Escuela Técnica Superior de
Ingeniería Informática

TRABAJO FIN DE GRADO

Análisis Topológico de las Matrices de Ocurrencia en la Red de Bitcoin

Realizado por
Joan Villalonga Álvaro

Para la obtención del título de
Grado en Ingeniería Informática - Tecnologías Informáticas

Dirigido por
Rocío González Díaz
Nieves Atienza Martínez

Realizado en el departamento de
Matemática Aplicada I

Convocatoria de Julio, curso 2022/23

Agradecimientos

En primer lugar, me gustaría agradecerles a mi pareja y a mi hermana el apoyo incondicional que me han dado siempre, los empujoncitos cuando creía que no lo conseguiría, y simplemente el haber estado ahí siempre que lo he necesitado. Son personas fundamentales para mí y espero seguir aprendiendo de ellas y creciendo a su lado. A mis padres, por buscar lo mejor para mí, y darme una vida lo más feliz y plena que han sabido y podido.

Por su puesto, tiene que haber un hueco aquí para mis tres compañeras de siempre Alicia, Celia y Paula, que hemos recorrido juntos el viaje por la universidad y que nos hemos apoyado siempre codo con codo. Les estaré eternamente agradecido por todo lo que han hecho por mí, ya que creo que si no fuera por ellas, hoy puede que no estuviera en este punto de mi vida.

También quiero agradecerles a mis familiares y amigos todo su apoyo, ya que gracias a ellos hoy soy quien soy, mis tías y mi abuela, las que siempre cuidaron de mi hermana y de mí cuando éramos pequeños. Y por supuesto a mis abuelos tanto a Pedro y Sisca como a mi abuela Manoli, por todo lo que he aprendido de ellos y darme tantos momentos bonitos.

Por otro lado, quiero agradecer a mis tutoras, la constancia con la que me han ayudado a poder realizar este proyecto, así como la forma de enseñarme a investigar y priorizar mi aprendizaje. Para mí ha sido un gran placer trabajar con dos grandes matemáticas como vosotras y simplemente deciros, que me alegro mucho de haberos elegido como tutoras para esta última etapa que culmina mi carrera universitaria.

Ya para terminar, aunque pueda sonar algo egocéntrico, me gustaría agradecerme a mí mismo el haber llegado hasta aquí, que me he demostrado que se puede contra todo pronóstico. He trabajado mucho para llegar hasta aquí y puesto que no solemos agradecernos las cosas a nosotros mismos, creo que aunque sea en ocasiones hay que hacerlo. Gracias a mí mismo por no rendirme, por haber aguantado esos días y días de estudio sin parar, por haber aguantado aquellos días de no parar y no parar pese a estar en medio de una pandemia y que el trabajo no dejara de aumentar, por haber perseguido el sueño de vivir en Francia un tiempo, y por no dejar de soñar (como diría María Patiño).

Resumen

Tras el nacimiento de Bitcoin, se ha producido una profunda transformación en los mercados digitales y la bolsa, dando lugar a conceptos completamente novedosos. Esto ha despertado un notable interés tanto por parte de empresas como de investigadores, quienes buscan comprender los fenómenos resultantes de los acontecimientos en este mercado digital.

Motivado por la propuesta de mis tutoras, no dudé en adentrarme en este campo totalmente desconocido para mí. A pesar de la gran cantidad de información que hay disponible sobre esta criptomoneda, aún existen aspectos del funcionamiento de Bitcoin y las criptomonedas en general que son poco conocidos.

Este trabajo se divide en dos secciones principales. En la primera, se presentan los términos esenciales sobre Bitcoin y Blockchain para comprender el contexto en el que se desenvuelve el proyecto. Para ello, principalmente se utilizarán los artículos "Blockchain: A Graph Primer"[3] y "Forecasting Bitcoin Price with Graph Chainlets"[1]. La segunda sección es más práctica, y tiene como objetivo obtener y estudiar matrices que reflejen las ocurrencias de los distintos tipos de transacciones realizadas.

Gracias al análisis exhaustivo llevado a cabo en este proyecto sobre las matrices de ocurrencia antes mencionadas, hemos logrado obtener resultados innovadores en cuanto a la entropía persistente de las mismas. Hemos conseguido este avance al aplicar dos tipos distintos de filtraciones, permitiéndonos conocer en qué casos es útil normalizar las matrices y en cuales no a la hora de estudiar la homología persistente y entropía persistente.

Link al repositorio: https://github.com/joavilalv/TFG_Joan_Villalonga_Alvaro.git

Palabras clave: Bitcoin, Blockchain, transacción, criptodivisa, grafo, chainlet, homología persistente, entropía persistente.

Abstract

After the birth of Bitcoin, there has been a profound transformation in the digital markets and the stock market, giving rise to completely new concepts. This has aroused considerable interest from both companies and researchers, who seek to understand the phenomena resulting from events in this digital market.

Motivated by the proposal of my tutors, I did not hesitate to enter this field totally unknown to me. Despite the large amount of information that is available about this cryptocurrency, there are still aspects of the operation of Bitcoin and cryptocurrencies in general that are little known.

This work is divided into two main sections. In the first, the essential terms on Bitcoin and Blockchain are presented to understand the context in which the project is developed. For this, the articles "Blockchain: A Graph Primer" [3] and "Forecasting Bitcoin Price with Graph Chainlets" [1] will mainly be used. The second section is more practical, and its objective is to obtain and study matrices that reflect the occurrences of the different types of transactions carried out.

Thanks to the comprehensive analysis carried out in this project on the aforementioned occurrence matrices, we have achieved innovative results regarding their persistent entropy. We have accomplished this breakthrough by applying two different types of filtrations, enabling us to understand when it is beneficial to normalize the matrices and when it is not, when studying persistent homology and persistent entropy.

Link to the repository: https://github.com/joavilalv/TFG_Joan_Villalonga_Alvaro.git

Keywords: Bitcoin, Blockchain, transaction, cryptocurrency, graph, chainlet, persistent homology, persistent entropy.

Índice general

Índice de figuras

Índice de extractos de código

1. Introducción

La transformación de la bolsa tras el nacimiento de Bitcoin, así como el uso de esta moneda, ha dado lugar a debates cada vez mayores sobre el futuro de las criptomonedas.

Gracias a las criptodivisas como Bitcoin, podemos registrar transacciones en un libro mayor distribuido llamado Blockchain sin necesidad de una autoridad central. Esta tecnología ha ganado mucha popularidad gracias a que sustenta un sistema de pago descentralizado que además es público y anónimo permitiendo que cualquier persona pueda acceder y analizar las transacciones realizadas, así como representarlas en un grafo y buscar propiedades en el mismo.

