{D_1, D_2, D_3\}$ un sistema de discos tal que $\rho_M(\lambda_M) < 0$. Entonces, existe una única raíz en la función ρ_M en $(\lambda_M, \sqrt{4/3}\lambda_M]$.

Demostración. Se puede verificar que si c_1, c_2 y c_3 son colineales, entonces con cualquier configuración del sistema M se obtiene que $\rho_M(\lambda_M) \geq 0$, se sigue que $\{c_1, c_2, c_3\}$ están en posición general.

Ahora, si los centros no son colineales, tomemos λ^* la escala de Čech del sistema M, que por hipótesis se tiene $\lambda^* \neq \lambda_M$. Podemos asumir sin pérdida de generalidad que $r_1 = \max_i\{r_i\}$.

Nótese que para cualquier $\lambda > \lambda^*$, por la desigualdad del triángulo, tenemos que

$$\|d_{23}(\lambda) - d_{23}(\lambda^*)\| \leq \|d_{23}(\lambda) - c_2\| - \|d_{23}(\lambda^*) - c_2\| = (\lambda - \lambda^*)r_2$$

y

$$\|d_{23}(\lambda) - d_{23}(\lambda^*)\| \leq \|d_{23}(\lambda) - c_3\| - \|d_{23}(\lambda^*) - c_3\| = (\lambda - \lambda^*)r_3.$$

Con la Figura 2.6 podemos ver que al menos una de las desigualdades anteriores debe ser estricta

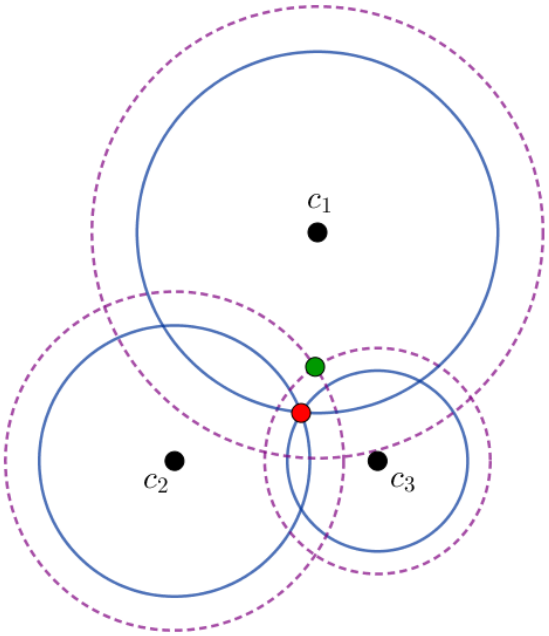


Figura 2.6: Las circunferencias azules representan los $D_i^{\lambda^*}$, las moradas a los D_i^{λ}. El punto rojo representa $d_{23}(\lambda^*)$ y el verde a $d_{23}(\lambda)$.

Para que se de la igualdad $\|d_{23}(\lambda) - d_{23}(\lambda^*)\| = (\lambda - \lambda^*)r_j$, los puntos $d_{23}(\lambda), d_{23}(\lambda^*), c_j$ deberían ser colineales. Si se cumpliera la igualdad para ambos puntos $(j = 2, 3)$, entonces $d_{23}(\lambda) = d_{23}(\lambda^*)$, lo cual no es posible por la hipótesis $\lambda > \lambda^*$.

De esta manera, tenemos lo siguiente

$$\|d_{23}(\lambda) - d_{23}(\lambda^*)\| < (\lambda - \lambda^*)\frac{r_2 + r_3}{2} \leq (\lambda - \lambda^*)r_1$$

así que, $d_{23}(\lambda) \in \mathrm{Int}(D_1^{\lambda})$ o equivalentemente $\lambda r_1 - \|d_{23}(\lambda) - c_1\| > 0$. Se sigue que $\rho_M(\lambda) > 0$ para toda $\lambda > \lambda^*$ entonces λ^* es la única solución de ρ_M. ■

2.2.2. El algoritmo `CechScale`

Como se vio, con el Teorema de Helly (2.1.3), solamente necesitamos calcular las escalas de Čech de los 2–simplejos. Es decir, se calcula la escala de Čech de subsistemas

de tres discos y con esas escalas se pueden calcular las de los demás k–simplejos ($k \geq 3$). Además, con el lema anterior, se probó que nuestro algoritmo proporciona la única raíz de ρ_M, la cual es la escala de Čech del 2–simplejo.

Algoritmo 7: CechScale usando el Teorema de Helly

Entrada: M.
Salida: λ^*.

```
λ* ← λ_M
si ρ_M(λ*) ≥ 0 entonces
    regresar(λ*)
en otro caso
    para 1 ≤ i < j < k ≤ |M| hacer
        N ← {D_i, D_j, D_k}
        Calculamos la escala de Rips, λ_N
        si √(4/3)λ_N ≥ λ* entonces
            si ρ_N(λ_N) ≥ 0 entonces
                λ*_N ← λ_N
            en otro caso
                λ*_N ← Biseccion(λ_N, √(4/3)λ_N)
        si λ*_N existe y λ*_N > λ* entonces
            λ* ← λ*_N
regresar(λ*)
```

El siguiente teorema es la confirmación de que obtenemos lo que buscamos del algoritmo `CechScale`, que es la escala de Čech de un sistema de discos.

Teorema 2.2.9. Para cualquier sistema de discos M, el algoritmo `CechScale` tiene como salida a la escala de Čech λ^* de M.

Demostración. Si ρ_M toma un valor no negativo en la escala de Vietoris-Rips λ_M, entonces M_{λ_M} es un sistema de Čech, por el Lema 2.2.7. Además, por la minimalidad de la escala, λ_M es la escala de Čech.

En el caso de $\rho_M(\lambda_M) < 0$, asignamos el valor $\lambda^* := \sqrt{4/3} \cdot \lambda_M$. Del Lema 2.2.7, $\rho_M(\lambda^*) \geq 0$ y por la continuidad de ρ_M, existe al menos una raíz de ρ_M en el intervalo $[\lambda_M, \lambda^*]$. Este cambio de signos de ρ_M también garantiza la convergencia del método de bisección usado en (`5`). Después de todo esto, obtenemos los resultados esperados. ■

En la siguiente gráfica (Figura 2.7) se muestra el tiempo promedio en el que el algoritmo `CechScale` se ejecuta, con respecto al tamaño del sistema de discos. La comparación se hizo con $1,000$ repeticiones para sistemas de n discos, con $3 \leq n \leq 100$.

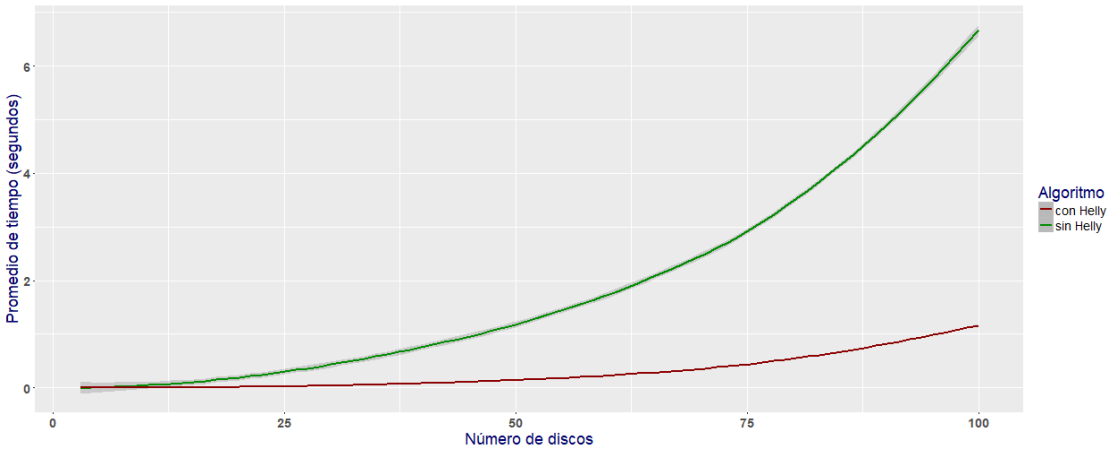


Figura 2.7: Gráfica con los tiempos de `CechScale` con respecto al Teorema de Helly

Desafortunadamente no es claro como generalizar el algoritmo anterior para determinar la escala de Čech de un sistema de bolas en $\mathbb{R}^d$. Sin embargo si es posible determinar la escala de Čech para tres bolas $B(c_i; r_i) \subset \mathbb{R}^d$ con $i = 1, 2, 3$.

La observación clave es que

$$\bigcap_{i=1}^{3} B(c_i; r_i) \neq \varnothing \quad \Leftrightarrow \quad \bigcap_{i=1}^{3} [B(c_i; r_i) \cap P] \neq \varnothing,$$

donde P es el espacio afín generado por c_1, c_2 y c_3.

Más aún, determinar si $\bigcap_{i=1}^{3} [B(c_i; r_i) \cap P]$ es vacío o no, puede plantearse como un problema en $\mathbb{R}^2$, construyendo un *nuevo sistema de discos* en el plano que preserve la configuración de los puntos $\{c_i\}_{i=1}^{3}$ en el espacio afín.

Más precisamente, consideramos la configuración de discos en el plano que se muestra en la Figura 2.8:

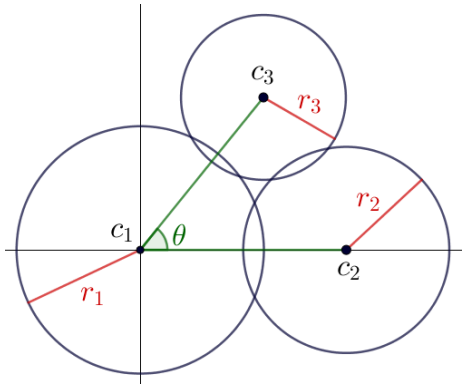


Figura 2.8: Configuración general de tres discos en el plano.

donde θ es el ángulo formado por los vectores $c_2 - c_1$ y $c_3 - c_1$, el cual satisface la siguiente relación:

$$\cos\theta = \frac{\langle c_2 - c_1, c_3 - c_1\rangle}{\|c_2 - c_1\| \cdot \|c_3 - c_1\|}.$$

Así, dado un sistema de tres bolas en $\mathbb{R}^d$,

$$S = \{B(c_1; r_1), B(c_2; r_2), B(c_3; r_3)\},$$

definimos el sistema de discos en el plano como

$$M := \{D_1, D_2, D_3\},$$

dado por $D_1((0,0); r_1), D_2((\|c_2 - c_1\|, 0); r_2), D_3((\|c_3 - c_1\|\cos\theta, \|c_3 - c_1\|\operatorname{sen}\theta); r_3)$.

Por lo tanto, tenemos el siguiente resultado:

Lema 2.2.10. La escala de Čech del sistema de bolas S en $\mathbb{R}^d$ es dada por la escala de Čech del sistema de discos M dado anteriormente.

De esta manera, para sistemas de bolas en $\mathbb{R}^d$ $(d > 2)$ solamente podemos calcular hasta el 2−esqueleto, o bien, en dimensiones mayores del esqueleto si el sistema de bolas en $\mathbb{R}^d$ es coplanar. Pero para $\mathbb{R}^2$ no tenemos restricción para trabajar con dimensiones mayores.

2.3. Aplicación: El problema de la bola minimal en el plano

Consideremos un conjunto finito de puntos en el plano $N \subset \mathbb{R}^2$. El problema de la bola minimal consiste en encontrar el centro $c \in \mathbb{R}^2$ y el radio mínimo $r \in \mathbb{R}^+$ de un disco $D = D(c; r) \subset \mathbb{R}^2$, tal que $N \subset D$. Este problema es un problema clásico de la geometría computacional y fue propuesto inicialmente por el matemático inglés Sylvester en 1857. Actualmente existen maneras de calcular D para un sistema de discos estándar (ver [10]).

Este problema es equivalente a determinar la escala para que un sistema de discos con radios unitarios se intersecte mínimamente. Para encontrar el radio del disco D (o la

escala del sistema) podemos utilizar nuestro algoritmo `CechScale`. Para poder realizar esto, debemos definir el sistema de discos $N_1 = \{D_i(c_i; 1) \subset \mathbb{R}^2 \mid c_i \in N\}$ asociado al conjunto finito de puntos N. La Figura 2.9 muestra un sistema de discos N_1.

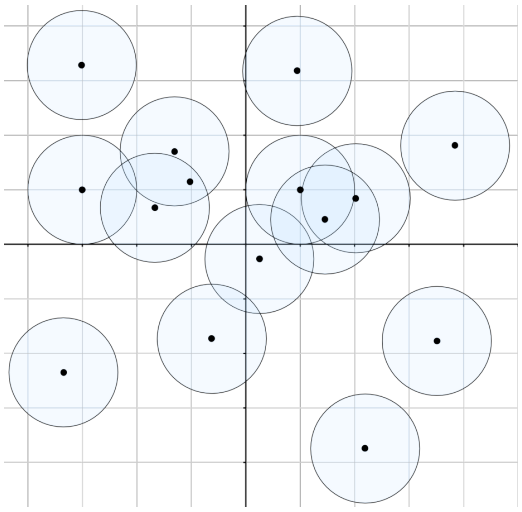


Figura 2.9: Sistema de discos N_1.

Lema 2.3.1. Sea N un conjunto finito de puntos en $\mathbb{R}^2$. Entonces, la escala de Čech λ^* del sistema de discos $N_1 = \{D_i(c_i; 1) \subset \mathbb{R}^2 \mid c_i \in N\}$, es el radio del mínimo disco que acota N.

Demostración. Sea λ^* la salida de `CechScale`(N_1). Entonces realmente estamos trabajando con $N_1^{\lambda^*}$ y se tiene que $\bigcap_{D_i^{\lambda^*} \in N_1^{\lambda^*}} D_i^{\lambda^*} \neq \varnothing$. De tal manera que $\|c_i - c_j\| \leq 2\lambda^*$ para todo $c_i, c_j \in N$, que es lo que buscábamos. ■

Por tanto, con `CechScale` se calculó λ^* y en la Figura 2.10 podemos observar el comportamiento de la bola minimal del conjunto N.

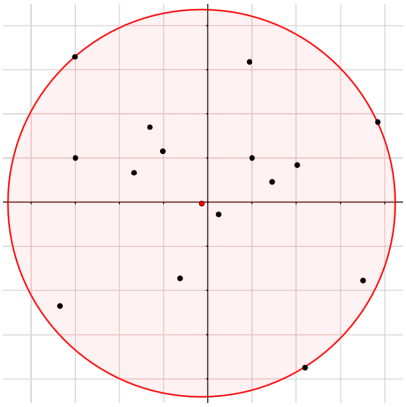


Figura 2.10: Bola minimal de los puntos N con radio λ^*.

Recordemos que nuestro objetivo es calcular la filtración de Čech generalizada pero `CechScale` es capaz de proporcionar el radio del disco D para puntos en $\mathbb{R}^2$ si consideramos el sistema estándar unitario asociado y se calcula su escala de Čech.

2.4. Estructuras simpliciales filtradas de sistemas de Vietoris-Rips y Čech

En esta sección vamos a ver cómo utilizar el algoritmo `CechScale` para construir una filtración de Čech a partir de un conjunto finito de puntos en $\mathbb{R}^d$. Como bien se sabe, no es una forma rápida ni económica de estudiar la topología del conjunto de puntos, pero con el algoritmo mostramos una forma un poco más viable de realizar el cálculo.

La idea general de la sección es encontrar la filtración del conjunto finito de puntos (o un sistema de bolas) con respecto a las escalas de Čech encontradas con `CechScale`. Por cada escala diferente, se tendrá un nivel de la filtración.

Sea M un sistema de bolas. Denotamos por $\mathcal{R}(M)$ a la familia de los subsistemas de Vietoris-Rips de M. Para $\lambda \geq 0$, denotamos por $\mathcal{R}_\lambda(M)$ a la familia $\mathcal{R}(M_\lambda)$, de los subsistemas de Vietoris-Rips del sistema de bolas reescalado por λ, i.e., M_λ. En particular, para $\lambda' \leq \lambda$ tenemos la siguiente contención de familias

$$\mathcal{R}_{\lambda'}(M) \subset \mathcal{R}_\lambda(M).$$

Declaramos las definiciones análogas para subsistemas de Čech con $\mathcal{C}(M)$ y $\mathcal{C}_\lambda(M)$, para $\lambda \geq 0$. También tenemos la contención $\mathcal{C}_{\lambda'}(M) \subset \mathcal{C}_\lambda(M)$, para $\lambda' \leq \lambda$.

Nótese que para cualquier $\sigma \in \mathcal{R}(M)$, todo subsistema de bolas $\tau \subset \sigma$ está también en la familia, i.e., $\tau \in \mathcal{R}(M)$. La misma propiedad es válida para la familia $\mathcal{C}(M)$. Estas propiedades implican que $\mathcal{R}(M)$ y $\mathcal{C}(M)$ son *complejos simpliciales*. Nos referiremos a $\mathcal{R}(M)$ como el *complejo de Vietoris-Rips* asociado al sistema de bolas M y a $\mathcal{C}(M)$ como el *complejo de Čech* de M.

Otra propiedad de estos complejos se cumple con sistemas de discos en el plano gracias al resultado del Lema 2.2.4, que es la siguiente relación entre los diferentes complejos y escalas específicas

$$\mathcal{R}_{\lambda'}(M) \subset \mathcal{C}_\lambda(M) \subset \mathcal{R}_\lambda(M),$$

donde $\sqrt{4/3}\lambda' \leq \lambda$. Si el sistema de discos M se encuentra en $\mathbb{R}^n$ (con $n \geq 3$), el resultado es cierto para los 2−esqueletos.