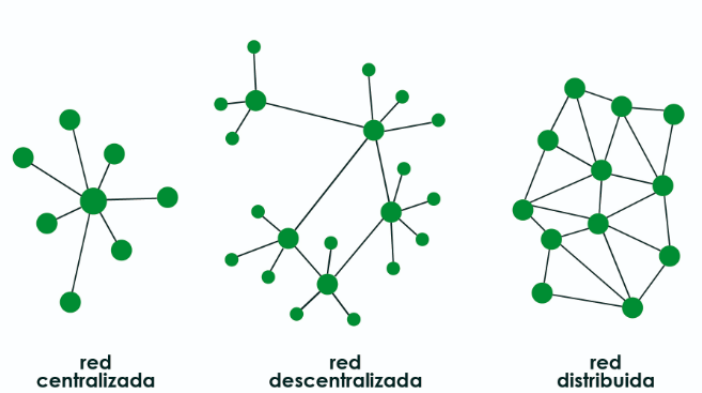

Figura 1.1: Tipos de redes

Aunque gran parte de la aceptación y uso de esta moneda se debe a este anonimato, en muchas ocasiones es utilizado por ciberatacantes, que secuestran los datos de una víctima cifrándolos y pidiendo un rescate por ellos. A este tipo de ciberataque se le denomina Ransoware y representa un reto para los investigadores que ven en el estudio de transacciones posibilidades de encontrar pistas que ayuden a contratacarlo [2].

Por otro lado, un grafo de transacciones de Bitcoin nos permite estudiar aspectos como la manera en la que afecta la forma de éste al precio de la criptomoneda en cuestión. En función de la cantidad de información que queramos incluir en el estudio, el grafo de transacciones resultante será más o menos complejo.

La idea inicial del proyecto era formar el grafo de transacciones inspirados por el modelo del artículo Forecasting Bitcoin Price with Graph Chainlets, el cual aporta gran cantidad de información que no se había tenido en cuenta en estudios anteriores. Pero una vez descubrimos la dificultad de montar dicho modelo de grafo, decidimos que la investigación siguiera su curso por un camino algo diferente. Nuestro estudio se centra en las matrices de ocurrencia de los diferentes tipos de transacciones formadas a partir de los datos de la red de Bitcoin. Concretamente, buscamos obtener resultados al aplicar herramientas de Análisis Topológico de Datos (TDA). Con estas herramientas

podremos analizar la aleatoriedad de los tipos de transacciones reflejados en el grafo y estudiar en qué o a qué afecta la ocurrencia de los mismos.

A pesar de la cantidad de estudios que se han realizado sobre la red de Bitcoin desde diferentes perspectivas (búsqueda de predicción de ataques de Ransomware, predicción del precio de esta y otras criptomonedas...), nunca se habían aplicado dichas herramientas para analizar dicha red.

La homología persistente será una de estas herramientas de TDA que nos permitirá extraer información topológica del grafo de transacciones revelando ciertas características de su funcionalidad [4]. Concretamente, gracias a la homología persistente, podremos detectar anomalías en la forma de los datos estudiando la distribución de las ocurrencias de los distintos tipos de transacciones que aparecen.

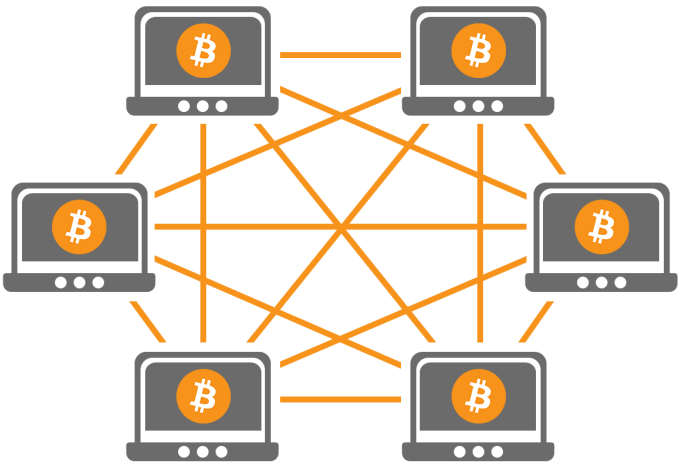

Figura 1.2: Red de Bitcoin

Al abordar el proyecto de mi trabajo de fin de grado, me motivaba mucho que la temática estuviera relacionada con el estudio de Bitcoin por diferentes razones. En primer lugar, las criptomonedas son un tema muy concurrido actualmente debido a la novedad de su tecnología. A diario, tanto inversores como grandes empresas deciden invertir considerables sumas de capital en proyectos que hacen uso de estas criptomonedas, con el objetivo de obtener más beneficios económicos. Por lo que una de mis motivaciones era comprender y analizar el funcionamiento de este sistema. Además, otro motivo por el que encontré esta propuesta sumamente interesante es debido al potencial que podría tener este trabajo para introducir a personas sin conocimientos tecnológicos ni financieros el tema de las criptomonedas. Por último, es importante destacar, que este proyecto se me hizo atractivo por su relación con las matemáticas y la teoría de grafos, ya que gracias a las asignaturas impartidas a lo largo de la carrera es un tema que me ha interesado desde el comienzo de la misma.

2. Objetivos y metodología

2.1. Introducción

La planificación de proyectos es una disciplina para afirmar cómo llevar a cabo un proyecto en un plazo determinado, por lo general con etapas definidas, y con recursos designados. [18]

El sentido de esta planificación reside en simplificar las tareas a realizar, estudiar las posibles desviaciones en la implementación de este proyecto y con la finalización del mismo, analizar el rendimiento de cada tarea realizada, comparando las estimaciones con el tiempo real empleado.

En este capítulo, hablaremos detalladamente del análisis y decisiones tomadas para llevar a cabo este proyecto. A pesar de ser un tema muy presente en la actualidad y haber mucha información, no sabemos con qué podemos encontrarnos cuando empecemos a trabajar, por lo que puede que se produzcan desviaciones a lo largo del desarrollo del mismo, pero aun así, los objetivos quedarán claros.

2.2. Objetivos

Como objetivo inicial de este proyecto tenemos la extracción de datos sobre las transacciones de Bitcoin de un archivo .dat (se trata de un tipo de archivos que contienen un formato genérico) con la ayuda del Trabajo de Fin de Grado de Fernando Claros Barrero [6]. La idea inicial fue que una vez tuviéramos los datos, procederíamos a la formación del grafo de transacciones, replicando la estructura utilizada en el artículo Forecasting Bitcoin Price with Graph Chainlet [2].

Ya con el grafo formado, pretendíamos buscar una solución no demasiado compleja computacionalmente, para analizar la estructura del mismo y estudiar la dinámica de las transacciones, la cual impacta en el precio de la criptomoneda.

Finalmente, los objetivos a los que nos ha llevado esta investigación han sido: la búsqueda de las matrices de ocurrencia de los Chainlets (concepto que se explicará en el capítulo 4), reflejando los tipos de transacciones que hay, así como su posterior procesamiento mediante diferentes bibliotecas de Python, de forma que, aplicando ciertas herramientas topológicas a estas matrices, podamos encontrar anomalías o elementos interesantes en estos resultados.

2.3. Metodología

Este proyecto, llevado a cabo como un trabajo de fin de grado en un campo técnico, se divide en dos componentes fundamentales que deben distinguirse. En primer

lugar, se llevó a cabo una investigación exhaustiva sobre el funcionamiento de Bitcoin y Blockchain. En segundo lugar, se realizó una parte más práctica que involucró el desarrollo de código utilizando librerías de Python preexistentes, con el propósito de aplicar técnicas topológicas a los datos y realizar un análisis exhaustivo de los resultados obtenidos.

Previamente a aplicar estas técnicas topológicas tuve el placer de asistir al curso "Applications of topology in online social networks and Blockchain networks" impartido por Cüneyt Gürcan Akçora (el autor del artículo sobre el que se basa este trabajo) en el que se explicaron entre otros los conceptos de topología que se utilizan en este proyecto. Dicho curso me permitió entender algunos términos que ya se habían nombrado en las reuniones de seguimiento, además de facilitar la comprensión de otras aplicaciones de la topología que desconocía.

2.4. Planificación

Podemos dividir este proyecto en las siguientes tareas principales:

1. Seguimiento y comunicación con las tutoras del proyecto.
2. Lectura de los artículos sobre los que basa en gran parte el proyecto.
3. Contacto con el autor principal de los artículos en los que se basa el proyecto y asistencia al curso impartido por él.
4. Investigación sobre Bitcoin, Blockchain y transacciones con criptomonedas.
5. Investigación sobre cómo obtener los datos.
6. Investigación sobre las librerías a usar para tratar los datos.
7. Extracción de datos y transformación al formato adecuado.
8. Producción del grafo de transacciones.
9. Búsqueda de patrones en el grafo.
10. Analizar los resultados obtenidos.
11. Redacción de la memoria del proyecto.
12. Realización y preparación de la presentación del proyecto.

En el cuadro 2.1 se muestra la estimación de horas que se piensa que se le va a dedicar a cada tarea así como las tareas que deben preceder a cada una de ellas.

2.5. Presupuesto

A la hora de desarrollar el proyecto, a pesar de la limitación de recursos, estos han cumplido su función y aunque a veces nos hemos encontrado con algún problema de procesamiento, finalmente, hemos podido solucionarlo. Esto ha sido posible ya que

Tarea	Tarea(s) predecesora(s)	Aproximación en horas
1	-	15
2	1	10
3	2	10
4	-	20
5	-	20
6	-	20
7	2,3,4,5,6	35
8	2,7	35
9	8	40
10	9	45
11	2,10	40
12	11	10

Cuadro 2.1: Tabla de planificación inicial. Se prevén alrededor de 300 horas en total.

pudimos descargarnos los datos ya procesados en vez de descargarlos en bruto, tener que transformarlos y desarrollar el proyecto a partir de ellos. Este proyecto se ha realizado en mi computadora personal, con un procesador intel core i7 que ha soportado las tareas realizadas.

De este modo, los costes que tendría el proyecto serían el sueldo de un programador o analista de datos para la investigación y el desarrollo del mismo, amortización del equipo utilizado y los gastos básicos de consumo (luz e internet), además del curso sobre Topología aplicada a las redes sociales y de Blockchain de Cüneyt Gürcan Akçora. El sueldo medio de un analista programador en España en el año 2023 es de 29600 € al año de media [16], suponiendo que al año se trabajan unas 40 horas semanales, nos queda que se le pagaría unos 15,41 € la hora. Escogiendo un 20 % como porcentaje de amortización anual (como se indica en la web de la Agencia Tributaria [21]) , a 10 meses que ha durado este proyecto y sabiendo que el equipo utilizado tiene un precio de 799 €, nos queda un gasto de:

$$\frac{799\ € * 0,2}{12} * 10 = 133,17\ €$$

Por último, eligiendo una tarifa mensual media de internet tendría un costo de 20 €mensuales además de un gasto medio de 52 €mensuales ya podríamos calcular el precio total del proyecto

$$300 \text{ horas x } 15,41\ € + 500\ € + 133,17\ € + (20\ € + 52\ €) * 10 = 5978,17\ €$$

De estos cálculos, deducimos que el proyecto completo tendría un coste total de 5978,17 € en total.

3. Blockchain: A Graph Primer

En este capítulo, explicaremos los términos y funcionamientos de Bitcoin y Blockchain apoyándonos en el artículo Blockchain: A Graph Primer [3], de esta forma tendremos el conocimiento necesario para adentrarnos en el proyecto.

3.1. Blockchain

Blockchain es una base de datos distribuida y segura por su diseño, fue propuesta por un grupo anónimo de matemáticos que trabajaban bajo el nombre de Satoshi Nakamoto en 2008. Consiste en un sistema de bloques de transacciones que se verifican y confirman sin una autoridad central. Esta tecnología se dio a conocer gracias a su uso en la criptodivisa Bitcoin, donde estas transacciones son de tipo financiero.

Figura 3.1: Blockchain

Blockchain permite a las empresas rastrear y analizar las transacciones financieras para usarlas en sus investigaciones. Gracias al historial de uso de Bitcoin, desde 2009 se han podido observar los usos y limitaciones de llevar la tecnología Blockchain a la práctica.

Gracias a las bases en las que se fundamenta Blockchain, podremos representar el historial de transacciones en un grafo de transacciones. En este grafo, tendremos por un lado nodos de dirección, en los que podremos observar las cantidades de la moneda de Bitcoin que pertenecen a cada una (en el caso de la Blockchain de Bitcoin). A su vez, habrá nodos de transacción que relacionarán nodos de dirección de los que gastan Bitcoins con nodos de dirección que los reciben.

3.2. Bitcoin

Bitcoin es una moneda virtual diseñada de forma segura para utilizarse como forma de pago sin necesidad de intervención por parte de una autoridad central [13]. Hasta la creación de esta criptodivisa, en varias ocasiones, otras monedas creían haber conseguido crear un método de pago factible pero finalmente no era así. Esto se debía a leyes y reglamentos que las obstaculizaban, pero sobre todo a deficiencias técnicas. Sin embargo, los creadores de Bitcoin se sirvieron de los errores de sus predecesores para aprender y poder crear esta moneda digital.

Otra opción distinta al modelo de Blockchain que se planteaba como alternativa a una autoridad central, era que todos los usuarios llevaran un registro de los saldos de todos los usuarios. El problema de esta idea era la velocidad y seguridad de transmisión de la información sobre las transacciones. Además, si hubiera cualquier problema en las redes, usuarios malintencionados podrían beneficiarse mintiendo sobre los saldos.

Mientras que el problema de la autoridad central seguía siendo un desafío en el campo de las monedas digitales, se encontró una solución en el ámbito no relacionado de la detección de spam en los correos electrónicos. Los proveedores de correo electrónico se enfrentaban al problema de recibir un gran número de correos electrónicos no deseados, aunque estos podían ser analizados y etiquetados como spam, el proceso consumía excesivos recursos del sistema. Para prevenir esto surgió HashCash, técnica que propone un acuerdo entre los remitentes de correo electrónico y los proveedores de este. Los remitentes de correo electrónico deben hacer un cálculo para cada uno y adjuntar una prueba del calculo a dicho correo. El cálculo está diseñado para consumir mucho tiempo, mientras que la verificación de la prueba es fácil. El proveedor de correo electrónico acepta aquellos que contengan pruebas válidas y descarta a todos los demás. Esta metodología se llama **Proof-of-Work**. Aunque no evita que un remitente pueda crear un correo electrónico no deseado, calcular su prueba y enviarlo hace que sea demasiado difícil enviar muchos correos de este tipo.