Definición 2.4.1. Para un sistema de bolas dado M, definimos la **filtración de Vietoris-Rips** de M como la cadena maximal de complejos de Vietoris-Rips

$$\mathcal{R}_*(M) : M_0 =: \mathcal{R}_0(M) \subsetneq \mathcal{R}_{\lambda_1}(M) \subsetneq \cdots \subsetneq \mathcal{R}_{\lambda_M}(M),$$

donde λ_M es la escala de Vietoris-Rips de M y $\lambda_i \neq \lambda_j$ para $i \neq j$. La **filtración de Čech** de M es la cadena maximal de complejos de Čech

$$\mathcal{C}_*(M) : M_0 =: \mathcal{C}_0(M) \subsetneq \mathcal{C}_{\lambda_1}(M) \subsetneq \cdots \subsetneq \mathcal{C}_{\lambda^*}(M,)$$

donde λ^* es la escala de Čech de M.

La definición anterior nos permite estudiar la topología de los sistemas de bolas a través de la homología persistente de sus filtraciones de Vietoris-Rips (o de Čech, respectivamente), como se mostrará en el siguiente capítulo.

Al tener un sistema de bolas M, denotamos por λ_σ^* la escala de Čech del subsistema de bolas $\sigma \subset M$. Observamos que si $\sigma \in \mathcal{R}_\lambda(M)$, entonces $\sigma \in \mathcal{C}_\lambda(M)$ si y sólo si $\lambda_\sigma^* \leq \lambda$.

Sea $\{\lambda_\sigma^* \mid \sigma \in \mathcal{R}_t(M) - \mathcal{R}_s(M)\}$ el conjunto de todas las escalas de Čech de cada subsistema de Vietoris-Rips contenidos entre los términos consecutivos $\mathcal{R}_s(M) \subsetneq \mathcal{R}_t(M)$ de la filtración de Vietoris-Rips del sistema de bolas M. Si $\{\lambda_1, \ldots, \lambda_q\}$ es una sucesión creciente de (diferentes) elementos de la colección, entonces

$$\mathcal{C}_{s_l}(M) := \mathcal{C}_s(M) \bigcup_{\lambda_\sigma^* = \lambda_i} \{\sigma\},\ 1 \leq i \leq l,$$

define la filtración de Čech comprendida entre $\mathcal{C}_s(M)$ y $\mathcal{C}_t(M)$, esto es,

$$\mathcal{C}_s(M) \subsetneq \mathcal{C}_{s_1}(M) \subsetneq \cdots \subsetneq \mathcal{C}_{s_q}(M) \subsetneq \mathcal{C}_t(M).$$

La filtración de Vietoris-Rips es una de las herramientas más utilizadas en el análisis topológico de datos para estudiar la topología de un conjunto finito de puntos, principalmente porque es fácil, rápido y computacionalmente económico de construir. Desafortunadamente, la filtración de Čech no comparte las mismas propiedades.

Un algoritmo eficiente para construir el complejo de Vietoris-Rips, junto con otras estructuras simpliciales, puede encontrarse en [26]. Además, una estructura de datos eficiente para almacenar complejos simpliciales y sus filtraciones está disponible en [2] y [3].

Lo que buscamos conseguir es elaborar un algoritmo como el de Zomorodian en [26], pero en lugar de que genere un complejo de Vietoris-Rips, que genere una filtración de Čech generalizada.

2.5. Algoritmo para la filtración de Čech

Como hemos mencionado durante el capítulo, la clave para definir el nivel de la filtración de Čech de un simplejo es encontrar su escala de Čech, de esta manera tenemos exactamente en qué momento se convierte en un simplejo de Čech. En esta sección mostraremos los algoritmos que utilizamos para generar la filtración y decir explícitamente qué simplejos la forman.

Combinamos los algoritmos Inductive-VR y Compute-Weight de [26], con la función ρ para obtener la filtración de Čech. Los siguientes pseudocódigos muestran la manera de calcular la filtración de Čech de un sistema de bolas M.

Como se mencionó anteriormente, al trabajar con bolas en $\mathbb{R}^d$ $(d \geq 3)$, el problema se modifica para trabajar con discos en $\mathbb{R}^2$. Por tanto si el problema es originalmente en $\mathbb{R}^2$, los simplejos pueden tener dimensión mayor a 2, pero si es en $\mathbb{R}^d$ $(d \geq 3)$ sólo podemos calcular hasta el 2−esqueleto.

Este primer pseudocódigo calcula los discos vecinos, es decir, al tomar un disco del sistema, obtenemos sus discos vecinos o los discos con los que forma aristas:

Algoritmo 8: LowerNbrs.

Entrada: S_1, u.
Salida: Vecinos menores de u.

para $v \in N$ **hacer**
  si $u > v$ y $\{u, v\} \in S_1$ **entonces**
    Añadir v a LN_u.
regresar(LN_u)

El siguiente pseudocódigo proporciona la escala de Čech para un k–simplejo σ ($k \geq 2$), haciendo uso del Teorema de Helly (Teorema 2.1.3) para acelerar un poco el proceso de encontrar la escala de Čech de σ. Puesto que se calcula como el máximo de los valores de sus caras inmediatas, que fueron calculadas anteriormente. En el pseudocódigo definimos a A como las combinaciones con los vértices de σ, para generar sus caras inmediatas.

Algoritmo 9: Weight.

Entrada: σ.
Salida: Parámetro de σ.

si $\dim(\sigma) = 2$ **entonces**
  $\lambda^*_\sigma \leftarrow \texttt{CechScale}(\sigma)$
en otro caso
  $A = \binom{|\sigma|}{|\sigma| - 1}$
  $\lambda^*_\sigma \leftarrow \max_{\alpha \in A}\{\texttt{CechScale}(\alpha)\}$
regresar(λ^*_σ)

Por último, mostramos el pseudocódigo del algoritmo principal `CechFiltration`. En este algoritmo llamamos a todos los demás algoritmos mencionados durante el capítulo:

Algoritmo 10: CechFiltration.

Entrada: M, dimensión máxima, escala máxima.
Salida: Filtración de Čech del sistema M.

```
1  S_0 ← N
2  para cada c_i ∈ N hacer
3      λ_{c_i} ← 0
4  para cada e = {c_i, c_j} ∈ S_0 hacer
5      λ_e ← ||c_i − c_j|| / (r_i + r_j)
6      si λ_e ≤ escalamax entonces
7          Añadir e a S_1
8  para i ← 1 a dimensionmax−1 hacer
9      para cada σ ∈ S_i hacer
10         LN_σ ← ⋃_{u∈σ} LowerNbrs(u).
11         para cada v ∈ LN_σ hacer
12             τ ← v ∪ σ
13             si Weight(τ) ≤ escalamax entonces
14                 Añadir τ a S_{i+1}
```

Con el fin de hacer los cálculos más fáciles (rápidos) para encontrar los simplejos de la filtración, se puede utilizar la función `ripsFiltration` de la biblioteca `TDA` que desarrolla R. De esta manera solamente es eliminar ciertos simplejos con la propiedad $\lambda_M < \lambda^*$, ya que se encontró la filtración de Vietoris-Rips y no de Čech.

El algoritmo `CechFiltration` fue implementado usando R (versión 3.4.3) en una computadora personal con un sistema Windows 10, procesador Intel i7 (2.5 GHz), memoria RAM 4GB, sin paralelización. Pero igualmente se puede adecuar para utilizarse en C. Los scripts de todos los algoritmos usados para este trabajo están en el Apéndice.

Una consideración que se tomó para realizar el algoritmo de R fue que la salida fuera en el formato que PERSEUS pide que sea su entrada. De esta manera no hay necesidad de hacerle cambios a la filtración y se puede calcular la homología persistente del sistema de bolas original M.

Capítulo 3

Topología de un sistema de discos

En este capítulo veremos la topología de un sistema de discos, principalmente veremos cómo calcular la homología simplicial y la homología persistente. Además, explicamos las diferencias entre ellas con un ejemplo y las dos formas principales de exponer la información topológica obtenida de los cálculos de la homología persistente.

Después, mostraremos aplicaciones del análisis topológico de datos y las limitaciones del mismo. Una aplicación importante que resaltamos en este capítulo es poder estudiar la homología persistente de un sistema de discos a través de un software que utiliza la teoría de Morse discreta como un cálculo previo a los cálculos convencionales.

3.1. Homología simplicial

En esta sección definiremos los grupos de homología simplicial, supondremos en adelante que tenemos un complejo simplicial abstracto finito K de dimensión finita. La idea principal es ver qué simplejos son frontera de otros. Nuestro primer paso es definir los objetos algebraicos que utilizaremos.

La homología es una herramienta esencial en la topología algebraica. La idea intuitiva es que se estudia la forma de un objeto mediante sus *agujeros* y cómo se relacionan éstos entre sí. Hay muchas clases de homología, pero aquí trataremos solamente con la homología simplicial.

Para definir los grupos de homología y los operadores que los relacionan, debemos entender primero la idea de tal construcción: Cada grupo está generado por los simplejos de cierta dimensión. Teniendo un elemento del grupo, se envía al grupo generado de los simplejos de dimensión menor consecutiva, a lo que llamamos su frontera.

Para todo $d \geq 0$, definimos el grupo de las d–**cadenas** como el **grupo abeliano libre generado por los d-simplejos orientados de K**, sobre $\mathbb{Z}$, denotado por:

$$C_d(K, \mathbb{Z}).$$

Si $\alpha^d > \sigma^{d-1}$ son simplejos con sus vértices ordenados de la siguiente manera: $\alpha = [c_0, \ldots, c_d]$ y $\sigma = [c_0, \ldots, \hat{c}_i, \ldots, c_d]$, entonces definimos

$$\epsilon(\alpha, \sigma) := (-1)^i,$$

como el número de incidencia de α y σ. Esta función se basa en la orientación de los simplejos que vimos en el primer capítulo (Definición 1.1.8).

Para un simplejo α de dimensión d, definimos su **frontera** como la $(d-1)$–cadena que tiene como sumandos todas las caras de dimensión $d-1$ de α. Extendiendo linealmente, podemos definir el siguiente **operador frontera**:

$$\begin{aligned} \partial_d : C_d(K,\mathbb{Z}) &\to C_{d-1}(K,\mathbb{Z}) \\ \alpha^d &\mapsto \sum_{\sigma^{d-1}<\alpha} \epsilon(\alpha,\sigma)\sigma \end{aligned}$$

donde $\epsilon(\alpha,\sigma)$ es el número de incidencia definido anteriormente. Como definimos este operador para cada simplejo α^d, extendemos $\mathbb{Z}$–linealmente a todo $C_d(K,\mathbb{Z})$.

Al definir estos operadores, buscamos tener una sucesión larga que nos relacione todos los grupos de d–cadenas definidos anteriormente, pero para poder definir esta sucesión es importante que los operadores cumplan la siguiente propiedad:

Proposición 3.1.1. Los operadores frontera cumplen que para cada d,

$$\partial_{d-1} \circ \partial_d = 0.$$

Utilizaremos la idea de la demostración que se presenta en [22], pues ahí se prueba para complejos singulares. Para la demostración, debemos definir un tipo de funciones que nos permiten calcular las caras de un simplejo.

Sea K un complejo y sea $\sigma = [c_0,\dots,c_d]$ un d–simplejo de K, definimos la **aplicación cara** como $\epsilon_i^d(\sigma^d) = [c_0,\dots,\hat{c}_i,\dots,c_d]$. Estas funciones tienen la siguiente propiedad, que es importante para demostrar la Proposición 3.1.1.

Lema 3.1.2. Sea K un complejo. Si $k \le j$, para toda $d \ge 0$, las aplicaciones cara satisfacen

$$\epsilon_j^{d-1} \circ \epsilon_k^d = \epsilon_k^{d-1} \circ \epsilon_{j+1}^d.$$

Demostración. Sea K un complejo y sea $\sigma = [c_0,\dots,c_d]$ un d–simplejo de K. Primero veamos el caso donde $k < j$, por un lado tenemos que

$$\begin{aligned} \epsilon_j^{d-1} \circ \epsilon_k^d([c_0,\dots,c_d]) &= \epsilon_j^{d-1}([c_0,\dots,\hat{c}_k,\dots,c_d]) \\ &= [c_0,\dots,\hat{c}_k,\dots,\hat{c}_j,\dots,c_d] \end{aligned}$$

y por el otro tenemos que

$$\begin{aligned} \epsilon_k^{d-1} \circ \epsilon_{j+1}^d([c_0,\dots,c_d]) &= \epsilon_k^{d-1}([c_0,\dots,\hat{c}_{j+1},\dots,c_d]) \\ &= [c_0,\dots,\hat{c}_k,\dots,\hat{c}_{j+1},\dots,c_d] \end{aligned}$$

pero ahora $\hat{c}_{j+1}$ está realmente en la posición j, por lo que se cumple la igualdad. Ahora veamos el caso donde $k = j$,

$$\begin{aligned} \epsilon_j^{d-1} \circ \epsilon_j^d([c_0,\dots,c_d]) &= \epsilon_j^{d-1}([c_0,\dots,\hat{c}_j,\dots,c_d]) \\ &= [c_0,\dots,\hat{c}_j,\hat{c}_{j+1},\dots,c_d] && \text{pues } c_{j+1} \text{ está en la posición } j \\ &= \epsilon_j^{d-1}([c_0,\dots,\hat{c}_{j+1},\dots,c_d]) \\ &= \epsilon_j^{d-1} \circ \epsilon_{j+1}^d([c_0,\dots,c_d]) \end{aligned}$$

Que es lo que se buscaba. ■

Ahora tenemos las herramientas necesarias para la prueba de la Proposición 3.1.1. Consideremos al complejo K y sea σ un d–simplejo de K, basta probarlo con un d–simplejo ya que el operador es lineal.

$$\begin{aligned}
\partial_{d-1}\circ\partial_d(\sigma) &= \partial_{d-1}\left(\sum_{i=0}^{d}(-1)^i\epsilon_i^d(\sigma)\right)\\
&= \sum_{i=0}^{d}(-1)^i\partial_{d-1}(\epsilon_i^d(\sigma))\\
&= \sum_{i=0}^{d}\sum_{j=0}^{d-1}(-1)^{i+j}\epsilon_j^{d-1}\epsilon_i^d(\sigma)\\
&= \sum_{i\leq j}(-1)^{i+j}\epsilon_j^{d-1}\epsilon_i^d(\sigma)+\sum_{i>j}(-1)^{i+j}\epsilon_j^{d-1}\epsilon_i^d(\sigma)\\
&= \sum_{i\leq j}(-1)^{i+j}\epsilon_i^{d-1}\epsilon_{j+1}^d(\sigma)+\sum_{i>j}(-1)^{i+j}\epsilon_j^{d-1}\epsilon_i^d(\sigma) \qquad \text{por el Lema 3.1.2}\\
&= \sum_{p<q}(-1)^{p+q-1}\epsilon_p^{d-1}\epsilon_q^d(\sigma)+\sum_{j<i}(-1)^{i+j}\epsilon_j^{d-1}\epsilon_i^d(\sigma) \qquad \text{donde } p=i, q=j+1\\
&= 0
\end{aligned}$$

Con lo anterior, podemos considerar el siguiente complejo de cadenas:

$$C_*:\ 0\longrightarrow C_n(K,\mathbb{Z})\xrightarrow{\partial_n}C_{n-1}(K,\mathbb{Z})\xrightarrow{\partial_{n-1}}\cdots\xrightarrow{\partial_1}C_0(K,\mathbb{Z})\longrightarrow 0.$$

Esto nos lleva a la siguiente definición:

Definición 3.1.3. Sea K un complejo simplicial abstracto finito. Definimos el d–**ésimo grupo de homología simplicial de** K sobre $\mathbb{Z}$, como

$$H_d(K,\mathbb{Z}) := \ker\partial_d \big/ \mathrm{im}\partial_{d+1}\,.$$

Al rango de $H_d(K,\mathbb{Z})$ se le llama el d–**ésimo número de Betti** y lo denotamos por $\beta_d(K)$.

Cuando no haya confusión, escribiremos simplemente H_d o β_d, sin mencionar el complejo y los coeficientes a usar. Igualmente, a los operadores frontera los denotaremos por ∂, sin hacer referencia al subíndice que indica la dimensión.

El siguiente resultado, es bastante importante, pues nos muestra como se deben calcular los grupos de homología con diferentes coeficientes. La prueba se puede ver en [22, p. 262].

Teorema 3.1.4 (Teorema de Coeficientes Universales)**.** Sean K un complejo simplicial abstracto y G un grupo abeliano, para toda $n\geq 0$,

$$H_n(K;G)\cong H_n(K;\mathbb{Z})\otimes G\oplus \mathrm{Tor}(H_{n-1}(K,\mathbb{Z}),G).$$

Por lo que se tiene el isomorfismo $H_n(K;G)\cong H_n(K;\mathbb{Z})\otimes G$ cuando la torsión desaparece. Como por ejemplo, con los campos. Esto es de alta importancia cuando se cambia de $\mathbb{Z}$ a $\mathbb{Z}_2$, lo cual es bastante usual en la literatura, pues de esta manera se evitan trabajar con orientación y facilita ciertos cálculos. Pero ésto no se verá a detalle aquí.

3.1.1. Propiedades funtoriales

Nuestro próximo objetivo, es ver que H_d es un funtor entre la categoría de los complejos simpliciales y la de los grupos abelianos. Pero primero veamos los siguientes resultados que darán pie a la prueba.

Sean K y L dos complejos simpliciales abstractos, si $\phi : K \to L$ es un morfismo simplicial, definimos

$$\begin{aligned} \phi_{\#} : C_d(K) &\to C_d(L) \\ \sum a_\sigma \sigma &\mapsto \sum a_\sigma \phi(\sigma) \end{aligned}$$

para cada $d \geq 0$.

De este modo, podemos inducir un morfismo en las $d-$cadenas de complejos en los cuales se tiene definido un morfismo simplicial. Veamos que este morfismo cumple la siguiente propiedad importante:

Lema 3.1.5. Si $\phi : K \to L$ es un morfismo simplicial, entonces $\phi_{\#} : C_d(K) \to C_d(L)$ cumple que $\phi_{\#}\partial = \partial\phi_{\#}$.

Demostración. Sea $\sum a_\sigma \sigma \in C_q(K)$, por un lado se tiene

$$\begin{aligned} \phi_{\#}\partial\left(\sum a_\sigma \sigma\right) &= \phi_{\#}\left(\sum a_\sigma \partial\sigma\right) \\ &= \phi_{\#}\left(\sum a_\sigma \sum (-1)^i \hat{\sigma}_i\right) \\ &= \sum a_\sigma \sum (-1)^i \phi(\hat{\sigma}_i). \end{aligned}$$

Y por otro lado, se tiene

$$\begin{aligned} \partial\phi_{\#}\left(\sum a_\sigma \sigma\right) &= \partial\left(\sum a_\sigma \phi(\sigma)\right) \\ &= \sum a_\sigma \sum (-1)^i \phi(\hat{\sigma}_i). \end{aligned}$$

Siendo así, $\phi_{\#}\partial = \partial\phi_{\#}$, como se buscaba. ■

Tenemos lo necesario para poder ver el siguiente resultado. Dicho resultado es al que se quería llegar, pues es el resultado principal de la sección.

Teorema 3.1.6. Para toda $d \geq 0$, H_d es un funtor entre la categoría $\mathcal{K}$ (complejos simpliciales) y Ab (grupos abelianos), i.e., $H_d : \mathcal{K} \to \mathrm{Ab}$.

Demostración. El operador H_d está definido para objetos de $\mathcal{K}$ y, para morfismos simpliciales $\phi : K \to L$, por lo que definimos de manera natural el siguiente morfismo inducido

$$\begin{aligned} \phi_* : H_d(K) &\to H_d(L) \\ z + \mathrm{Im}_{d+1}(K) &\mapsto \phi_{\#}(z) + \mathrm{Im}_{d+1}(L) \end{aligned}$$

el cual ya está definido con objetos de Ab. Por tanto $H_d : \mathcal{K} \to \mathrm{Ab}$ es un funtor. ■

Por lo que, si tenemos un morfismo simplicial $\phi : K \to L$, podemos hablar del morfismo inducido en homología $\phi_* : H_d(K) \to H_d(L)$ definido en la demostración anterior.

Otra propiedad funtorial que se obtiene, es entre la categoría de los complejos simpliciales y la categoría de los espacios topológicos, a través de la realización geométrica de un complejo simplicial abstracto.

Teorema 3.1.7. El espacio subyacente de un complejo simplicial define a $|\cdot| : \mathcal{K} \to \text{Top}$, el funtor entre las categorías $\mathcal{K}$ y Top (espacios topológicos).

Demostración. Sea $\phi : K \to L$ un morfismo simplicial entre los complejos simpliciales K y L. Para cada $s \in K$ definimos $f_s : s \to |L|$ como la aplicación afín determinada por

$$\phi|_{\text{Vert}(s)} : s \to |L|$$

que es un morfismo simplicial.

Entonces, por la condición de intersección de la definición de complejo simplicial, las funciones coinciden en sus intersecciones. También, por el lema del pegado, tenemos una única función continua

$$|\phi| : |K| \to |L|,$$

veamos que esta función es un funtor:

Primero veamos que sucede con el morfismo identidad:

$$|\text{Id}|(s) = \text{Id}|_{\text{Vert}(s)}\left(\sum t_i c_i\right) = \sum t_i \text{Id}|_{\text{Vert}(s)}(c_i) = \sum t_i c_i = \text{Id}(s)$$

Ahora con la composición, por un lado tenemos que

$$|f \circ g|(s) = f \circ g|_{\text{Vert}(s)}\left(\sum t_i c_i\right) = \sum t_i (f \circ g|_{\text{Vert}(s)}(c_i)) = \sum t_i (f \circ g(c_i)),$$

y por el otro lado tenemos que

$$|f| \circ |g|(s) = |f|(g|_{\text{Vert}(s)}\left(\sum t_i c_i\right)) = |f|\left(\sum t_i g(c_i)\right) = \sum t_i (f \circ g(c_i)).$$

De esta manera, se tiene que $|\cdot|$ es un funtor. ■

3.1.2. Ejemplos

Para dejar claros los conceptos y resultados vistos en la sección, mostraremos algunos ejemplos donde se hacen los cálculos pertinentes. Primero, consideremos este ejemplo al que regresaremos más adelante.

Ejemplo 3.1.8. Consideremos el complejo simplicial abstracto $K = \{\varnothing, c_1, c_2, c_3, c_4, c_5, [c_1, c_2], [c_2, c_3], [c_2, c_4], [c_3, c_4], [c_3, c_5], [c_4, c_5], [c_3, c_4, c_5]\}$ (Figura 3.1):

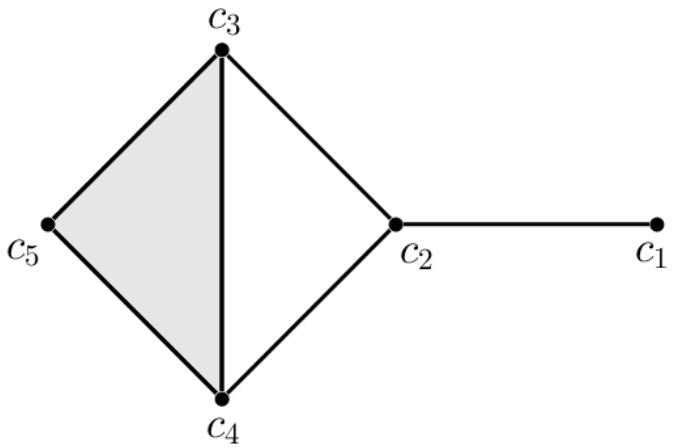


Figura 3.1: Complejo simplicial K.

Éste induce el siguiente complejo de cadenas:

$$C_*(K): 0 \longrightarrow \langle [c_3,c_4,c_5]\rangle \xrightarrow{\partial_2} \langle [c_1,c_2],[c_2,c_3],[c_2,c_4],[c_3,c_4],[c_3,c_5],[c_4,c_5]\rangle \xrightarrow{\partial_1}$$

$$\langle c_1,c_2,c_3,c_4,c_5\rangle \xrightarrow{\partial_0} 0$$

Entonces, como el contradominio de ∂_0 es 0, tenemos que

$$\operatorname{Ker}\partial_0 \cong \mathbb{Z}[c_1]\oplus\mathbb{Z}[c_2]\oplus\mathbb{Z}[c_3]\oplus\mathbb{Z}[c_4]\oplus\mathbb{Z}[c_5].$$

Sea α un elemento arbitrario de $C_1(K)$, es decir,

$$\alpha = a_1[c_1,c_2]+a_2[c_2,c_3]+a_3[c_2,c_4]+a_4[c_3,c_4]+a_5[c_3,c_5]+a_6[c_4,c_5].$$

Por tanto,

$$\partial_1(\alpha) = a_1(c_2-c_1)+a_2(c_3-c_2)+a_3(c_4-c_2)+a_4(c_4-c_3)+a_5(c_5-c_3)+a_6(c_5-c_4).$$

Entonces,

$$\operatorname{Im}\partial_1 \cong \mathbb{Z}[c_2-c_1]\oplus\mathbb{Z}[c_3-c_2]\oplus\mathbb{Z}[c_4-c_2]\oplus\mathbb{Z}[c_4-c_3]\oplus\mathbb{Z}[c_5-c_3]\oplus\mathbb{Z}[c_5-c_4].$$

Y así,

$$H_0(K) \cong \frac{\mathbb{Z}[c_1]\oplus\mathbb{Z}[c_2]\oplus\mathbb{Z}[c_3]\oplus\mathbb{Z}[c_4]\oplus\mathbb{Z}[c_5]}{\mathbb{Z}[c_2-c_1]\oplus\mathbb{Z}[c_3-c_2]\oplus\mathbb{Z}[c_4-c_2]\oplus\mathbb{Z}[c_4-c_3]\oplus\mathbb{Z}[c_5-c_3]\oplus\mathbb{Z}[c_5-c_4]} \cong \mathbb{Z}[c_1],$$

pues $c_1 = c_2 = \cdots = c_5$. Ahora, vemos que

$$\begin{aligned}\partial_1(\alpha) &= a_1(c_2-c_1)+a_2(c_3-c_2)+a_3(c_4-c_2)+a_4(c_4-c_3)+a_5(c_5-c_3)+a_6(c_5-c_4)\\ &= -a_1c_1+(a_1-a_2-a_3)c_2+(a_2-a_4-a_5)c_3+(a_3+a_4-a_6)c_4+(a_5+a_6)c_5,\end{aligned}$$

si $\partial_1(\alpha)=0$, tenemos que $a_1=0$, $a_2=-a_3=a_4+a_5$ y $a_6=-a_5$. Por tanto,

$$\operatorname{Ker}\partial_1 \cong \mathbb{Z}[w_1]\oplus\mathbb{Z}[w_2],$$

donde:

$$w_1 = [c_2,c_3]-[c_2,c_4]+[c_3,c_4],$$
$$w_2 = [c_2,c_3]-[c_2,c_4]+[c_3,c_5]-[c_4,c_5].$$

Ahora, sea β un elemento arbitrario de $C_2(K)$, es decir,

$$\beta = b[c_3,c_4,c_5].$$

Entonces,

$$\partial_2(\beta) = b([c_4,c_5]-[c_3,c_5]+[c_3,c_4]).$$

Por tanto,

$$\operatorname{Im}\partial_2 \cong \mathbb{Z}[[c_4,c_5]-[c_3,c_5]+[c_3,c_4]].$$

Y así,

$$\begin{aligned}H_1(K) &\cong \frac{\mathbb{Z}[[c_2,c_3]-[c_2,c_4]+[c_3,c_4]]\oplus\mathbb{Z}[[c_2,c_3]-[c_2,c_4]+[c_3,c_5]-[c_4,c_5]]}{\mathbb{Z}[[c_4,c_5]-[c_3,c_5]+[c_3,c_4]]}\\ &\cong \mathbb{Z}[[c_2,c_3]-[c_2,c_4]+[c_3,c_4]],\end{aligned}$$

pues $[c_3,c_4]=[c_3,c_5]-[c_4,c_5]$. Para $n\geq 2$, tendremos que $H_n(K)=0$.

♦

3.2. Homología persistente

Hemos definido la homología simplicial de un complejo simplicial abstracto dado, y también analizamos el concepto de filtración. La homología persistente une estos dos conceptos, asociados a un mismo complejo simplicial abstracto K, para formar un nuevo objeto algebraico que nos de información topológica sobre el crecimiento de K a través de su filtración.

Cabe mencionar que no existe una única filtración para K, de hecho, la cuestión de determinar una buena filtración que se ajuste a nuestros fines o que facilite nuestro análisis es una cuestión muy importante. Pues con cada filtración se podrían obtener diferentes resultados.

Sea $K_0 \subset K_1 \subset \cdots \subset K_m$ una filtración, y denotamos por

$$\iota_{i,j} : K_i \to K_j$$

a la inclusión canónica entre pasos de la filtración. Podemos ver que $\iota_{i,j}$ es una función simplicial, pues K_i es un subcomplejo de K_{i+1} y además $\iota_{i,j} = \iota_{j-1,j} \circ \cdots \circ \iota_{i,i+1}$.

Como vimos en la sección anterior, para cada $0 \leq i \leq m$ podemos calcular la homología simplical de K_i. Este proceso nos arroja ciertos espacios vectoriales, si trabajamos con coeficientes sobre algún campo, denotados por $H_d(K_i)$. Observemos que si la dimensión de K es n, entonces para $d > n$ todos los grupos de homología serán triviales.

Asimismo, las inclusiones $\iota_{i,j}$ inducen morfismos en homología para cada d:

$$\iota_{i,j_*} : H_d(K_i) \to H_d(K_j).$$

El objetivo principal de la homología persistente es entender como se relacionan los distintos grupos de homología mediante las inclusiones de la filtración. Para esto, analizemos el siguiente ejemplo, el cual utiliza el mismo complejo del Ejemplo 3.1.8.

Ejemplo 3.2.1. Sea $K = \{\varnothing, c_1, c_2, c_3, c_4, c_5, [c_1, c_2], [c_2, c_3], [c_2, c_4], [c_3, c_4], [c_3, c_5], [c_4, c_5], [c_3, c_4, c_5]\}$ y en la Figura 3.2 mostramos la filtración que utilizaremos.

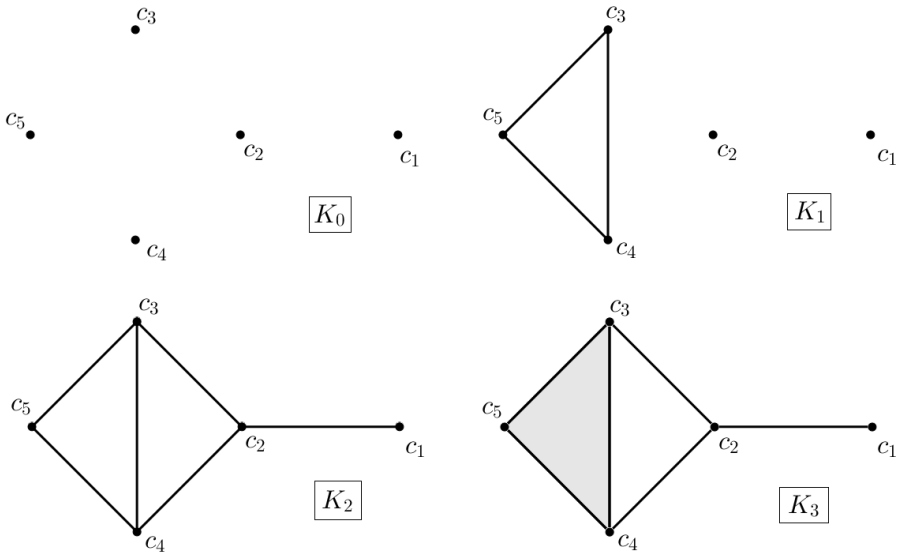


Figura 3.2: Filtración de la realización geométrica del complejo K.

Podemos ver como c_1 pertenece a una clase de homología 0-dimensional, y para K_2 se une a las demás para así tener una sola componente conexa. Por otro lado, vemos que $[c_3, c_4] + [c_3, c_5] + [c_4, c_5]$ pertence a una clase de homología 1-dimensional que aparece en K_1 y desaparece en K_3.

♦

Esta clase de comportamientos serán los que estudiaremos en esta sección, los comportamientos de las clases de homología a través de la filtración.

Definición 3.2.2. Sea K un complejo simplicial abstracto y la filtración $K_0 \subset K_1 \subset \cdots \subset K_m$. Definimos el d-ésimo **grupo de homología persistente** de nivel i, j como

$$H_{i,j;d}(K) \coloneqq \mathrm{Im}(\iota_{i,j})_*$$

a la dimensión de este la denotamos por $\beta_{i,j;d}(K)$ y la llamamos el d-ésimo número de Betti persistente de nivel (i, j).

3.3. Código de barras y diagrama de persistencia

Lo que hemos visto anteriormente es cómo calcular la persistencia de un complejo a través de su filtración, ahora mostraremos una forma de visualizarlo más fácilmente. Como el nacimiento y muerte de las clases tienen un valor cuantitativo, se podrán expresar en algún tipo de gráfica. Por lo que en esta sección se presentan dos tipos de gráficas de la persistencia en una filtración, cada una de ellas tiene sus ventajas y desventajas, las cuales también se mencionarán.

Un **código de barras** es una representación gráfica de $H_{i,j;d}(K)$ que consiste de una colección de segmentos de líneas horizontales en un plano cuyos ejes corresponden al intervalo de persistencia (eje horizontal) y a un orden, que puede ser arbitrario, de los generadores de homología (eje vertical).

Analicemos la Figura 3.3 que muestra el código de barras del Ejemplo 3.2.1:

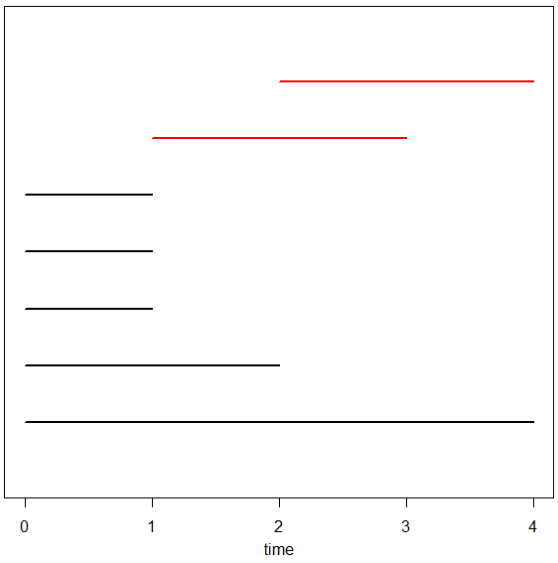


Figura 3.3: Código de barras del Ejemplo 3.2.1.

Las líneas inferiores denotan las clases de homología 0−dimensionales, en tiempo 0 son 5, pues cada 0−simplejo representa una componente conexa. Para tiempo 1 tenemos sólo dos componentes, y en tiempo 2 todas las clases se unen y se vuelven una sola componente conexa. Las líneas superiores (rojas) representan las clases de homología 1−dimensionales, podemos ver que en tiempo 1 nace una clase, que muere en tiempo 3, y en tiempo 2 nace otra, que no muere durante la filtración.