Figura 3.2: Proof-of-Work

Nakamoto utilizó esta idea para evitar que los mineros mintieran sobre los bloques a la hora de minarlos (puesto que se necesita gastar una cantidad considerable de energía para minar un bloque). Además, cada bloque contiene el hash del bloque anterior formando una cadena inmutable, de forma que si esta cambiara, el hash de los bloques posteriores al modificado sería distinto. Esta disposición de los bloques dio nombre a este sistema: **Blockchain** (cadena de bloques).

3.3. Construyendo los bloques de Blockchain

Una vez comprendidas las bases sobre Blockchain y Bitcoin, en esta sección se explicará cómo funciona la Blockchain de Bitcoin, así como la verificación de las transacciones y el minado de los bloques.

3.3.1. Direcciones

Una **dirección Blockchain** es una cadena única de 26 a 35 caracteres creada a partir de claves públicas generadas a partir del método de encriptación Elliptic Curve Digital Signature Algorithm (ECDSA), que como bien indica su nombre, el algoritmo se basa en la estructura algebraica de curvas elípticas sobre campos finitos. Este método de encriptación consigue el mismo nivel de seguridad que otros con claves mayores, pero en este caso utilizando claves de menor tamaño [12]. Bitcoin utiliza dos tipos de direcciones:

- Dirección de pago PubkeyHash que comienza con 1 como en:

  1Pudc88gyFynBVZccRJeYyEV7ZnjfXnfKn.

- Dirección de pago ScriptHash que comienza con 3 como en :

  3J4kn4QoYDj95S3fqajUzonFhLyjjfKjP3

El tipo de dirección mas utilizada o más común es PubkeyHash, que usa una sola clave privada para gastar los Bitcoins recibidos de una transacción. ScriptHash vino más tarde para permitir transacciones que necesitaran aprobación de múltiples usuarios.

Desde una perspectiva gráfica, el tipo de dirección no cambia nada, ambos pueden usarse en transacciones como direcciones de entrada o de salida. La diferencia es que ScriptHash permite a varios usuarios participar en decisiones de gasto. El titular de la dirección puede administrar los activos (cantidad de la moneda) adquiridos por transacciones ya que sólo él es conocedor de la clave privada asociada a la dirección. No podrá iniciarse una transacción sin la dirección del destinatario y un código de verificación de errores junto a la dirección (para evitar errores ortográficos en la misma).

3.3.2. Transacción

Una **transacción** es una trasferencia de activos entre direcciones. La tecnología Bitcoin permite realizar transacciones de múltiples direcciones a múltiples direcciones, aunque en la práctica las direcciones de entrada suelen pertenecer al mismo usuario y las direcciones de salida a diferentes usuarios. Para cada transacción es imprescindible aportar tres datos: el o los ID de las transacciones anteriores que realizó la dirección o direcciones de entrada, el índice de salida de la transacción previa y la cantidad a transferir. De esta forma, se firma la transferencia evitando que se altere la transacción, quedando reflejada la autorización del usuario para que la misma se lleve a cabo.

Cuando una transacción envía activos a una dirección, la red desconoce la clave pública de la dirección del receptor, sólo se muestra cuando los activos recibidos se gastan en otra transacción. Conociendo su clave pública cualquier usuario de la red podrá hacer un hash de la misma para verificar que es igual al hash de la dirección, comprobando así la legitimidad de sus transacciones.

En la figura 3.3 podemos ver cuatro tipos de transacciones diferentes. Cada uno de los números naranjas representan transacciones, los rectángulos verdes representan un **Satoshi** que es la unidad mínima de un Bitcoin (1 Bitcoin = 10^8 Satoshi) y los círculos representan los nodos de dirección. En la transacción 1 nos muestra una transacción 1/1 puesto que tiene una dirección de entrada y una dirección de salida. La transacción 2 es de tipo 2/2, la 3 es de tipo 1/2 y la 4 es de tipo 2/1. Cuando hay más de una entrada, realmente no se distingue de qué dirección viene qué cantidad del activo, por ejemplo en la transacción 2 no podemos saber si la cantidad de a_6 viene de a_2 o de a_3.

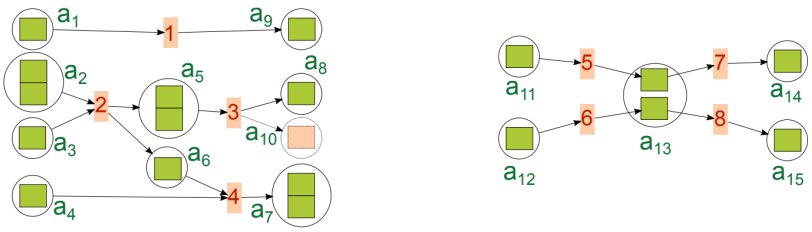

Figura 3.3: Tipos de transacciones

En la práctica, no se indica una tarifa de transacción como tal, este monto se calcula como la resta entre la cantidad total enviada por todos los usuarios en dicha transacción y la cantidad total que se retira de esa misma transacción. La tasa se envía a la dirección del minero que confirma la transacción (por ejemplo, en la figura 3.3 en la transacción 3, el rectángulo naranja sería la tasa que se le paga al minero). También puede darse el caso de realizar transacciones sin cargo, pero conforme mayor sea la tasa, más probable es que la transacción se confirme al ser más atractiva para los mineros que las elijen.

Una dirección puede recibir transferencias de múltiples transacciones y los resultados de esta pueden gastarse por separado. Pero la comunidad de Bitcoins tiene establecida la regla de gastar todos los activos en una sola transacción. Por ejemplo, en la figura 3.3, la dirección a_3 gasta 1 Satoshi en cada transacción y esto es posible porque cada uno venía de una transacción diferente. En cambio, si nos fijamos en a_5, recibe 2 Satoshi de la transacción 2, pero tiene que gastarlos en la transacción 3. Si sólo envía 1 Satoshi, la cantidad no gastada será para el minero. En la práctica, si un usuario tiene que gastar una fracción del monto recibido, envía el saldo restante a otra dirección suya, o bien crea una dirección nueva a la que envía el dinero.

3.3.3. Verificación y confirmación

Cuando hablamos de verificar el pago nos referimos a crear un mecanismo que confirme que la dirección emisora tiene saldo suficiente para realizar el pago y también corrobore que el dueño de la dirección quiere gastar la cantidad indicada. Para ello,

el usuario presenta su clave pública (demostrando así que la dirección le pertenece) y firma con su clave privada que quiere transferir dicha cantidad.

La transacción también almacena como entrada el número de la transacción anterior de donde viene la cantidad (por ejemplo, en la figura 3.3), la transacción 8 almacena que su entrada proviene de la transacción 6.

En el caso ideal, tan pronto como se gasta una cantidad de la moneda, todos verían la transacción y registrarían los nuevos saldos. Si no fuera así, el emisor de la transacción podría gastar unos mismos Bitcoins para múltiples transacciones (conocido como problema del doble gasto).