El código de barras nos permite ver quienes son las clases que persisten más y que por tanto son las que más aportan información topológica en nuestra filtración. Podemos ver que este tipo de gráfica es muy visual en el sentido de qué clase puede sustituir a otra, o comparar más fácilmente los nacimientos y muertes de las clases. La desventaja es que entre más clases exista, la gráfica deja de ser clara, pues sería una línea por cada clase.

El **diagrama de persistencia** es una representación gráfica de $H_{i,j;d}(K)$ que consiste de una colección de puntos en un plano, cuyo eje horizontal representa el tiempo en el que nacen las clases y el vertical el tiempo de muerte, donde cada uno los puntos representa a una clase de homología.

Ahora analicemos la Figura 3.4 que muestra el diagrama de persistencia del Ejemplo 3.2.1:

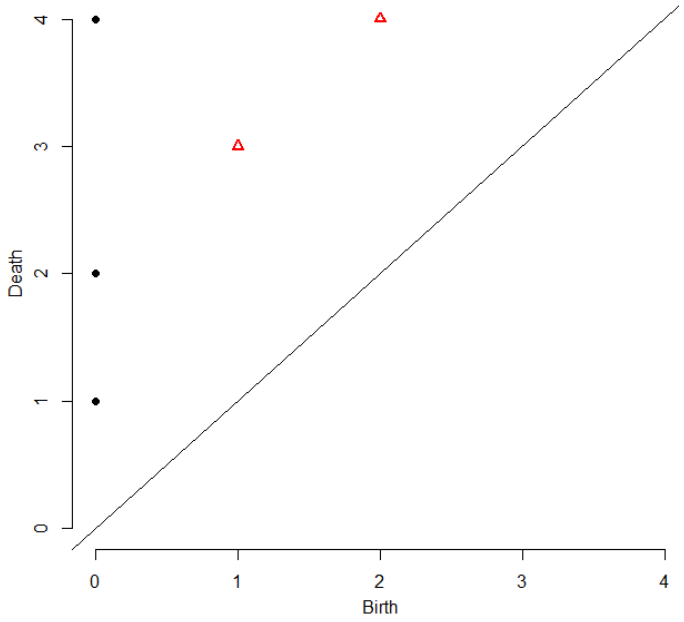


Figura 3.4: Diagrama de persistencia del Ejemplo 3.2.1.

Podemos ver que los puntos nacen en tiempo cero, pues son los que representan a las clases de homología 0−dimensionales, mientras que algunas mueren en tiempo 1, y otras en tiempo 2. El punto que sobrevive es porque se convierten en una sola clase. Los triángulos representan las clases de homología 1−dimensionales.

De esta manera, podremos decir que entre más alejados estén los puntos de la diagonal, implica que son clases que viven más tiempo. Por tanto, los más cercanos a la diagonal podremos considerarlos como ruido topológico. Es decir, consideraremos *ruido topológico* a aquellas clases que mueren rápidamente después de nacer y realmente no dan información relevante del espacio.

Con respecto al ruido topológico, tenemos que considerar una franja de confianza centrada en la diagonal del diagrama, con la cual se necesita tener cierta consideración estadística pues es importante decidir qué se ignora y qué no en el diagrama. Esto no es fácil, pero estos detalles no se tratarán aquí.

Otra desventaja de este tipo de gráficas es que no vemos claramente cuántas clases son las que tienen el mismo nacimiento y muerte. Lo que hace esta gráfica es darle la misma ubicación en el diagrama, por lo tanto sólo sabemos que *existe* al menos una con los datos que nos importa estudiar. Pero aún así, una ventaja de esta gráfica sería que al tener la franja de confianza, podemos estudiar solamente las clases importantes.

3.3.1. Diferencia entre homología persistente y homología en cada nivel de la filtración

Debe de tenerse claro que es diferente tener la homología persistente a tener la homología simplicial en cada nivel de la filtración. Se espera que al exponer el siguiente ejemplo, la diferencia quede expuesta de una manera más clara.

Ejemplo 3.3.1. Sea $N = \{c_1, c_2, c_3, \ldots, c_{3R}\}$ el conjunto de vértices que generará al complejo simplicial K. Consideremos la siguiente filtración (Figura 3.5) de K:

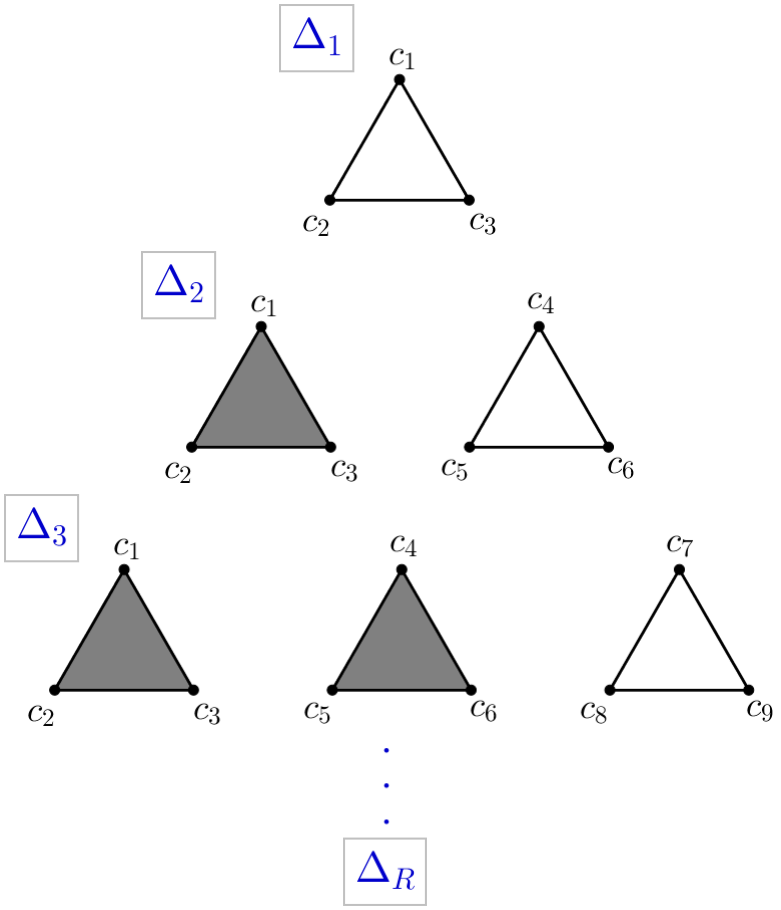


Figura 3.5: Filtración del complejo K.

Los cálculos de homología simplicial para cada nivel de la filtración nos indican que $H_1(\Delta_r, \mathbb{Z}) \cong \mathbb{Z}$ para toda $1 \leq r \leq R$. Sin embargo $H_{i,j;1} = 0$ para $1 \leq i \neq j \leq R$.

⧫

3.4. Análisis topológico de datos

En esta sección se busca aclarar lo que es el análisis topológico de datos (ATD), de esta manera se espera mostrar el interés y la motivación de este trabajo para utilizar estas herramientas para acelerar los cálculos que se han mencionado hasta el momento. Después se mostrarán algunos ejemplos de aplicaciones que ha tenido este campo de estudio.

3.4.1. ¿Qué es el ATD?

El estudio de los datos es, en general, en forma de un conjunto finito de puntos en un espacio desconocido. El ATD se encarga de recuperar la topología del espacio en cuestión. Una herramienta muy utilizada, es la computación para de esta manera agilizar los cálculos pertinentes, por lo tanto se considera una subárea de la topología computacional.

Hablaremos un poco más de cada parte de ATD, primero de la topología: se estudia la forma y las propiedades que no cambian a través de las transformaciones continuas, además de las características que comparten varios espacios. Segundo, los datos: son procesados principalmente de manera digital, lo cual es posible pues se tiene que son una cantidad finita. Pero aún así, puede haber errores y añadir *ruido*.

Por último, el análisis: supongamos que tenemos un conjunto de datos S que está en algún $d-$espacio $\mathbb{Y}$. Supondremos que estos datos se muestrean de algún $k-$espacio $\mathbb{X} \subset \mathbb{Y}$ desconocido. La idea es recuperar la información de $\mathbb{X}$ del conjunto dado S. Para buscar más información y ejemplos de ATD, se puede ver en [27]. En dicho texto se muestran más detalles de todo lo mencionado en esta sección.

3.4.2. ATD en la ciencia

El análisis topológico de datos se utiliza en diferentes áreas de la ciencia y algunos trabajos donde se puede encontrar son los siguientes: El primer ejemplo es en el artículo [4] donde se presenta un estudio sobre pozos de gas shale.

Artículos de medicina que también utilizan ATD son [20] donde exponen un estudio sobre cáncer de mama y en [23] se presenta un estudio sobre embolias pulmonares. También podemos encontrar un artículo que realiza un estudio sobre diabetes mellitus (tipo II) ([24]) y en el artículo [25] se presenta un estudio sobre neurociencias.

3.4.3. Limitaciones en el análisis de datos

Aunque esta herramienta es altamente utilizada, también tiene sus desventajas o limitaciones. El objetivo es encontrar la mejor manera de utilizar estos procedimientos y cálculos para que no sean tan complejos o tan tardados como en los ejemplos que se muestran en este capítulo.

Una desventaja que se estuvo mencionando al momento de crear los complejos, es la cantidad de puntos que se utilizan para el proceso y los cálculos. Siendo más específicos, cuando se tiene un conjunto finito de datos, pero muy grande, se crean complejos simpliciales muy grandes (dimensiones grandes). De este modo, también los grupos de homología son generados por muchos elementos.

Un trabajo que muestra varios ejemplos de complejos es [21]. Lo que hacen es un análisis comparativo entre diferentes complejos (con la mayor dimensión permitida del software), diferentes softwares (librerías abiertas al público), diferentes procesadores y capacidades de una computadora (memoria requerida para cada proceso) y el tiempo que se requiere para cada caso planteado. Ellos muestran tablas comparando con cada

característica antes mencionada.

Una manera de simplificar estos cálculos es utilizando **teoría de Morse discreta**. En lo que nos ayuda esta teoría es en "darle importancia a ciertos simplejos" y al tomar "los importantes", los generadores de los grupos de homología se simplifican. Pero aunque no utilizamos todos los simplejos, se tiene la propiedad de que los grupos de homología son isomorfos (ver [9]).

Actualmente existen varios software que nos ayudan a calcular homología persistente. Uno de ellos es PERSEUS, que calcula la homología persistente de diferentes tipos de complejos filtrados. Consiste en introducir la filtración codificada y obtener los intervalos de homología persistente presentados, de igual manera, en código. Quizá una de las ventajas más importantes que ofrece PERSEUS es que está basado en la teoría de Morse, por tanto no importa el tipo de complejo que se introduce ni la dimensión de éste.

Hay un trabajo, en la literatura, que muestra una manera eficaz de generar funciones de Morse, las cuales son la herramienta de conseguir esos "simplejos importantes". Lo que se hace en [14] es que dado un complejo (más bien, los simplejos que lo componen) junto con una función inyectiva en los vértices, indica específicamente quienes son los simplejos de nuestro interés. Una limitación de este algoritmo, es que el complejo debe de ser a lo más dimensión 2, es decir, los simplejos de mayor dimensión que podemos tener son 2−simplejos.

3.5. Software PERSEUS

En esta sección se va a exponer el funcionamiento del software PERSEUS con un ejemplo. Este software fue diseñado y programado por Mischaikow y Nanda (ver [17]), dicho software está creado para calcular la homología persistente de alguna filtración dada y tiene la particularidad de aceptar distintos tipos de complejos. Para verificar la teoría que utiliza PERSEUS exponemos el Apéndice B.

El algoritmo se basa en la Teoría de Morse Discreta expuesta en [9]. En la literatura, se tiene una manera de calcular la homología persistente (ver [28]), pero este algoritmo primero reduce el complejo y después calcula la homología persistente (ver [19]). El software es gratuito y, para más detalles, se puede consultar [17]. Para que el algoritmo funcione correctamente, debemos entender qué información pide (entrada) y qué información arroja (salida).

Existen varios trabajos en donde se compara la eficiencia de PERSEUS con otros softwares que también calculan homología persistente. Aunque hay más opciones de algoritmos, PERSEUS es el más capaz al momento de comparar el tamaño del conjunto finito de puntos y el tiempo transcurrido por el algoritmo. Uno de estos trabajos es [21].

3.6. Homología persistente de un sistema de discos

Para esta sección debemos recordar que un sistema de discos M es el siguiente conjunto

$$M = \{D_i := D_i(c_i; r_i) \mid c_i \in N\},$$

y podemos obtener una estructura de complejo de Čech de él. Por lo que al obtener la filtración del complejo, cada simplejo es un subsistema con intersección no vacía.

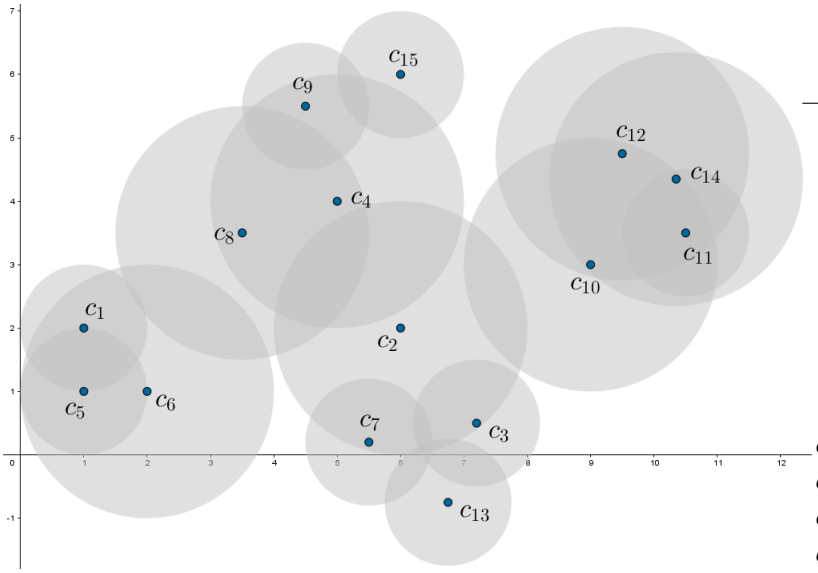
Con la filtración del complejo y las inclusiones canónicas entre los niveles de la misma, podemos obtener los subcomplejos que corresponden a cada nivel y así también los parámetros que los conforman

$$\mathscr{C}(N, \lambda_0) \subset \mathscr{C}(N, \lambda_1) \subset \cdots \subset \mathscr{C}(N, \lambda_k).$$

De esta manera, podemos definir la **homología persistente de un sistema de discos** como

$$H_{i,j;d}(\mathscr{C}(N, \lambda_k)) := \mathrm{Im}(\iota_{i,j})_*.$$

En esta esta sección, vamos calcular la homología persistente del sistema de discos en $\mathbb{R}^2$ expuesto en el Ejemplo 1.2.3. Recordemos el sistema de discos original (Figura 1.9):



N	v	(x, y)	r
c_1	A	$(1, 2)$	1
c_2	B	$(6, 2)$	2
c_3	C	$(7.2, 0.5)$	1
c_4	D	$(5, 4)$	2
c_5	E	$(1, 1)$	1
c_6	F	$(2, 1)$	2
c_7	G	$(5.5, 0.2)$	1
c_8	H	$(3.5, 3.5)$	2
c_9	I	$(4.5, 5.5)$	1
c_{10}	J	$(9, 3)$	2
c_{11}	K	$(10.5, 3.5)$	1
c_{12}	L	$(9.5, 4.75)$	2
c_{13}	M	$(6.75, -0.75)$	1
c_{14}	N	$(10.35, 4.35)$	2
c_{15}	O	$(6, 6)$	1

Al utilizar nuestro algoritmo `CechFiltration` obtenemos la filtración de Čech donde consideramos que los parámetros no pasen de 0.6. Así, los niveles de la filtración con sus parámetros son los siguientes:

nivel	parámetro	nivel	parámetro	nivel	parámetro
1	0.0000000	7	0.4714045	13	0.5273601
2	0.2348537	8	0.4772971	14	0.5335937
3	0.2877113	9	0.4835933	15	0.5590170
4	0.3333333	10	0.5000000	16	0.5813799
5	0.3952847	11	0.5114520		
6	0.4550069	12	0.5270463		

La salida de nuestro algoritmo es un archivo de texto que se ve como la Figura 3.6, donde se expone la información de la siguiente manera:

```
0 1 1
0 2 1
0 3 1
0 4 1
0 5 1
0 6 1
0 7 1
0 8 1
0 9 1
0 10 1
0 11 1
0 12 1
0 13 1
0 14 1
0 15 1
1 1 5 10
1 1 6 7
1 2 4 15
1 4 8 5
1 4 9 12
1 5 6 4
1 10 11 12
1 10 12 6
1 10 14 8
1 11 12 14
1 11 14 3
1 12 14 2
2 10 12 14 9
2 1 5 6 11
2 10 11 14 13
2 10 11 12 16
2 11 12 14 14
3 10 11 12 14 16
```

- La primera columna indica la dimensión del simplejo (k–simplejo).
- La última columna indica el nivel de la filtración y por tanto el parámetro que indica la escala de Čech del simplejo (λ_i).
- De la segunda columna a la penúltima columna indica el simplejo en cuestión ($\sigma = [D_1, \ldots, D_{k+1}]$).

Si tomamos el renglón 29, tenemos al 2–simplejo $\sigma = [D_1, D_5, D_6]$ donde su escala de Čech es $\lambda_{11} = 0.5114520$.

Figura 3.6: Filtración formato .txt del Ejemplo 1.2.3

Recordemos que con cada elemento en la lista anterior de parámetros se genera un nivel de la filtración con al menos un simplejo de diferencia. En la Figura 3.7 mostramos sólo tres niveles de la filtración, para exponer algunos casos:

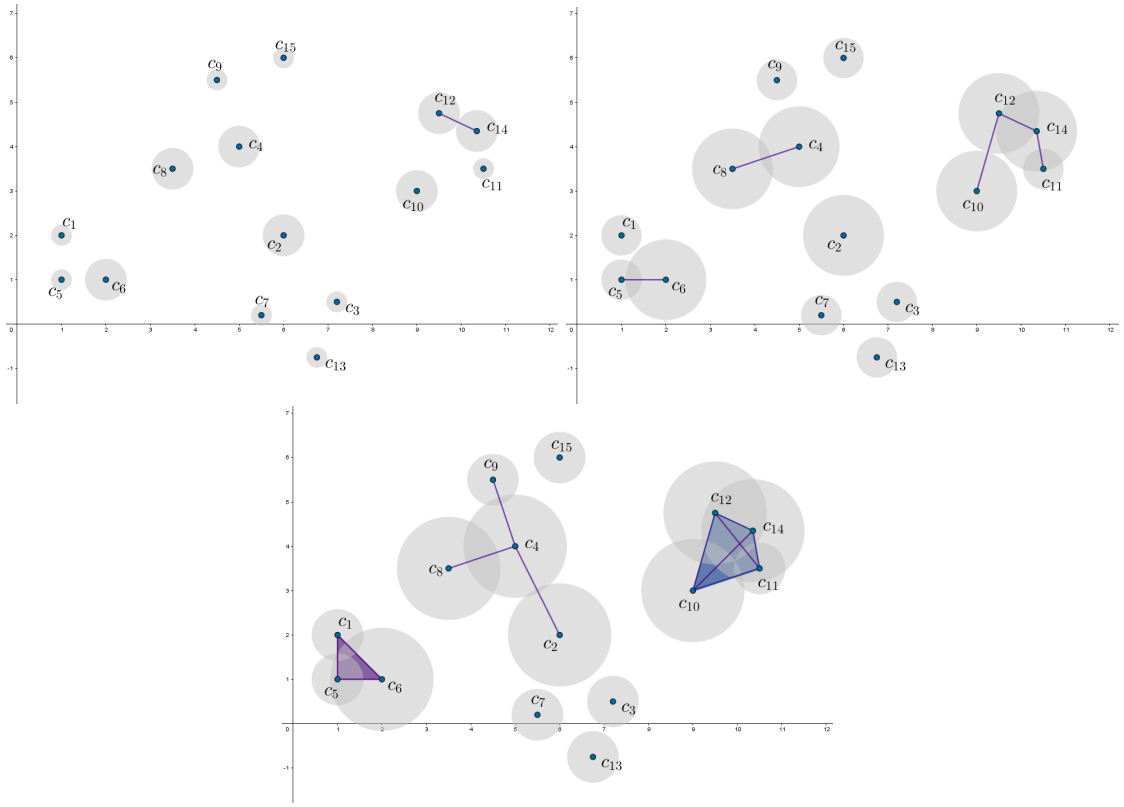


Figura 3.7: Niveles: 2 (arriba izquierda), 6 (arriba derecha), 16 (abajo).

Al tener el archivo de texto que arrojó nuestro algoritmo `CechFiltration`, es momento de utilizar PERSEUS. En [17] se muestran los pasos a realizar para obtener los archivos de texto con la información necesaria para hacer un diagrama de persistencia (Figura 3.8).

A continuación vamos a exponer la homología persistente del sistema de discos $M \subset \mathbb{R}^2$. Por la construcción de la filtración tenemos que todos los centros c_i aparecen en el nivel 1, i.e,

$$H_0(\mathscr{C}(N,0)) = \bigoplus_{i=1}^{15} \mathbb{Z}[c_i],$$

y que en cada nivel de la filtración se anexa al menos un simplejo. Lo que presentamos a continuación son los 0−ésimos grupos de homología persistente de nivel $i, i+1$ de $\mathscr{C}(N, 0.6)$, que denotaremos simplemente por $\mathscr{C}$:

- $H_{1,2;0}(\mathscr{C}) = \bigoplus_{\substack{i=1 \\ i\neq 14}}^{15} \mathbb{Z}[c_i]$

- $H_{2,3;0}(\mathscr{C}) = \bigoplus_{\substack{i=1 \\ i\neq 12,14}}^{15} \mathbb{Z}[c_i]$
- $H_{3,4;0}(\mathscr{C}) = \bigoplus_{\substack{i=1 \\ i\neq 6,12,14}}^{15} \mathbb{Z}[c_i]$
- $H_{4,5;0}(\mathscr{C}) = \bigoplus_{\substack{i=1 \\ i\neq 6,8,12,14}}^{15} \mathbb{Z}[c_i]$
- $H_{5,6;0}(\mathscr{C}) = \mathbb{Z}[c_1]\oplus\mathbb{Z}[c_2]\oplus\mathbb{Z}[c_3]\oplus\mathbb{Z}[c_4]\oplus\mathbb{Z}[c_5]\oplus\mathbb{Z}[c_7]\oplus\mathbb{Z}[c_9]\oplus\mathbb{Z}[c_{10}]\oplus\mathbb{Z}[c_{13}]\oplus\mathbb{Z}[c_{15}]$
- $H_{6,7;0}(\mathscr{C}) = \mathbb{Z}[c_1] \oplus \mathbb{Z}[c_2] \oplus \mathbb{Z}[c_3] \oplus \mathbb{Z}[c_4] \oplus \mathbb{Z}[c_7] \oplus \mathbb{Z}[c_9] \oplus \mathbb{Z}[c_{10}] \oplus \mathbb{Z}[c_{13}] \oplus \mathbb{Z}[c_{15}]$
- $H_{11,12;0}(\mathscr{C}) = \mathbb{Z}[c_1] \oplus \mathbb{Z}[c_2] \oplus \mathbb{Z}[c_3] \oplus \mathbb{Z}[c_4] \oplus \mathbb{Z}[c_7] \oplus \mathbb{Z}[c_{10}] \oplus \mathbb{Z}[c_{13}] \oplus \mathbb{Z}[c_{15}]$
- $H_{14,15;0}(\mathscr{C}) = \mathbb{Z}[c_1] \oplus \mathbb{Z}[c_2] \oplus \mathbb{Z}[c_3] \oplus \mathbb{Z}[c_7] \oplus \mathbb{Z}[c_{10}] \oplus \mathbb{Z}[c_{13}] \oplus \mathbb{Z}[c_{15}]$

Con éstos grupos, podemos calcular los grupos de nivel i, j que faltan. Luego, presentamos los 1−ésimos grupos de homología persistente de nivel $i, i+1$ de $\mathscr{C}(N, 0.6)$, que denotaremos simplemente por $\mathscr{C}$:

- $H_{7,8;1}(\mathscr{C}) = \mathbb{Z}[[c_{10}, c_{14}] - [c_{12}, c_{14}] - [c_{10}, c_{12}]]$
- $H_{8,9;1}(\mathscr{C}) = 0$
- $H_{9,10;1}(\mathscr{C}) = \mathbb{Z}[[c_1, c_5] + [c_5, c_6] - [c_1, c_6]]$
- $H_{10,11;1}(\mathscr{C}) = 0$
- $H_{11,12;1}(\mathscr{C}) = \mathbb{Z}[[c_{10}, c_{11}] + [c_{11}, c_{14}] - [c_{10}, c_{14}]]$
- $H_{12,13;1}(\mathscr{C}) = 0$
- $H_{13,14;1}(\mathscr{C}) = \mathbb{Z}[[c_{10}, c_{11}] + [c_{11}, c_{12}] - [c_{10}, c_{12}]]$
- $H_{15,16;1}(\mathscr{C}) = 0$

En la siguiente figura mostramos el código de barras para nuestro ejemplo.

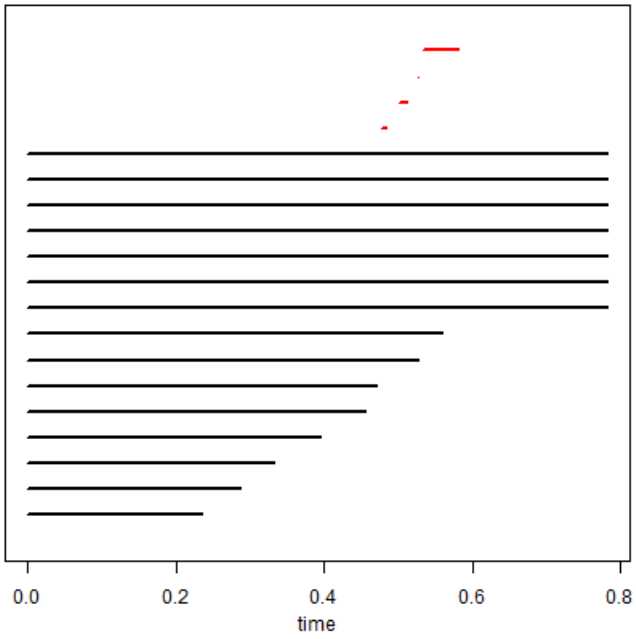


Figura 3.8: El diagrama se mide según los valores de los parámetros encontrados en `CechFiltration`.

Apéndices

Apéndice A

Códigos

En este capítulo se expondrán los scripts que se utilizaron en este trabajo para calcular la filtración de Čech generalizada de un sistema de discos en $\mathbb{R}^2$. De igual manera, los scripts se pueden encontrar en [1]. Primero damos el código principal para luego ir mostrando una a una cada función realizada.

Las bibliotecas que se utilizaron fueron las siguientes:

- `library(‘‘gtools’’)` : para calcular las combinaciones.
- `library(‘‘foreign’’)` : para extraer la filtración en un archivo .txt.
- `library(TDA)`.
- `library(hash)` : para almacenar información.

CechFiltration

Entrada: Sistema de discos, dimensión máxima y parámetro máximo.
Salida: Archivo .txt con los niveles de la filtración.

```
### CREAR R PUNTOS EN S^n ###
info <- CellsGenerator(50, 2)
write.csv(info, file = ‘‘points.csv’’)

### O LEER ARCHIVO .csv CON CENTROS Y RADIOS (x,y,r) ###
info <- scan(‘‘points.csv’’, sep=‘‘,’’)
info <- matrix(info, ncol=3, byrow=TRUE)

prec <- 1e-9   # precision en calculos
k <- 3   # dimension mayor a calcular (0-N)
maxscale <- 0.6   # parametro maximo

param <- Filtration(prec,k,maxscale,info)
```

El siguiente script es el encargado de generar un sistema de discos arbitrario. El sistema se puede generar en $\mathbb{R}^n$ y por cuestiones de experimentación los puntos están en

S^n y los radios tienen medida entre $[0, 1]$.

CellsGenerator

Entrada: Cantidad de discos a crear (r) y la dimensión (n).
Salida: Sistema de r discos en $\mathbb{R}^n$.

```
CellsGenerator} <- function(r, n){
     # coleccion de puntos
     M <- matrix(0, nrow=r, ncol=n+1)
     # puntos de interseccion
     c <- runif(n, 0.5, 0.5)

     for(i in 1:r){
          # radio del disco que contendra ‘‘c’’
          M[i,n+1] <- ratio <- runif(1, 0, 1)
          M[i,1:n] <- as.vector(c + sphereUnif(n=1, d=n-1, r=0.5))
     }
     return(M)
}
```

El siguiente script es el encargado de generar la filtración del sistema de discos y expulsar el archivo de texto con las especificaciones de PERSEUS. En este algoritmo es donde se crean los simplejos y sus parámetros.

Filtration

Entrada: Dimensión máxima, parámetro máximo y sistema de discos.
Salida: Filtración de Čech del sistema de discos.

```
Filtration <- function(prec,k,maxscale,info){
     if(k<0 || maxscale<=0 || nrow(info)==0){
          print(‘‘Nada que calcular, modificar informacion.’’)
     } else {
          # empieza el archivo .txt
          write(1, file = ‘‘Filtration.txt’’, append = FALSE, sep = ‘‘ ’’)

          # S0 es la matriz (x,y) ; N el numero de 0-simp
          S0 <- matrix(info[,-3], ncol=2)
          N <- nrow(S0)

          # agregar 0-esqueleto al archivo .txt
          esq_0 <- matrix(c(rep.int(0,N), 1:N, rep.int(1,N)), nrow=N)
          write(t(esq_0),
                         file = ‘‘Filtration.txt’’,
                         ncolumns = 3,
                         append = TRUE,
                         sep = ‘‘ ’’)
```

```
if(k>=1){
     radii <- matrix(info[,3], ncol=1)

     # matriz de distancia
     D = dist(S0)
     DistM <- as.matrix(D)

     # obtenemos el 1-esqueleto con los parametros
     esq_1 <- OneEsq(DistM, maxscale, prec, radii)

     if(!is.null(esq_1[[2]]) && abs(min(esq_1[[2]])-prec)>0){
          # relacion de simplejos con sus parametros
          tree <- esq_1[[1]]
          # lista de simplejos
          Cech.s <- list(esq_1[[2]])
          # vector de parametros
          param <- c(0,esq_1[[3]])

          if(k>=2){
               # complejo para k-simplejos x (k>=2)
               simplex <- Complex(k, Cech.s,
                    S0, radii,
                    tree, maxscale)
               tree <- simplex[[1]]
               Cech.s <- simplex[[2]]
               param <- c(param, simplex[[3]])
          }

          # remover repeticiones y
          # acomodar los valores de forma creciente
          param <- sort(unique(param), decreasing = FALSE)
          aid <- NULL

          for(i in 1:(length(param)-1)){
               if(abs(param[i]-param[i+1])<=prec){
                    aid <- c(aid, i+1)
               }
          }

          if(!is.null(aid)) {param <- param[-(aid)]}

          # lista de simplejos y niveles de la filtracion
          if(!is.null(Cech.s)){
               Cech.f <- Levels(Cech.s, param, tree)

               # agregar i-esqueletos (i=1, ..., k)
```

```
                        # al archivo .txt
                        for(i in 1:length(Cech.f)){
                             write(t(Cech.f[[i]]),
                                  file = ``Filtration.txt'',
                                  ncolumns = i+3,
                                  append = TRUE,
                                  sep = `` '')
                        }
                   }
              }
         }

         print(``Tu archivo Filtration.txt esta listo.'')
    }
    return(param)
}
```

Este script es el que se utiliza para generar los 1−simplejos del sistema. Cada simplejo tendrá su parámetro y los que cumplan la condición de ser menores o iguales que el máximo serán considerados para la filtración.

OneEsq

Entrada: Matriz de distancias, parámetro máximo y radios de los discos del sistema.
Salida: Los 1−simplejos de la filtración y sus parámetros.

```
OneEsq <- function(DistM, maxscale, prec, radii){
    tree <- hash()
    simplex <- matrix(NA, 0, 2)
    lambda <- NULL

    # consideramos las aristas con parametro <= maxscale
    for(i in 1:(nrow(DistM)-1)){
         for(j in (i+1):ncol(DistM)){
              p <- DistM[i,j]/(radii[i]+radii[j])

              if((p-maxscale)<=prec){
                   simplex <- rbind(simplex, c(i,j))
                   lambda <- c(lambda, p)
                   name <- c(paste0(i, ``.'', j))
                   tree[[name]] <- p
              }
         }
    }
    esq_1 <- matrix(simplex, ncol=2)
    return(list(tree, esq_1, lambda))
}
```

Con el siguiente script calculamos los k–simplejos ($k \geq 2$) y sus parámetros, estos parámetros deben cumplir la condición del máximo. La salida del algoritmo es una *lista*.

Complex

Entrada: El sistema de discos, dimensión máxima, parámetro máximo, el 1-esqueleto y sus parámetros.

Salida: Simplejos de la filtración y sus parámetros.

```
Complex <- function(k, Cech.s, S0, radii, tree, maxscale){
     lambda <- NULL
     simplex <- matrix(NA,1,1)
     i <- 1
     while(i<=(k-1) && nrow(simplex)>0){
          simplex <- matrix(NA, 0, i+2)
          for(jr in 1:nrow(Cech.s[[i]])){
               # para cada (i+1)-simplex:

               # vecinos menores del simplejo jr
               LN <- InterLN(Cech.s, i, jr)
               if(!is.null(LN)){
                    for(l in 1:length(LN)){
                         # supercara del simplejo jr
                         s <- c(LN[l], Cech.s[[i]][jr,])
                         # parametro del nuevo simplejo
                         p <- Weight(s, S0, radii, tree)

                         # si parametro <= maxscale
                         # cuenta en la filtracion
                         if((p-maxscale)<=prec){
                              lambda <- c(lambda, p)
                              simplex <- rbind(simplex, s)
                              name <- s[1]
                              for(n in 2:length(s)){
                                   name <- c(paste0(name, ‘‘.’’, s[n]))
                              }
                              tree[[name]] <- p
                         }
                    }
               }
          }

          # agregar los simplejos a la lista
          if(!is.null(simplex) && nrow(simplex)>0){
               Cech.s[[i+1]] <- matrix(simplex, nrow(simplex))
          }
          i <- i+1
```

```
     }
     return(list(tree, Cech.s, lambda))
}
```

El siguiente script muestra como encontrar los discos vecinos de un discos dado. El siguiente script es para encontrar los discos vecinos pero ahora de un simplejo, los cuales son la intersección de los vecinos de cada vértice. Al tener a los vecinos, generas sus supercaras y son los candidatos a ser simplejos de la filtración.