Para solucionar este problema, Nakamoto propuso utilizar un libro contable público y distribuido que contuviera bloques con identificadores y marca de tiempo así como las transacciones verificadas. Cada bloque lleva un hash sobre el bloque anterior, para facilitar el seguimiento de la cadena de bloques en el tiempo.

Los bloques tienen algunas limitaciones como en el caso de las transferencias muy grandes ya que podrían llegar a obstruir la red. Es por eso que en Bitcoin, el tamaño de bloque está limitado a 1 MB.

La creación de bloques está limitada a un bloque cada diez minutos a través del mecanismo antes explicado de Proof-of-Work. Esto se debe a que es muy difícil de crear cada bloque, pero fácil de firmar una vez creado. Esta prueba de trabajo requiere encontrar un número de 32 bits conocido como *nonce* mediante prueba y error. La dificultad se va regulando en función del número de mineros y la capacidad de los mismos de encontrar el valor del nonce correcto. Los usuarios que trabajan para encontrar bloques confirmados se denominan **mineros**.

El minero recibe una lista de transacciones verificadas y las incluye en el bloque. Normalmente, se habrán verificado aquellas cuya tasa sea mayor. Primero, se concatenan unos datos concretos de cada bloque, como el hash del bloque anterior y un hash correspondiente a cada una de las transacciones que pertenecen a este bloque. La prueba consiste en encontrar este número *nonce* de tal forma que si hacemos el hash SHA-256 del bloque, el resultado comience por un número determinado de ceros o de forma equivalente que sea menor que un número fijado llamado hash target, entonces, se podrá decir que se ha minado el bloque.

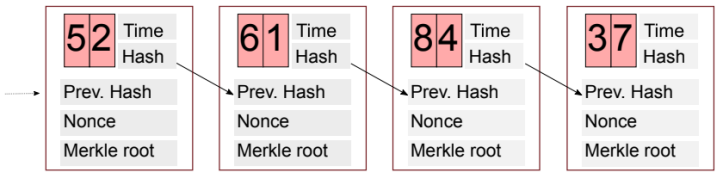

Figura 3.4: Estructura de Blockchain

Una vez se mina un bloque B, el minero lo envía a sus vecinos en la red y espera a que se incluya el bloque a la cadena. Conforme la información de B se propaga en la red, los usuarios actualizan sus cadenas, y añaden a B como el último bloque. En este punto, el resto de mineros deberán detener sus cálculos, cambiar el hash del bloque anterior con el de B y reanudar la minería.

Una vez entendido el funcionamiento de Blockchain, podemos plantearnos la siguiente pregunta. Si en el bloque necesitamos el hash del bloque anterior, ¿cómo es el primer bloque de la cadena? Pues bien, este bloque se conoce como bloque Génesis y fue extraído por Nakamoto.

Otra duda que puede surgirnos es, ¿puede haber bifurcaciones en la cadena de bloques? La respuesta es que sí, los usuarios pueden ser notificados de dos bloques alternativos que se bifurcan de un bloque anterior, y los mineros deberán elegir un bloque sobre el que construir y continuar la cadena. Y, ¿cómo se soluciona esto? Pues la cadena de bloques acaba eligiendo como cadena principal a la que vuelva a minar bloques más rápido. Es decir, que si por una rama, se vuelve a minar un bloque antes que en la otra, esta pasará a ser la cadena principal.

Gracias a la prueba de trabajo, los desarrolladores de Bitcoin se aseguran de que si un usuario mal intencionado crea una bifurcación para reemplazar la cadena de bloques principal deberá extraer nuevos bloques más rápido que todos los demás mineros de la red. Existe un tipo de ataques llamados **ataques de eclipse**, en el que a un usuario se le notifica una bifurcación diferente a la de la cadena principal con fines de estafarle.

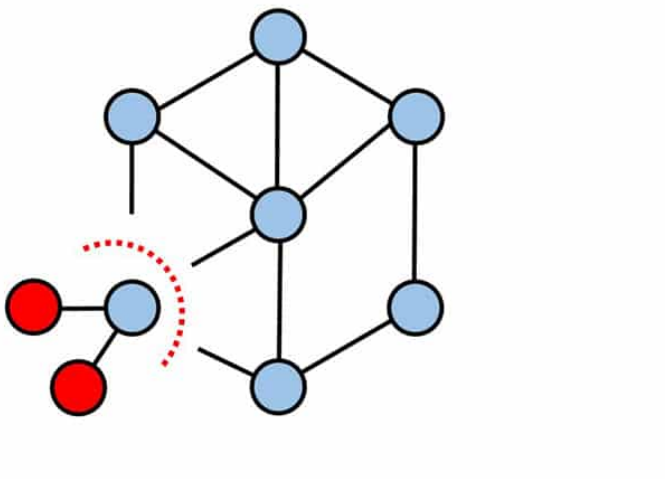

Figura 3.5: Ataques de eclipse

Pese a que las transacciones están protegidas por los mecanismos anteriormente explicados, hay usuarios maliciosos que han encontrado múltiples formas de estafar wallets y a usuarios. Aunque las transacciones son inmutables, los identificadores de las transacciones no lo son, por lo que se podrían modificar. De forma que si un usuario abre un wallet, compra Bitcoins y los envía a una dirección, tan pronto como la transacción se publica, el propio usuario podría modificar la identificación y enviarla de nuevo a la red, y por tanto habría dos copias de esta transacción. Gracias a la estructura de la red, no deberemos preocuparnos por esto, puesto que al estar duplicada, los mineros rechazarían la segunda transacción, ya que esos Bitcoins ya se han gastado.

Además de las tasas de transacción, Bitcoin crea una recompensa minera por cada bloque minado, esto se conoce como transacción de base de monedas y suele ser

la primera transacción en un bloque.

Un último aspecto que debe quedar bien claro, es que aunque una transacción se haya añadido a un bloque, no es definitiva hasta que el bloque se haya minado y se confirme que este pertenece a la rama principal. Ya que puede ocurrir que el bloque se mine, pero el bloque quede huérfano porque se haya producido una bifurcación y no se siga por la rama a la que pertenece. Es por ello que se recomienda esperar a que se hayan minado al menos 5 bloques después de este dentro de la misma rama, para confirmar que la transacción se ha completado correctamente.

3.4. Ejemplo

Para la mejor comprensión de cómo funciona Blockchain, a continuación presentaremos un ejemplo de forma que dicho funcionamiento quede claro. Vamos a suponer que tenemos un bloque al que le vamos a poner un índice $n-1$, un bloque n y finalmente un bloque $n+1$.