LowerNbrs

Entrada: Un vértice u y los 1−simplejos.
Salida: Vértices vecinos a u con etiquetas menores.

```
LowerNbrs <- function(S1, u){
     L <- NULL
     j <- 1

     while(j<=nrow(S1)){
          if(S1[j,1]<u){
               if(S1[j, 2]==u) {L <- c(L, S1[j, 1])}
               j <- j+1
          } else {
               j <- nrow(S1)+1
          }
     }
     return(L)
}
```

InterLN

Entrada: Un simplejo s y los simplejos de la misma dimensión.
Salida: Vértices vecinos a todos los vértices de s con etiquetas menores.

```
InterLN <- function(Cech.s, i, jr){
     # vecinos menores del primer vertice
     LN <- LowerNbrs(Cech.s[[1]], Cech.s[[i]][jr, 1])

     # de esos vecinos, tambien deben ser para los todos los vertices
     if(!is.null(LN)){
          jc <- 2
          while(jc<=ncol(Cech.s[[i]])){
               LN.i <- LowerNbrs(Cech.s[[1]], Cech.s[[i]][jr, jc])
               if(is.null(LN.i)){
                    return(LN <- NULL)
               } else {
                    aid <- NULL
                    for(u in 1:length(LN)){
```

```
                    ver <- FALSE
                    v <- 1
                    while(v<=length(LN.i) && ver==FALSE){
                        if(LN[u]==LN.i[v]){
                            ver <- TRUE
                            aid <- c(aid,u)
                        }
                        v <- v+1
                    }
                }
                if(!is.null(aid)){
                    LN <- LN[(aid)]
                } else {
                    LN <- NULL
                    return(LN)
                }
            }
            jc <- jc+1
        }
    }
    return(LN)
}
```

Este script nos muestra cómo encontrar la escala de Čech de cualquier simplejo. Aquí es donde utilizamos el Teorema de Helly con $k-$simplejos donde $k \geq 3$. El siguiente script es el encargado de discernir si la escala de Rips es suficiente para ser la escala de Čech o necesitamos utilizar la función ρ (más adelante).

Weight

Entrada: Sistema de discos, lista de parámetros y simplejo s.
Salida: Escala de Čech del simplejo s.

```
Weight <- function(sigma, S0, radii, tree){
    if(length(sigma)==3){
        B <- S0[sigma,]
        B <- cbind(B, radii[sigma,])
        p <- CechScale(B)
    } else {
        # utilizamos el Teorema de Helly
        A <- combinations(n=length(sigma),
            r=(length(sigma)-1),
            v=sigma,
            repeats=FALSE)
        p <- 0   # parametro de las caras
        for(a in 1:nrow(A)){
            nameA <- A[a,1]
            for(n in 2:ncol(A)){
```

```
                    nameA <- c(paste0(nameA, ‘‘.’’, A[a,n]))
               }
               p <- max(p,tree[[nameA]])
          }
     }
     return(p)
}
```

CechScale

Entrada: Un sistema de discos.
Salida: La escala de Čech del sistema.

```
CechScale <- function(M, prec=1e-9){
     lambda.M <- RipsScale(M)   # escala de Rips

     # rho = max_{i, j} ( min_k ( r_k - ||d_{ij} - c_k|| ))
     rho.M.star <- Rho(M, lambda.M)

     # la escala de Rips es suficiente
     if(rho.M.star >= -prec) {return(lambda.M)}

     # la escala de Rips no es suficiente
     # escala de Rips < escala de Cech <= sqrt(4/3) * escala de Rips
     lambda.star <- RhoRoot(M, lambda.M, sqrt(4/3)*lambda.M)

     return(lambda.star)
}
```

RipsScale

Entrada: Un sistema de discos.
Salida: La escala de Rips del sistema.

```
RipsScale <- function(M){
     m <- dim(M)[1]   # renglones de M
     dist.matrix <- dist(M[, 1:2])   # matriz de distancia
     scales <- c()
     count <- 1

     # parametros de 1-simplejos
     for(i in 1:(m-1)){
          for(j in (i+1):m){
               scales[count] <- dist.matrix[count]/(M[i, 3] + M[j, 3])
               count <- count + 1
          }
     }
     return(max(scales))
}
```

El siguiente script muestra cómo calcular el valor de la función ρ para un sistema de discos y un parámetro, con este valor es el indicador de si el parámetro es o no la escala de Čech.

Rho

Entrada: Un sistema de discos y un parámetro.
Salida: El valor de $\rho(M_\lambda)$.

```
Rho <- function(M, lambda, prec=1e-9){
     m <- dim(M)[1]   # renglones de M
     c <- M[, 1:2]; r <- M[, 3]   # centros y radios
     rho.output <- -Inf   # empezamos con el menor posible

     for(i in 1:(m-1)){
          for(j in (i+1):m){
               # puntos de interseccion de D_i y D_j
               d.M  <- InterPoints(c(c[i, ],
                    lambda*r[i]), c(c[j, ], lambda*r[j]))
               d.ij <- d.M[[1]]
               if(length(d.M)==2) {d.ji <- d.M[[2]]}

               # valores que puede tomar k
               k.values <- (1:m)[-c(i, j)]
               k.index <- 1

               # variable para controlar el maximo
               rho.ij.is.higher <- TRUE

               # valores de rho para i y j
               rho.ijk <- rep(0, m-2)

               while(rho.ij.is.higher && k.index != m-1){
                    k <- k.values[k.index]
                    k.center <- c[k, ]

                    # rho con d.ij
                    rho.ijk[k.index] <- lambda*r[k] -
                         sqrt(sum( (d.ij -
                         c[k, ])^2 ))

                    # si rho.ijk < rho, no sera el maximo
                    if(rho.ijk[k.index] < rho.output){
                         rho.ij.is.higher <- FALSE
                    }
                    k.index <- k.index + 1
               }
```

```
                # si rho.ijk puede ser rho, tomamos el minimo
                if(rho.ij.is.higher) { rho.output <- min(rho.ijk) }

                # si d.ji existe, hacemos lo mismo
                if(length(d.M)==2){
                     k.index <- 1
                     rho.ji.is.higher <- TRUE
                     rho.jik <- rep(0, m-2)

                     while(rho.ji.is.higher && k.index != m-1){
                          k <- k.values[k.index]
                          k.center <- c[3, ]
                          rho.jik[k.index] <- lambda*r[k] -
                               sqrt(sum((d.ji - c[k, ])^2))

                          if(rho.jik[k.index] < rho.output){
                               rho.ji.is.higher <- FALSE
                          }

                          k.index <- k.index + 1
                     }

                     if(rho.ji.is.higher) { rho.output <- min(rho.jik) }
                }
           }
     }
     return(rho.output)
}
```

Para el script anterior es necesario tener los puntos de intersección entre los discos del sistema, este script identifica el (los) punto(s) de intersección entre dos discos. El algoritmo considera las diferentes configuraciones que pueden tener los discos, dependiendo de la ubicación del punto y los valores de los radios.

InterPoints

Entrada: Dos discos del sistema.
Salida: Punto(s) de intersección de los dos discos.

```
Dist <- function(P, Q){ dist(rbind(P,Q))[1] }

InterPoints <- function(D.i, D.j){
     # centros y radios de los discos D.i y D.j
     c.i <- D.i[1:2]; r.i <- D.i[3]
     c.j <- D.j[1:2]; r.j <- D.j[3]

     c.ij <- c.j - c.i
     dist.ij <- Dist(c.i, c.j)
```

```
    # sin interseccion
    if(dist.ij >= r.i+r.j+prec){
        stop(``Los discos tienen interseccion vacia.'')
    }

    # radios demasiado cercanos, seran iguales
    if(abs(r.i-r.j) <= prec) { r.j <- r.i }

    # discos concentricos
    if(dist.ij <= prec){
        d.ij <- c.i
        return(list(d.ij))
    }

    # solo un punto, de manera externa
    if(abs(dist.ij - (r.i+r.j)) < prec){
        d.ij <- (r.j*c.i + r.i*c.j)/(r.i+r.j)
        return(list(d.ij))
    }

    # contencion completa pero no concentricos
    if(all(dist.ij > prec, dist.ij <= abs(r.i-r.j)+prec)){
        r.ij <- abs(r.i-r.j)
        if(r.j > r.i){
            c.i <- c.j; r.i <- r.j; c.ij <- -c.ij
        }
        d.ij <- c.i + r.i*c.ij/r.ij
        return(list(d.ij))
    }

    # dos puntos de interseccion
    n.ij <- c(-c.ij[2], c.ij[1])

    # calculamos los coeficientes a y b tales que
    # d.ij = c.i + a * c.ij + b * n.ij and
    # d.ji = c.i + a * c.ij - b * n.ij
    a <- 0.5*(1 + (r.i-r.j)*(r.i+r.j)/dist.ij^2)
    b <- sqrt(r.i^2 - (a*dist.ij)^2)/dist.ij
    d.ij <- c.i + a * c.ij + b * n.ij
    d.ji <- c.i + a * c.ij - b * n.ij

    return(list(d.ij, d.ji))
}
```

En el siguiente script se presenta el algoritmo para encontrar la raíz de la función ρ. Para ello utilizamos el método numérico de *bisección* y por tanto los valores del intervalo se modifican de esta manera.

RhoRoot

Entrada: Un sistema de discos y dos parámetros a, b (donde $\rho(a) \cdot \rho(b) < 0$).
Salida: La raíz de ρ.

```
RhoRoot <- function(M, a, b, prec=1e-9){
     # numero de iteraciones
     num.iter <- ceiling(abs(log(b-a)-log(prec))/log(2))

     rho.M.a <- Rho(M, a)   # rho(a)
     rho.M.b <- Rho(M, b)   # rho(b)

     n <- 1
     mp <- 0.5*(a + b)   # punto medio
     rho.M.mp <- Rho(M, mp)   # rho(mp)

     # hacemos lo mismo, por <= numero de iteraciones
     while(abs(rho.M.mp)>=prec && n<=num.iter){
          mp <- 0.5*(a + b)
          rho.M.mp <- Rho(M, mp)

          # cambiamos el intervalo
          if(rho.M.mp * rho.M.b < 0){
               a <- mp
          } else {
               b <- mp
          }

          n <- n+1
     }
     return(mp)
}
```

Con este script podemos realizar la filtración como la pide PERSEUS, es decir, con los niveles y no los valores de los parámetros de los simplejos. Lo que hace el algoritmo es cambiar el parámetro por el lugar en la lista (de manera creciente).

Levels

Entrada: Lista de simplejos con sus parámetros y los parámetros en orden creciente.
Salida: Lista de simplejos con sus niveles.

```
Levels <- function(Cech.s, param, tree){
     Cech.f <- Cech.s

     # cambiamos el parametro del simplejo por el nivel de la filtracion
     for(i1 in 1:length(Cech.s)){
          if(nrow(Cech.s[[i1]])>0){
               scale <- NULL
               for(i2 in 1:nrow(Cech.s[[i1]])){
                    i3 <- 1
                    while(i3<=length(param)){
                         namef <- Cech.s[[i1]][i2, 1]
                         for(n in 2:length(Cech.s[[i1]][i2, ])){
                              namef <- c(paste0(namef,
                                   ``.'',
                                   Cech.s[[i1]][i2, n]))
                         }
                         if(abs(tree[[namef]] - param[i3])<=prec){
                              scale <- c(scale, i3)
                              i3 <- length(param)+1
                         } else {
                              i3 <- i3+1
                         }
                    }
               }

               Cech.f[[i1]] <- cbind(rep.int(i1,
                    nrow(Cech.s[[i1]])),
                    Cech.s[[i1]],
                    scale)
          }
     }
     return(Cech.f)
}
```

Al obtener la filtración del sistema de discos, podemos calcular la homología persistente con PERSEUS. Para poder hacer el código de barras de dicha filtración podemos utilizar el siguiente algoritmo.

Persistence

Entrada: Archivos 'output_*.txt' (salida de PERSEUS).
Salida: Archivo .png con el código de barras.

```
Persistence <- function(param){
     # archivos .txt por leer
     filelist = list.files(pattern = ‘‘output_’’)
     filelist <- filelist[-length(filelist)]

     # lista con la info de los archivos .txt
     A <- list()
     for(i in 1:length(filelist)){
          info <- file.info(filelist[i])
          if(info[1]!=0){
               A[[i]] <- read.table(filelist[i])
               A[[i]] <- cbind(rep.int(i-1, nrow(A[[i]])), A[[i]])
               A[[i]] <- matrix(data=as.matrix(A[[i]]), ncol=3)
          }
     }

     # cambiamos el parametro por el nivel
     D <- NULL
     for(i in 1:length(A)){
          B <- A[[i]]
          if(!is.null(B)){
               for(j in 1:nrow(B)){
                    if(B[j, 2]==-1){
                         B[j,2] <- param[length(param)] +(maxscale/3)
                    } else {
                         B[j,2] <- param[B[j,2]]
                    }

                    if(B[j, 3]==-1){
                         B[j, 3] <- param[length(param)] +(maxscale/3)
                    } else {
                         B[j, 3] <- param[B[j, 3]]
                    }
               }
          }

          D <- rbind(D, B)
     }

     # guardar la filtracion en un codigo de barras
     png(file = ‘‘BarcodeFiltration.png’’)
     plot.diagram(D, barcode=TRUE)
     dev.off()

     print(‘‘Tu archivo BarcodeFiltration.png esta listo.’’)
}
```

Apéndice B

Teoría de Morse discreta

En este capítulo se va a explicar el funcionamiento del software PERSEUS. Como se mencionó en el tercer capítulo, este software fue diseñado y programado por Mischaikow y Nanda ([17]) y está creado para calcular la homología persistente de alguna filtración dada. Se presentarán definiciones y resultados imprescindibles sobre álgebra homológica y Teoría de Morse Discreta ([9]) para el entendimiento del algoritmo presentado en [19].

B.1. Complejos y su homología

A partir de esta sección, debemos considerar ciertas nociones que nos servirán como notación a lo largo del escrito. Por ejemplo, como que $\mathbb{N}$ será para denotar a los naturales junto con el 0 y también R será para denotar a un dominio de ideales principales.

Vamos a definir lo que es un complejo y su función de incidencia. También recordaremos lo que es su complejo de cadenas y su homología. De esta manera, comencemos con la definición formal de *complejo*.

Definición B.1.1. Considera un conjunto finito graduado $\chi = \bigsqcup_{n\in\mathbb{N}} \chi_n$ con una función $\kappa : \chi \times \chi \to R$. Denotamos que $x \in \chi_n$ por $\dim x = n$. Entonces (χ, κ) es un **complejo** si la función κ cumple lo siguiente:

i) Para cada par $x, x' \in \chi$, $\kappa(x, x') \neq 0$ implica que $\dim x = \dim x' + 1$.

ii) Para cada par $x, x'' \in \chi$, $\sum_{x'\in\chi} \kappa(x, x')\kappa(x', x'') = 0$.

A un elemento $x \in \chi$ se le llama **celda**, $\dim x$ indica la **dimensión de** x (ó en que χ_n pertenece), y la función κ del complejo (χ, κ), se llama la **función de incidencia**. Cuando no haya confusión del complejo con el que estemos trabajando, podemos simplemente escribir χ.

Es importante mencionar el **orden parcial de cara** ($\leqslant$) que se utilizará a lo largo del capítulo. Este orden está inducido en los elementos de χ de la siguiente manera: para cada par $x, x' \in \chi$:

$$x' < x \quad \text{si} \quad \kappa(x, x') \neq 0.$$

Así, por la definición de κ, la *función dimensión* $\dim : \chi \to \mathbb{N}$ preserva el orden.

Definición B.1.2. Dado (χ, κ), el **complejo de cadenas asociado** $C(\chi)$ es

$$\cdots \xrightarrow{\partial_{n+1}} C_n(\chi) \xrightarrow{\partial_n} C_{n-1}(\chi) \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_1} C_0(\chi)$$

con $C_n(\chi)$ como R–módulos libres con base en las n–celdas. Tenemos al operador frontera $\partial_n : C_n(\chi) \to C_{n-1}(\chi)$ donde, para $x \in \chi$,

$$\partial'_n(x) := \sum_{x' \in \chi} \kappa(x, x')x' = \sum_{x' \in \chi_{n-1}} \kappa(x, x')x'.$$

Así, extendemos ∂'_n a $C_n(\chi)$ como

$$\partial_n \left(\sum a_i x_i \right) := \sum a_i \partial'_n(x_i),$$

donde $a_i \in R$, $x_i \in \chi$.

Por la propiedad de que κ es una función de incidencia, se obtiene que $\partial_n \circ \partial_{n+1} = 0$. También para el R–módulo $C_n(\chi)$, podemos definir los siguientes submódulos Z_n, B_n y el módulo de homología $H_n(\chi)$ que ya conocemos.

Cuando consideramos un subconjunto $\chi' \subset \chi$, la restricción $\kappa' = \kappa|_{\chi' \times \chi'}$ no es una función de incidencia. Sin embargo, tenemos la siguiente definición:

Definición B.1.3. Si χ' es subconjunto de χ, decimos que tiene la **propiedad de subcomplejo** si para cada $\eta \in \chi'$,

$$\{x \in \chi \mid x \leqslant \eta\} \subset \chi'.$$

Cuando se tiene que χ' es un subcomplejo, el complejo de cadenas asociado $C(\chi')$ es subcomplejo de cadenas de $C(\chi)$. También, al tener a χ' como subcomplejo de χ, los módulos de homología $H_n(\chi, \chi')$ están definidos como los módulos de homología relativa $H_n(C(\chi), C(\chi'))$ de complejos de cadena.

B.2. Filtraciones y homología persistente

En esta sección introducimos el concepto de *filtración*, pero para el caso de un complejo en general. Recordemos que una filtración nos indica el avance o crecimiento de un complejo a través del tiempo. De este modo, podemos estudiar la *homología persistente*, para saber qué clases de homología sobreviven a través de la filtración. De aquí en adelante, consideraremos al complejo (χ, κ) sobre el dominio de ideales principales R.

Definición B.2.1. Una **filtración** F de χ es una sucesión finita de subcomplejos $\{\chi^m\}_{m=1}^M$ de χ que cumplen la siguiente condición:

$$\chi^1 \subset \chi^2 \subset \cdots \subset \chi^M = \chi.$$

Así, la **filtración de cadena** asociada $\mathrm{F}(\chi)$ está definida como la siguiente sucesión de complejos de cadena

$$C(\chi^1) \xrightarrow{i^1} C(\chi^2) \xrightarrow{i^2} \cdots \xrightarrow{i^{M-1}} C(\chi^M)$$

donde los morfismos de cadena i^m surgen de las inclusiones de F.

Para cada $m, n \in \mathbb{N}$, tenemos $C_n(\chi^m)$, $Z_n(\chi^m)$ y $B_n(\chi^m)$ del complejo de cadena $C_*(\chi)$. También, para $p \in \mathbb{N}$, consideramos

$$i^{m,p} : C(\chi^m) \to C(\chi^{m+p})$$

como la composición $i^{m+p-1} \circ \cdots \circ i^m$. Pero necesitamos tomar las siguientes convenciones:

- Para $p = 0$, $i^{m,p} = \mathrm{Id}$
- Si $m + p > M$, $i^{m,p} = 0$

De este modo, podemos considerar la siguiente definición:

Definición B.2.2. El n–ésimo módulo de **homología persistente** de χ^m se define como

$$H_n^p(\chi^m) := (i_n^{m,p})_*(H_n(\chi^m)) \cong \frac{i^{m,p}(Z_n(\chi^m))}{i^{m,p}(Z_n(\chi^m)) \cap B_n(\chi^{m+p})},$$

donde el isomorfismo se sigue por inducción sobre p.

B.3. Teoría de Morse discreta para filtraciones

En esta sección se presentan las definiciones y resultados de la Teoría de Morse Discreta, con las cuales se busca combatir las desventajas y complicaciones de las que se hablaron en el tercer capítulo.

Tradicionalmente, la teoría de Morse estudia las conexiones entre variedades compactas suaves y las *funciones de Morse*. Pero en nuestro caso, se hablará más de la Teoría de Morse Discreta descrita por Forman en [9]. Lo que buscamos es un emparejamiento que respete la filtración, por lo que utilizaremos lo que se presenta en [15, Cap.11].

Como vimos en las secciones pasadas, todos los conceptos y resultados se han generalizado para *complejos*. Por lo que aquí no haremos la excepción. Buscamos generalizar la Teoría de Morse Discreta para complejos, para lo cual necesitaremos un complejo (χ, κ) sobre un dominio de ideales principales R.

Definición B.3.1. Un **emparejamiento parcial μ de** (χ, κ) consiste en una partición de χ en tres conjuntos $\mathcal{A}, \mathcal{K}, \mathcal{Q}$ junto con una biyección $\omega : \mathcal{Q} \to \mathcal{K}$ tal que para cada $q \in \mathcal{Q}$, $\kappa(\omega q, q)$ es una unidad en R.

Si tenemos un emparejamiento μ en (χ, κ). Por la Definición B.1.1 y el requisito de la incidencia unitaria, tenemos que

$$\dim \omega q = \dim q + 1 \quad \& \quad q < \omega q \quad \text{para toda } q \in \mathcal{Q}.$$

Además, las celdas en $\mathcal{A}$ se llaman **celdas críticas** y el resto de las celdas son **parejas o emparejados por** μ. Ahora necesitamos sucesiones de celdas, lo que da pie a la siguiente definición.

Definición B.3.2. Un **camino de** μ es una sucesión de celdas

$$\rho = (q_1, \omega q_1, \ldots, q_d, \omega q_d)$$

tal que $q_j \in \mathcal{Q}$ donde $1 \leq j \leq d$ y $q_j \neq q_{j+1} < \omega q_j$ para toda $j = 1, \ldots, d-1$. Este camino se dice ser de longitud d.

Al tener un camino ρ, podemos definir su **principio** y **fin** como:

$$s_\rho = q_1 \quad \& \quad e_\rho = \omega q_d.$$

También podemos hablar de los **ciclos**, que son caminos con $d > 1$ y $s_\rho < e_\rho$. Por lo que un **emparejamiento acíclico** es un emparejamiento donde ninguno de sus caminos es un ciclo. Para nuestros fines, en el resto del capítulo utilizaremos puros emparejamientos acíclicos. A cuyos caminos les podemos definir lo siguiente:

- El **índice de** ρ:

$$ind\rho := \frac{\prod_{j=1}^{d-1} \kappa(\omega q_j, q_{j+1})}{\prod_{j=1}^{d} -\kappa(\omega q_j, q_j)},$$

  en el caso en que ρ sea de longitud 1, se tiene que su índice es $\dfrac{-1}{\kappa(\omega q, q)}$.

- La **multiplicidad de** ρ **de** $a \to a'$:

$$mul(a \xrightarrow{\rho} a') := \kappa(a, s_\rho) \cdot ind\rho \cdot \kappa(e_\rho, a')$$

Estas propiedades de los caminos, nos dan pie a la siguiente definición que nos permitirá hablar de complejos.

Definición B.3.3. Sea $\kappa_\mu : \mathcal{A} \times \mathcal{A} \to R$ definida como

$$(a, a') \mapsto \kappa(a, a') + \sum_\rho mul(a \xrightarrow{\rho} a'),$$

donde la suma se toma sobre todos los caminos ρ de μ.

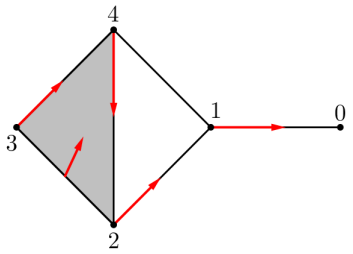
Con las definiciones expuestas en el capítulo, podemos exponer el teorema principal del capítulo y de la Teoría de Morse Discreta. La idea del siguiente teorema es que al tener un complejo más pequeño pero con la información necesaria, tenemos los mismos grupos de homología.

Teorema B.3.4. Sea (χ, κ) un complejo sobre un DIP R y sea $\mu = (\mathcal{A}, \omega : \mathcal{Q} \to \mathcal{K})$ un emparejamiento acíclico de χ. Entonces $(\mathcal{A}, \kappa_\mu)$ es un complejo sobre R y

$$H_*(\chi) \simeq H_*(\mathcal{A}).$$

A $(\mathcal{A}, \kappa_\mu)$ lo llamaremos **complejo de Morse** asociado a μ y a κ_μ la llamaremos **función de incidencia de** $\mathcal{A}$. Veamos un emparejamiento para el Ejemplo 3.1.8.

Ejemplo B.3.5. Consideremos a $R = \mathbb{Z}$, así, en la Figura B.1, cada flecha roja indica una pareja del emparejamiento y los elementos sin ser inicio ni final de flecha son críticos.



$$\mu = \begin{cases} \mathcal{A} = \{0, 14\} \\ \mathcal{Q} = \{1, 2, 3, 4, 23\} \\ \mathcal{K} = \{01, 12, 34, 24, 234\} \end{cases}$$

Figura B.1: Realización geométrica de un complejo simplicial.

Entonces tomemos, por ejemplo, el camino $\rho = (3, 34, 4, 24, 2, 12, 1, 01)$ y le calcularemos su índice:

$$ind\rho = \frac{\kappa(34,4)\kappa(24,4)\kappa(12,1)}{(-1)^4\kappa(34,3)\kappa(24,4)\kappa(12,2)\kappa(01,1)} = -1$$

Calculemos ahora la función de incidencia κ_μ, para la cual sólo ocupamos los simplejos 14 y 0:

$$\begin{aligned} \kappa_\mu(14,0) &= \kappa(14,0) + \sum_\rho mul(14 \xrightarrow{\rho} 0) \\ &= mul\left(14 \xrightarrow{\rho_1=(4,24,2,12,1,01)} 0\right) + mul\left(14 \xrightarrow{\rho_2=(1,01)} 0\right) \\ &= \kappa(14,4)(-1)\kappa(01,0) + \kappa(14,4)(-1)\kappa(01,0) \\ &= (1)(-1)(-1) + (-1)(-1)(-1) = 0 \end{aligned}$$

De esta manera, $\kappa_\mu \equiv 0$ y tenemos siguiente el complejo de cadenas

$$\begin{aligned} C_*(\mathcal{A}) : 0 \longrightarrow \mathbb{Z} &\xrightarrow{\times 0} \mathbb{Z} \longrightarrow 0 \\ 14 &\longmapsto 0 \end{aligned}$$

⧫

B.3.1. El paso de reducción

En esta sección se va a ver un modo de reducir un complejo, es decir quitar elementos y de manera que el nuevo conjunto es efectivamente un subcomplejo del original. La idea general es que al hacer este paso una cierta cantidad finita de veces, el subcomplejo resultante es el complejo de Morse que buscamos.

Para aplicar el *paso de reducción*, necesitamos un complejo de cadenas asociado $(C(\chi), \partial)$ y un emparejamiento acíclico $(\mathcal{A}, \omega : \mathcal{Q} \to \mathcal{K})$ de χ. También necesitamos fijar un $q \in \mathcal{Q}$, de modo que definimos $\chi' \subset \chi$ como $\chi' = \chi \setminus \{q, \omega q\}$ y la función $\kappa' : \chi' \to \chi'$ como

$$\kappa'(\eta, z) = \kappa(\eta, z) - \frac{\kappa(\eta, q)\kappa(\omega q, z)}{\kappa(\omega q, q)}.$$

Al ser κ una función de incidencia, la definición de κ' nos permite considerar la siguiente propiedad para cualesquiera $\eta, \eta' \in \chi'$:

$$\sum_{z \in \chi'} \kappa(\eta, z)\kappa(z, \eta') = -\kappa(\eta, \omega q)\kappa(\omega q, \eta') - \kappa(n, q)\kappa(q, \eta').$$

Más aún, a lo más uno de los términos de la derecha puede ser no cero.

Con un cálculo directo podemos establecer que κ' es una función de incidencia en χ' y así (χ', κ') es un complejo. También se tiene que para los elementos en χ', las funciones κ, κ' coinciden. En particular, la incidencia unitaria de parejas de celdas es preservada en χ'.

Observe que el emparejamiento acíclico μ en χ induce un emparejamiento acíclico μ' en χ' de la forma $(\mathcal{A}, \omega' : \mathcal{Q}' \to \mathcal{K}')$ donde

$$\mathcal{Q}' = \mathcal{Q} \setminus \{q\} \quad \text{y} \quad \mathcal{K}' = \mathcal{K} \setminus \{\omega q\}.$$

Se prueba que la función de incidencia de Morse κ_μ no sufre cambios bajo el paso de reducción. Más específicamente, si $\kappa'_{\mu'}$ es la función de incidencia de Morse inducida por el emparejamiento acíclico μ' en el complejo reducido χ'. Entonces, en $\mathcal{A} \times \mathcal{A}$,

$$\kappa'_{\mu'} \equiv \kappa_\mu.$$

Una consecuencia de estos resultados y otros expuestos en [19] es:

$$H_*(\chi) \simeq H_*(\chi').$$

Finalmente, damos un bosquejo de la prueba del Teorema principal de Teoría de Morse Discreta (B.3.4). Como χ es un conjunto finito, también lo es $\mathcal{Q}$. Por lo que podemos ordenar $\mathcal{Q}$ de alguna manera arbitraria $\mathcal{Q} = \{q_j\}_{j=1}^{J}$.

Consideramos la sucesión de complejos $\chi(j)$ definido de forma inductiva:

$$\chi_0 = \chi \qquad \& \qquad \chi_j = \chi_{j-1} \setminus \{q_j, \omega q_j\} \text{ para } j > 0,$$

i.e., tenemos la siguiente sucesión

$$\mathcal{A} = \chi_J \subset \cdots \chi_j \subset \chi_{j-1} \subset \cdots \subset \chi_0 = \chi.$$

Por lo visto anteriormente y con una cantidad finita de estos pasos, obtenemos lo que buscabamos:

$$H_*(\chi) \simeq H_*(\mathcal{A}).$$

Si consideramos al complejo (χ, κ) con su filtración $\mathrm{F} = \{\chi^m\}_{m=1}^{M}$ y un emparejamiento acíclico $\mu = (\mathcal{A}, \omega : \mathcal{Q} \to \mathcal{K})$ en χ, entonces podemos buscar el emparejamiento en cada nivel de la filtración, i.e., que el emparejamiento respete la filtración F.

Definimos para cada $m \in \{1, \ldots, M\}$, los conjuntos

$$\mathcal{A}^m = \mathcal{A} \cap \chi^m \quad , \quad \mathcal{Q}^m = \mathcal{Q} \cap \chi^m \quad , \quad \mathcal{K}^m = \mathcal{K} \cap \chi^m.$$

En este caso, decimos que μ es F−**subordinado** si para cada $m \in \{1, \ldots, M\}$ la siguiente restricción es una biyección:

$$\omega^m = \omega|_{\mathcal{Q}^m} : \mathcal{Q}^m \to \mathcal{K}^m.$$

Proposición B.3.6. Si μ es F−subordinado, entonces induce un emparejamiento acíclico

$$\mu^m = (\mathcal{A}^m, \omega^m : \mathcal{Q}^m \to \mathcal{K}^m)$$

en cada subcomplejo χ^m, donde $\omega^m : \omega|_{\mathcal{Q}^m}$.

Una convención para indicar que el emparejamiento μ es F−subordinado es

$$\mu^1 \subset \mu^2 \subset \cdots \subset \mu^m = \mu.$$

Al tener un emparejamiento subordinado por una filtración, nuestro complejo de Morse resultante de dicho emparejamiento tiene una filtración que es respetada por el emparejamiento. Un resultado que se obtiene de todo esto, es que si tienes un camino ρ de μ donde $s_\rho \in \chi^m$, entonces también las otras celdas que componen a ρ pertenecen a χ^m.

Así como se tiene una filtración F para χ y un emparejamiento acíclico F−subordinado μ, para el complejo de Morse $(\mathcal{A}, \kappa_\mu)$ asociado, se tiene $(\mathcal{A}^m, \kappa_\mu^m)$ asociado a cada μ^m en χ^m.

Se puede probar que el complejo de Morse $(\mathcal{A}, \kappa_\mu)$ asociado a μ tiene como filtración a $\mathrm{F}_\mu = \{\mathcal{A}^m\}_{m=1}^M$. Llamaremos a F_μ la **filtración de Morse** asociada al emparejamiento acíclico μ que es F−subordinado.

Con todo lo anterior, buscamos generalizar el resultado del Teorema B.3.4 en términos de homología persistente. Así como mencionamos que se puede calcular la homología persistente de un complejo a través de su filtración, queremos tener la posibilidad de hacer comparaciones entre los grupos de homología persistente de ambos complejos.

Teorema B.3.7. Sea $\mathrm{F} = \{\chi^m\}_{m=1}^M$ una filtración del complejo (χ, κ), sea μ un emparejamiento acíclico F−subordinado en χ y sea $(\mathcal{A}, \kappa_\mu)$ el complejo de Morse asociado, con la filtración de Morse $\mathrm{F}_\mu = \{\mathcal{A}^m\}_{m=1}^M$. Entonces para cada $m \in \{1, \ldots, M\}$, $n, p \in \mathbb{N}$

$$H_n^p(\chi^m) \simeq H_n^p(\mathcal{A}^m).$$

B.4. Simplificando el cálculo de la homología persistente

El objetivo de esta sección es describir un algoritmo que toma un complejo filtrado χ (entrada), le impone un emparejamiento acíclico μ subordinado de la filtración, y arroja el complejo de Morse filtrado $\mathcal{A}$ asociado a μ (salida). Esto es posible por la Definición B.2.2 y el Teorema B.3.7.

La popularidad de la homología persistente ha crecido en los últimos tiempos y todo por tener una gran cantidad de datos. Esto ha llevado a una amplia variedad de aplicaciones y algoritmos. En el caso de este algoritmo, sólo se requiere la relación de caras en el complejo filtrado (entrada) codificado por la función de incidencia.

Para esta sección consideraremos, como es usual, al complejo (χ, κ) filtrado por $\mathrm{F} = \{\chi^m\}$. La utilidad de este algoritmo (o de este método, más bien) es encontrar una manera eficaz de construir F_μ y κ_μ. Pero esto depende en gran medida de la elección de μ.

Si consideramos $\mathcal{A} = \chi$ y $\mathcal{Q} = \mathcal{K} = \varnothing$ (llamado el *caso trivial*), obtenemos que las filtraciones son las mismas y no hubo ahorro computacional. Por tanto, la meta aquí es elegir un emparejamiento μ que minimice el número de celdas en $\mathcal{A}$.

B.4.1. Parejas de coreducción y celdas gradientes

En esta sección se verá como generar un emparejamiento acíclico a través del algoritmo de coreducción de homología expuesto en [18], también se utilizará la notación presentada en [12] y [13]. Todos han probado ser efectivos para calcular homología de complejos.

El algoritmo está basado en la siguiente idea: Sea χ un complejo y $\chi' \subset \chi$ un subconjunto que cumple la propiedad de que cualesquiera $z, \eta \in \chi'$ se consideran una pareja de coreducción en χ' si restringido a $C_*(\chi')$ tenemos

$$\partial z = u\eta,$$

donde u es una unidad de R. En este caso, $z \in \mathcal{K}$, $\eta \in \mathcal{Q}$, $\omega\eta = z$ y se remueven z, η de χ'.

Recordemos la Definición B.3.3, entonces κ_μ y ∂_μ se definen de sumar todas los caminos entre celdas de $\mathcal{A}$. Sin embargo, enumerar dichos caminos es poco práctico. Para evitar esta suma usaremos el hecho de que la construcción basada en la coreducción del emparejamiento se realiza construyendo caminos en orden inverso.