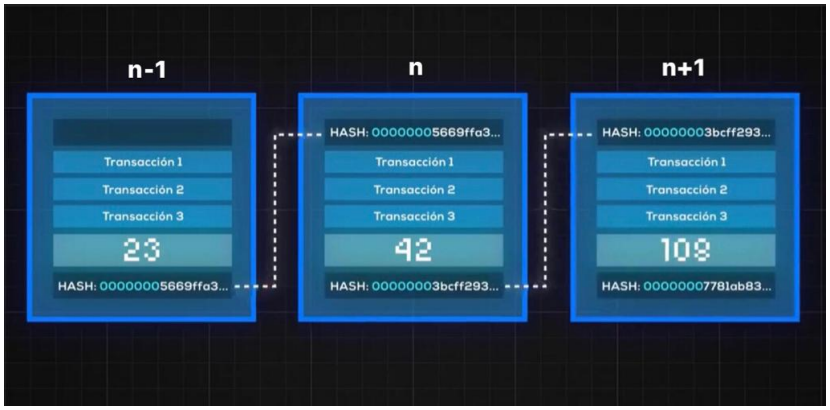

Figura 3.6: Ejemplo de Blockchain. Imagen extraída de [9].

Cada uno de estos bloques contiene sus transacciones verificadas (las cuales tienen cada una su ID que corresponde con el hash generado a partir de los inputs, outputs y cantidad de la moneda transferida de esa transacción), el nonce correspondiente al bloque una vez se haya minado y el hash del bloque anterior.

Si observamos el bloque n, podemos ver que está formado por: el hash del bloque $n-1$, el conjunto de transacciones verificadas y el nonce de dicho bloque que en este caso es el número 42. Si nos fijamos en el nonce del bloque $n-1$ es el 23 ya que si ciframos las transacciones de este bloque, junto con el resto de cabeceras, obtendremos un hash con 7 ceros ya que esta es la cantidad de ceros estipulada en este caso (00000003bcff293...). Este número es el que hace que al cifrarlo todo mediante SHA-256, el hash de este (el bloque n), empiece por una cantidad concreta de ceros.

De esta forma, si algún usuario mal intencionado quisiera modificar alguno de los bloques, por ejemplo, en este caso, el bloque $n-1$, el hash de los bloques n y $n+1$ sería totalmente diferente. Este tipo de ataque se llama **Sybil Atack**, y gracias a la prueba de esfuerzo, evitamos que pueda llegar a ser fructífero ya que el atacante necesitaría

minar el bloque modificado y todos los siguientes a una velocidad mucho mayor que el resto de usuarios de la red de Blockchain, cosa que es prácticamente imposible.

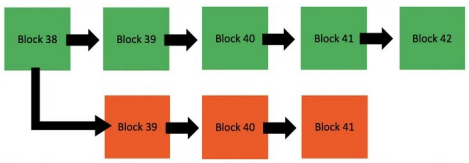

Figura 3.7: Ejemplo de Sybil Atack

3.5. Conclusión

De esta sección podemos concluir que cada transacción tiene unos nodos de entrada (inputs) y unos nodos de salida (outputs). Además, debe tenerse en cuenta que los inputs en realidad son nodos de salida de una transacción anterior. De esta forma, a la hora de verificar que el emisor de la transacción disponga de saldo suficiente, lo único que tendrá que hacer es comprobar las transacciones anteriores.

Otro aspecto importante a tener en cuenta, es que aunque una transacción se haya añadido a un bloque, no es definitiva hasta que se confirme que este bloque pertenece a la rama principal. Ya que puede ocurrir que el bloque se mine, pero quede huérfano porque se haya producido una bifurcación y este bloque no pertenezca a la rama principal. Es por ello que se recomienda esperar a que se hayan minado al menos 5 bloques después de este dentro de la misma rama, para confirmar que la transacción se ha completado correctamente.

Un bloque está formado por el hash del bloque anterior, el conjunto de sus transacciones y un número llamado nonce. Este número es el resultado de una prueba de trabajo o Proof-of-Work que se deberá llevar a cabo para minar el bloque. El minero habrá minado el bloque cuando por fuerza bruta, es decir, probando, haya encontrado el número nonce, tal que al obtener el hash (con codificación SHA-256) este empiece por una cantidad determinada de ceros. Dicha prueba de trabajo se pide para que la red distribuida funcione correctamente y de forma segura.

Una vez el minero haya minado el bloque, esta información se enviará al resto de usuarios de la red, para que estén actualizados y si estaban tratando de minar el bloque, paren y comiencen con el siguiente.

4. Graph chainlets

En este capítulo hablaremos sobre el artículo Forecasting Bitcoin Price with Graph Chainlets [1], cómo se modela en forma de grafo la blockchain de Bitcoin y qué son, cómo se definen y para qué sirven los chainlets.

4.1. Introducción a los chainlets

La criptomoneda Bitcoin ha provocado un gran interés gracias a su tecnología de blockchain, esto se debe a que es bastante interesante el hecho de que todos los participantes tienen un libro contable, en el que pueden verificar todas las transacciones que se realizan con Bitcoin. Gracias a este libro contable distribuido tendremos la oportunidad de analizar el grafo de transacciones.

Además, este grafo puede utilizarse para otros ámbitos, por ejemplo, estudiar si la estructura del grafo afecta al precio de una divisa como Bitcoin o cualquier otra. Por lo tanto, observando los subgrafos que se repiten con más o menos frecuencia de lo esperado, podemos saber la importancia que tienen dentro del grafo de transacciones.

Tanto la búsqueda de estos patrones como el estudio de los mismos en las redes financieras como la de Bitcoin, están aún en proceso. Por ello, en los siguientes puntos se desarrollarán técnicas para entender correctamente cómo se forma el grafo de transacciones, así como, qué son los chainlet y su impacto en la predicción de precios.

4.2. Aplicaciones del estudio de Blockchain mediante chainlets

Desde 2008 Bitcoin ha sido la aplicación más directa, destacada y estudiada de Blockchain. Los primeros estudios estaban enfocados en analizar el grafo de transacciones en búsqueda de actividades ilegales como el blanqueamiento de dinero o el chantaje con las diferentes monedas que hay.

En otras ocasiones la red de Bitcoin también ha sido estudiada con otros propósitos. Por ejemplo, en 2016 tras el análisis de las centralidades en el artículo Exploring the Bitcoin Network [7], los autores del artículo Analyzing the Bitcoin Network: The First Four Years [17] descubrieron que en ese año esta red podía considerarse una red sin escala, es decir, que un grupo de nodos están involucrados en un gran número de transacciones, mientras que la gran mayoría apenas lo están. Estos análisis se centraban en las características globales del grafo, pero el análisis que se plantea en este artículo pretende entender las estructuras topológicas de Bitcoin y su papel en la determinación de los precios de las monedas.

En general, los estudios más recientes muestran la utilidad de analizar el grafo de transacciones en búsqueda de características que ayuden a la predicción de precios. A veces, se analiza el coeficiente de agrupamiento (midiendo los nodos que tienden a crear grupos muy unidos caracterizados por una densidad relativamente alta de vínculos) y saldo promedio. Por otro lado, los chainlets ayudan a la puntualización de las características globales antes mencionadas, calculándolas sólo en ciertos subgrafos.

4.3. Metodología

En este artículo [1], el grafo de Bitcoin tiene tres componentes principales: direcciones, transacciones y bloques. En la siguiente figura podemos observar la representación de una red de este tipo para 4 transacciones y en las que se ven involucradas 13 direcciones.