Asignaremos a cada celda $z \in \chi$ una cadena $g(z) \in C_*(\mathcal{A})$ llamada la **cadena gradiente**, tal que si $a \in \mathcal{A}$ entonces $g(a) = \partial_\mu(a)$. Iniciamos con $g(z) = 0$, pero como el algoritmo de coreducción se utiliza para construir el emparejamiento acíclico, los valores de $g(z)$ se modifican.

Definición B.4.1. Sea $z \in \chi$, definimos la cofrontera de z como

$$\mathrm{cb}(z) := \{\eta \in \chi \mid z < \eta\}.$$

No será necesario ordenar las celdas ni almacenar una copia por cada vez que aparezca en cada nivel de la filtración, esto se verá a detalle más adelante.

Definición B.4.2. Para cada $m \in \{1, \ldots, M\}$, consideramos los conjuntos

$$N^m = \chi^m \setminus \chi^{m-1},$$

donde $\chi^0 = \varnothing$. Similarmente, definimos

$$N^m_{\mathcal{A}} = \mathcal{A}^m \setminus \mathcal{A}^{m-1}.$$

Podemos ver que cada celda $z \in \chi$ se encuentra exclusivamente en $N^{b(z)}$. También tendremos en cuenta que dada una celda $z \in \chi$, la cofrontera y frontera a $\{N^m\}$ se escribirán respectivamente como

$$\mathrm{cb}_N(z) \quad \text{y} \quad \partial^N(z).$$

Así, cuando una celda sea removida de N^m, también será removida del $\mathrm{cb}_N(z), \partial^N(z)$ correspondiente.

Ahora, presentamos el pseudocódigo donde se modifican las cadenas gradientes. Este algoritmo es llamado varias veces durante el algoritmo, por lo que sólo se explica qué hace al recibir una celda del complejo.

Algoritmo 11: UpdateGradientChain

Entrada: $z \in \chi$.
Salida: Actualizaciones de $g(x)$ para cada $x \in \mathrm{cb}_N(z)$

```
para x ∈ cb_N(z) hacer
    si z ∈ N_A^m para alguna m entonces
        g(x) ← g(x) + κ(x, z)z
    en otro caso
        g(x) ← g(x) + κ(x, z)g(z)
```

B.4.2. Subrutinas para eliminar celdas

En esta sección se van a exponer y explicar dos pseudocódigos que son llamados a través del algoritmo principal. Ambos realizan tareas relacionadas con la eliminación de celdas y los procedimientos que se han mencionado en secciones pasadas.

Dichos algoritmos trabajan en generar el emparejamiento. Recordemos que necesitamos enviar elementos a $\mathcal{A}$ (es lo que hace el primer algoritmo) y el resto de elementos se juntan por parejas para agregarse a $\mathcal{Q}, \mathcal{K}$ respectivamente (es lo que hace el segundo algoritmo).

Algoritmo 12: MakeCritical

Entrada: $m \in \{1, \ldots, M\}$ tal que $N^m \neq \varnothing$.
Salida: $a' \in N_{\mathcal{A}}^m$.

```
Elige a' ∈ N^m de dimensión mínima.
Añade a' a A.
UpdateGradientChain(a').
Quitar a' de N^m.
∂_μ ← g(a').
```

Para el primer algoritmo, se introduce un nivel de la filtración y se obtiene un elemento crítico del nivel. Podemos observar que el elemento a' que se añadió a $\mathcal{A}$, será un generador de $C_*(\mathcal{A})$. Además, la acción de ∂_μ en a' se recupera del correspondiente $g(a')$.

Ahora, el segundo pseudocódigo explica cómo se hace el paso de correducción para cierta pareja en el complejo. Se introducen dos elementos de un nivel de la filtración que serán pareja en μ, por lo tanto se descartan como futuros críticos.

Algoritmo 13: RemovePair

Entrada: $k, q \in N^m$ con $\partial^N k = uq$; Que; $n \in \mathbb{N}$.
Salida: Quita q, k de N^m.

Quita k de N^m
Añade $\mathrm{cb}_N(q)$ en Que.
si $\dim q = n$ **entonces**
    $g(q) \leftarrow -\dfrac{g(k)}{u}$
    `UpdateGradientChain`(q)
Quita q de N^m

Recordemos que, teóricamente, las parejas de coreducción se identifican como $\omega-$parejas y así definen caminos en μ. Antes que la pareja de coreducción se remueva, se deben hacer pasos adicionales que involucran la $\mathrm{cb}_N(q)$ restante.

B.4.3. Algoritmo principal: `MorseReduce`

En esta sección se muestra el algoritmo completo, el cual llama (dependiendo del complejo) las veces que sean necesarias a los algoritmos presentados en las secciones anteriores. Después de presentar el algoritmo, se mostrará un ejemplo donde es empleado el algoritmo `MorseReduce`.

Algoritmo 14: MorseReduce

Entrada: $\{N^m\}_{m=1}^M$.
Salida: $\{N_{\mathcal{A}}^m\}_{m=1}^M$.

para $m = \{1, \ldots, M\}$ **hacer**
    mientras $N^m \neq \varnothing$ **hacer**
        $a' \leftarrow$ `MakeCritical`(m)
        Que $:= \varnothing$
        Añade $\mathrm{cb}_N(a')$ en Que
        mientras Que $\neq \varnothing$ **hacer**
            Quita z de Que
            si $\partial^N z = 0$ **entonces**
                Añade $\mathrm{cb}_N(z)$ en Que
            si no, si $\partial^N z = u\eta$ *para algún* $\eta \in N^{b(z)}$ *y* u *unidad de* R **entonces**
                `RemovePair`$(z, \eta, \mathrm{Que}, \dim a')$

Consideremos el complejo del Ejemplo 3.1.8, el cual es un complejo CW regular. Para este ejemplo, tomaremos una filtración y obtendremos el complejo de Morse junto con su filtración asociada.

Ejemplo B.4.3. En la Figura B.2 se presenta la filtración que utilizaremos:

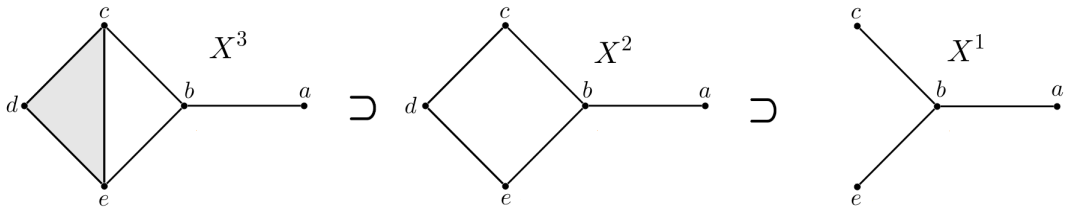


Figura B.2:

Por lo que se tienen a los siguientes conjuntos:

$$N^1 = \{a, b, c, e, ab, bc, be\} \qquad N^2 = \{d, cd, de\} \qquad N^3 = \{ce, cde\}.$$

1.1 $m = 1$

2.1 $N^1 \neq \varnothing$

3.1 $a \in N^1_{\mathcal{A}}$
$\dim a = 0$
$\mathrm{cb}_N(a) = \{ab\}$, entonces $g(ab) = \kappa(ab, a)a$
$\partial_\mu(a) := g(a) = 0$

4.1 Que $:= \varnothing$

5.1 $\{ab\} \subset$ Que

6.1 Que $\neq \varnothing$

7.1 $ab \notin$ Que

10.1 $\partial^N(ab) = \kappa(ab, b)b$

11.1 $\mathrm{cb}_N(b) = \{bc, be\} \subset$ Que

$$g(b) = \frac{\kappa(ab, a)}{\kappa(ab, b)}a$$
$$g(bc) = -\frac{\kappa(ab, a)\kappa(bc, b)}{\kappa(ab, b)}a$$
$$g(be) = -\frac{\kappa(ab, a)\kappa(be, b)}{\kappa(ab, b)}a$$

Por lo que en este primer paso se tiene

$$N^1 = \{c, e, bc, be\} \qquad N^1_{\mathcal{A}} = \{a\} \qquad \text{Que} = \{bc, be\}.$$

6.2 Que $\neq \varnothing$

7.2 $bc \notin$ Que

10.2 $\partial^N(bc) = \kappa(bc, c)c$

11.2 $\mathrm{cb}_N(c) = \varnothing$

$$g(c) = \frac{\kappa(ab, a)\kappa(bc, b)}{\kappa(ab, b)\kappa(bc, c)}a$$

6.3 Que $\neq \varnothing$

7.3 $be \notin$ Que

10.3 $\partial^N(be) = \kappa(be, e)e$

11.3 $\mathrm{cb}_N(c) = \varnothing$

$$g(e) = \frac{\kappa(ab, a)\kappa(be, b)}{\kappa(ab, b)\kappa(be, e)}a$$

Aquí, se terminó el ciclo para $m = 1$ y tenemos los siguientes conjuntos

$$N^1_{\mathcal{A}} = \{a\} \qquad N^1 = \varnothing \qquad \text{Que} = \varnothing.$$

Después de terminar el algoritmo, es decir, realizar los pasos necesarios para $m = 2, 3$, obtenemos los siguientes conjuntos:

$$N^1_{\mathcal{A}} = \{a\} \qquad N^2_{\mathcal{A}} = \{d, cd, de\} \qquad N^3_{\mathcal{A}} = \{ce, cde\},$$

y esto nos permite definir el complejo de Morse y su filtración. En la Figura B.3 podemos ver una realización geométrica de la filtración tanto del complejo original como del complejo de Morse.

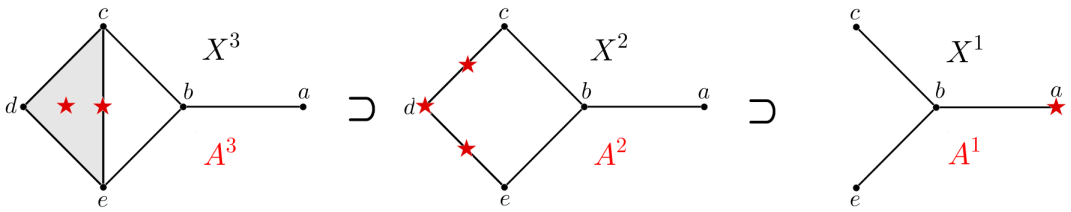


Figura B.3: Las estrellas indican los simplejos del complejo de Morse en cada paso de la filtración.

♦

Gracias al ejemplo anterior, tenemos un complejo con su filtración y su complejo de Morse con su filtración. Lo que nos permite hacer un cálculo de homología persistente de dicho complejo, que es lo que se presenta en el siguiente ejemplo.

Ejemplo B.4.4. Recordemos que la filtración del complejo de Morse es:

$$\mathcal{A}^1 = \{a\} \qquad \mathcal{A}^2 = \{a, d, cd, de\} \qquad \mathcal{A}^3 = \{a, d, cd, ce, de, cde\}.$$

También recordemos los valores del operador frontera del complejo de cadenas asociado, para lo cual tomaremos $\kappa = \kappa_\triangle$:

$$\begin{array}{lll} \partial_\mu(a) = \partial_\mu(d) = 0 & \partial_\mu(cd) = d & \partial_\mu(ce) = 0 \\ \partial_\mu(de) = -d & \partial_\mu(cde) = -ce & \end{array}$$

Ahora, calculemos los grupos de homología en cada nivel de la filtración.

$$C(\mathcal{A}^1): \quad 0 \xrightarrow{\partial_1} \langle a \rangle \xrightarrow{\partial_0} 0$$

Entonces $H_0 \cong \mathbb{Z}[a]$ y $H_i = 0$ para toda $i \geq 1$.

$$C(\mathcal{A}^2): \quad 0 \xrightarrow{\partial_2} \langle cd, de \rangle \xrightarrow{\partial_1} \langle a, d \rangle \xrightarrow{\partial_0} 0$$

Entonces $H_0 \cong \mathbb{Z}[a]$, $H_1 \cong \mathbb{Z}[cd + de]$ y $H_i = 0$ para toda $i \geq 2$.

$$C(\mathcal{A}^3): \quad 0 \xrightarrow{\partial_3} \langle cde \rangle \xrightarrow{\partial_2} \langle cd, ce, de \rangle \xrightarrow{\partial_1} \langle a, d \rangle \xrightarrow{\partial_0} 0$$

Entonces $H_0 \cong \mathbb{Z}[a]$, $H_1 \cong \mathbb{Z}[cd + de]$ y $H_i = 0$ para toda $i \geq 2$.
Por tanto, tenemos los siguientes grupos de homología persistente:

$$\begin{aligned} &H_0^1(\mathcal{A}^1) \cong H_0^1(\mathcal{A}^2) \cong H_0^2(\mathcal{A}^1) \cong \mathbb{Z}[a] \\ &H_1^1(\mathcal{A}^1) = H_1^2(\mathcal{A}^1) = 0 \\ &H_1^1(\mathcal{A}^2) \cong \mathbb{Z}[cd + de] \end{aligned}$$

⧫

En [19] se pueden encontrar los resultados que muestran que el algoritmo es eficiente y que su salida si es lo esperado. Empezando por decir que utilizando el Teorema B.3.7 se puede confirmar que F_μ generada por el algoritmo expuesto aquí tiene la misma homología persistente que la filtración F de χ.

El algoritmo, al ser computacional, se puede considerar con cierto *costo*. Se tiene toda una sección dedicada a este estudio. En dicha sección hay un apartado donde se muestran resultados experimentales con diferentes tipos de complejos (distintos ejemplos) y se hacen comparaciones de memoria, tiempo y detalles de los complejos.

Bibliografía

[1] https://github.com/bramonetti/CechFiltration.git, Septiembre 2018.

[2] J.-D. Boissonnat and C. Maria. The simplex tree: An efficient data structure for general simplicial complexes. Research Report RR-7993, June 2012.

[3] J.-D. Boissonnat and K. C. Srikanta. An efficient representation for filtrations of simplicial complexes. In *Symposium on Discrete Algorithms SODA 2017*, Barcelona, France, Jan. 2017.

[4] A. Cortis. *Topological Data Analysis of Marcellus Play Lithofacies*, pages 336–344. 2015.

[5] L. Danzer, B. Grünbaum, and V. Klee. *Helly's Theorem and It's Relatives*, volume 7 of *Convexity*. American Mathematical Society, 1963.

[6] V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7(1):339–358, 4 2007.

[7] M. E. Espinosa Lara. *Homología Persistente*. CIMAT, 2015.

[8] J. F. Espinoza, R. G. Hernández, H. A. Hernández, and B. Ramonetti. A numerical approach for the filtered generalized Čech complex in the plane. 2018.

[9] R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 3 1998.

[10] B. Gärtner. Fast and robust smallest enclosing balls. In *Proceedings of the 7th Annual European Symposium on Algorithms*, ESA '99, pages 325–338, London, UK, UK, 1999. Springer-Verlag.

[11] J. A. González Lemus. *Teoría de Morse Discreta y Gráficas de Reeb aplicadas a ATD*. CIMAT, 2015.

[12] S. Harker, K. Mischaikow, M. Mrozek, and V. Nanda. Discrete morse theoretic algorithms for computing homology of complexes and maps. *Found. Comput. Math.*, 14(1):151–184, Feb. 2014.

[13] S. Harker, K. Mischaikow, M. Mrozek, V. Nanda, H. Wagner, M. Juda, and P. Dlotko. The efficiency of a homology algorithm based on discrete morse theory and coreductions. 1:41–48, 01 2010.

[14] H. King, K. Knudson, and N. Mramor. Generating discrete morse functions from point data. *Experiment. Math.*, 14(4):435–444, 2005.

[15] D. Kozlov. *Combinatorial Algebraic Topology*. Algorithms and Computation in Mathematics. Springer Berlin Heidelberg, 2008.

[16] N.-K. Le, P. Martins, L. Decreusefond, and A. Vergne. Construction of the generalized Čech complex. *CoRR*, abs/1409.8225, 2014.

[17] K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2):330–353, September 2013.

[18] M. Mrozek and B. Batko. Coreduction homology algorithm. *Discrete & Computational Geometry*, 41(1):96–118, 2009.

[19] V. Nanda. *Discrete Morse Theory for Filtrations*. PhD thesis, The State University of New Jersey, 2012.

[20] M. Nicolau, A. J. Levine, and G. Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011.

[21] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, Aug 2017.

[22] J. J. Rotman. *An Introduction to Algebraic Topology*, volume 119 of *Graduate Texts in Mathematics*. Springer, New York, NY, 1991.

[23] M. Rucco, E. Merelli, D. Herman, D. Ramanan, T. Petrossian, L. Falsetti, C. Nitti, and A. Salvi. Using topological data analysis for diagnosis pulmonary embolism. *Journal of Theoretical and Applied Computer Science*, 9(1):41–55, 2015.

[24] P. Yildirim, M. Bloice, and A. Holzinger. *Knowledge Discovery and Visualization of Clusters for Erythromycin Related Adverse Events in the FDA Drug Adverse Event Reporting System*, pages 101–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[25] J. Yoo, E. Y. Kim, Y. M. Ahn, and J. C. Ye. Topological persistence vineyard for dynamic functional brain connectivity during resting and gaming stages. *Journal of Neuroscience Methods*, 267:1–13, July 2016.

[26] A. Zomorodian. Fast construction of the vietoris-rips complex. *Computers & Graphics*, 34(3):263–271, 2010. Shape Modelling International (SMI) Conference 2010.

[27] A. Zomorodian. Topological data analysis. In A. Zomorodian, editor, *Advances in Applied and Computational Topology*, volume 70, pages 1–39. American Mathematical Society, 2012.

[28] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274, Feb 2005.