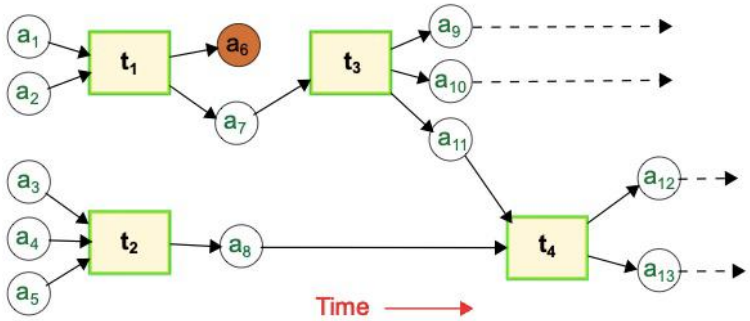

Figura 4.1: Representación gráfica de transacciones de la red Bitcoin

En dicha representación gráfica, las direcciones están representadas por círculos, las transacciones por rectángulos y las flechas indican una transferencia de criptomonedas. Como observación, añadimos que las monedas en la dirección a_6 están sin gastar.

En el artículo se remarca que los datos que se manejan provienen del software oficial de Bitcoin (instalaron la wallet central de Bitcoin y esta descargó todo el historial de Bitcoin desde 2009 hasta 2018). Tras la obtención de los datos analizaron la blockchain de Bitcoin y extrajeron los bloques, transacciones y direcciones.

El grafo queda modelado como una red heterogénea con dos tipos de nodos: transacciones y direcciones. Gracias a esto, podremos ver la red de Bitcoin representada como un grafo dirigido $G = (V, E, B)$ donde V es el conjunto de vértices, $E \subseteq V \times V$ es el conjunto de aristas y $B = \{\text{dirección, transacción}\}$ que hace referencia al conjunto de los tipos de vértices. Para cada vértice $u \in V$, existe un tipo de vértice perteneciente a B, y se verifica que para cada par de vértices u y v, conectados por una arista $e \in E$ el tipo de vértice de u, siempre será diferente al de v. Cada arista $e \in E$, representa una transacción de la moneda entre un nodo de dirección y un nodo de transacción. En este modelado de grafo, todas las transacciones tienen el mismo peso, eso quiere decir, que todos los tipos de nodos valen lo mismo ($\|B\| = 1$) y por lo tanto, no se tiene en cuenta la cantidad de moneda que se está transfiriendo. Este artículo se centra en el caso en

el que cada nodo de dirección está vinculado a través de un nodo de transacción a otro nodo de dirección.

Existen tres reglas que dan forma al grafo de Bitcoin y son las siguientes:

- En una sola transacción pueden entrar monedas de dos direcciones diferentes (como en la transacción t_4 de la figura 4.1)
- En una transacción no se registra detalladamente las entradas y salidas, por ejemplo, en la salida en dirección a_6 puede venir de a_1 o de a_2.
- Las monedas de transacciones con varias direcciones de entrada se pueden gastar por separado, pero las que salen de una transacción deben gastarse en una sola (podemos fusionar a_1 y a_2, pero no podemos gastar a_7 en transacciones diferentes o transferir parte de las monedas que acaba de recibir y quedarse con el resto).

Una pequeña observación, es que se desaconseja reutilizar nodos de dirección, ya que es fácil y gratuitamente pueden crear nuevas pero aún así hay usuarios que las reutilizan.

Los bloques aparecen por orden temporal, y cada transacción (incluyendo sus nodos de entrada y salida) queda representada por un subgrafo en la red de bitcoin, al que a partir de ahora nos referiremos como **chainlet**.

4.3.1. Graph Chainlets

Los k-chainlets son estructuras que se utilizan para evaluar la estructura topológica del grafo de Bitcoin (k hace referencia al número de nodos de transacción que van a estar incluidos en el chainlet, en este caso la k dentrá como valor 1).

Denotaremos las cadenas como $C_{x \to y}$ siendo x el número de entradas e y el número de salidas. En el caso particular en el que tenemos un número mayor de entradas que de salidas (es decir, $x > y$) lo llamaremos **merge chainlets**. Los otros dos casos que se pueden dar son que el número de entradas sea igual al número de salidas ($x = y$), llamado **transition chainlets** y por último si el número de entradas es menor que el número de salidas ($x < y$) se llama **split chainlets**. Nos referiremos a estos tipos de chainlets como **agregate chainlets** a partir de ahora. Podemos ver en la siguiente figura unos ejemplos.

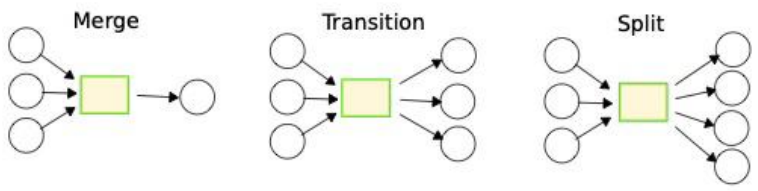

Figura 4.2: Tipos de chainlets. Merge ($C_{3 \to 1}$), Transition ($C_{3 \to 3}$), Split ($C_{3 \to 4}$).

4.3.2. Clustering Chainlets

El protocolo de Bitcoin restringe el número de nodos de dirección de entrada y salida en una transacción al poner el límite de 1MB a cada bloque, pero aún así puede llegar a haber miles de salidas o entradas, dando como resultado infinidad de chainlets posibles (como por ejemplo, $C_{1900\to200}$).

Para modelar el grafo de Bitcoin utilizaremos una representación matricial para un periodo de tiempo determinado como puede ser un día.

Para hablar de este modelo matricial haremos referencia al número de fila con la variable i y al número de columna con la variable j siendo ambas menores que n Así la matriz de ocurrencia O de orden n x n vendrá dada por:

$$O[i,j] = \begin{cases} \#\ C_{i\to j} & \text{si } i < n \text{ y } j < n, \\ \sum_{z=n}^{\infty} \#\ C_{i\to z} & \text{si } i < n \text{ y } j = n, \\ \sum_{y=n}^{\infty} \#\ C_{y\to j} & \text{si } i = n \text{ y } j < n, \\ \sum_{y=n}^{\infty} \sum_{z=n}^{\infty} \#\ C_{y\to z} & \text{si } i = n \text{ y } j = n. \end{cases}$$

Según este modelo, el valor de n lo elegiremos de tal forma que queden representados el 95 % de los tipos de chainlet en la matriz. Deberemos elegir hasta qué número de entradas o salidas se representarán de forma más específica para que la matriz no quede excesivamente grande y de forma que no haya partes de la matriz que no den información ninguna, o al menos minimizarlas todo lo posible.

Una vez elegido el número que dará dimensión a la matriz. Ésta quedará definida de la siguiente forma:

- Si i y j son menores que n, el elemento [i,j] será el número de chainlets de la forma $C_{i\to j}$)
- Si nos referimos a los elementos de la última columna (salvo el elemento [n,n]), el elemento [i,n] será igual al sumatorio de todos los chainlets con i entradas cuyo número de salidas sea mayor que n.
- Si nos referimos a los elementos de la última fila (salvo el elemento [n,n]), el elemento [n,j] será igual a la suma de todos los chainlets con j salidas cuyo número de entradas sea mayor que n.
- Y el último caso que queda es el elemento [n,n], que será igual al número de chainlets con un número de entradas y salidas mayor que n.

5. Herramientas de topología utilizadas en este proyecto

Una vez comprendidos los términos de Bitcoin, se explicarán las herramientas de topología que se han utilizado a lo largo de este proyecto, tales como: el diagrama de persistencia, la homología persistente y la entropía persistente, para que se entienda de forma correcta la implementación y desarrollo del mismo.

5.1. Homología persistente

La **homología persistente** ha adquirido una gran relevancia en el estudio de los datos. Esto se debe al crecimiento exponencial en la extracción y análisis de datos en la última década. Dicha herramienta es una forma de aplicar topología algebraica ayudándonos a estimar los grupos de homología. Es decir, nos ayudará a entender las estructuras matemáticas que dan forma a los datos que queremos analizar.

La homología persistente estudia la evolución de los agujeros k-dimensionales mediante una filtración concreta (las filtraciones son una colección de objetos que representan la forma de los datos que se están analizando). Gracias a ésta, podremos revelar patrones y características de conjuntos de datos complejos, comprendiendo mejor tanto su estructura como sus propiedades.

5.2. Diagrama de persistencia y barcodes

Los **diagramas de persistencia** fueron introducidos con el nacimiento de la homología persistente, siendo hasta hoy, la forma más habitual de representar los resultados de esta. Un diagrama de persistencia es una forma de representar en dos dimensiones la evolución de la homología persistente aplicada a un conjunto de datos [8].

Los **barcodes** son colecciones de segmentos horizontales que comienzan con la aparición del primer nivel de filtración que apliquemos y terminan cuando este desaparece. Realmente, representan la misma información que los diagramas de persistencia sólo que de una forma diferente. [14]

Como se comenta en el párrafo anterior, el diagrama de persistencia y el barcode son dos formas de representar la misma información. Pero, ¿cómo pasamos de un barcode a un diagrama de persistencia? Pues bien, en los diagramas de persistencia cada segmento del barcode está representado por un punto, en el eje x representaremos el nacimiento del segmento y en el eje y la muerte del mismo. Para que se comprenda mejor procederemos de pasar de una representación a otra, se explicará un ejemplo partiendo de un barcode.

Suponiendo que queremos obtener el diagrama de persistencia correspondiente al barcode de la figura 5.1, deberemos representar cada barra mediante un punto en el diagrama de persistencia. Como se ha indicado anteriormente, para hacer esto, deberemos representar cada barra gracias a las coordenadas de su nacimiento y muerte. El nacimiento de la barra nos dará la coordenada del eje x en el plano, y su muerte la coordenada del eje y. De esta forma, a partir del barcode de la figura 5.1 nos dará el diagrama de persistencia representado en la figura 5.2 (se pueden observar las correspondencias entre figuras gracias a los colores de los puntos y las barras).

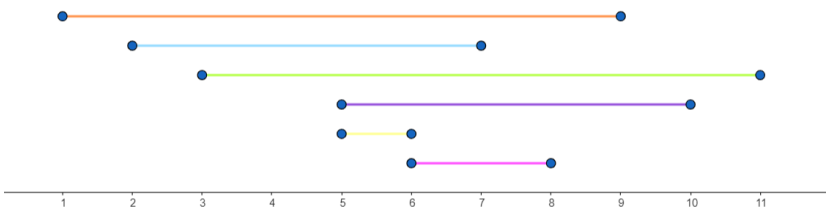

Figura 5.1: Ejemplo de barcode.

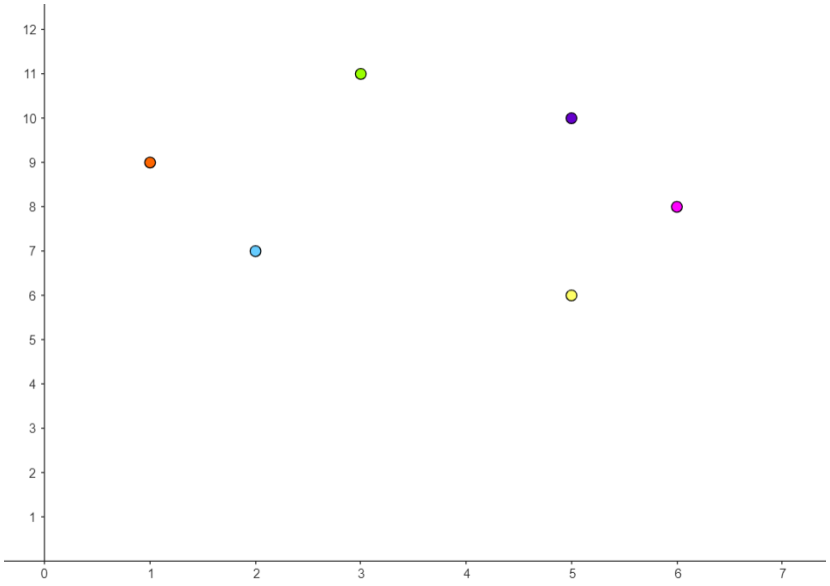

Figura 5.2: Ejemplo de diagrama de persistencia.

5.3. Entropía persistente

La **entropía persistente** es una herramienta que nos permite medir la similitud entre dos barcodes, permitiéndonos observar la heterogeneidad de las barras de un diagrama de persistencia. Cada barra representa una característica topológica, como puede ser un agujero en la representación del conjunto de datos. La longitud de la barra representa la duración de esa característica, cuanto más larga la barra, más persistente será la característica en cuestión.

La entropía de las barras que obtenemos al calcular la homología persistente, será lo que nos ayudará a medir la cantidad de desorden o aleatoriedad que hay en un sistema y cuánto tiempo puede persistir ese desorden.

Dadas una filtración $F = \{K(t) \,|\, t \in R\}$ y su diagrama de persistencia $dgm(F) = \{(x_i, y_i)\}_{i=1}^{n}$, la fórmula de la entropía persistente es la siguiente [5]:

$$E(F) = -\sum_{i=1}^{n} p_i log(p_i) \text{ donde } p_i = \frac{y_i - x_i}{S_L}, \text{ y } S_L = \sum_{j=1}^{n} y_j - x_j$$

5.4. Filtración Vietoris-Rips

Para explicar como funciona esta filtración, partiremos del ejemplo de una matriz $n \times n$, en la que sus elementos tendrán como número de fila i, número de columna j y el elemento contenido en esas coordenadas de la matriz será denotado como m_{ij}. Cada elemento de la matriz representará un punto en el espacio de coordenadas (i, j, m_{ij}), y gracias a esto podremos calcular su homología persistente y de ahí su entropía. La filtración Vietoris-Rips tiene en cuenta cada punto de forma independiente, representando el número de fila en el eje y, el número de columna en el eje x y cada valor de la matriz quedará representado como una altura en el eje z.

De esta forma tendremos cada uno de los puntos de la matriz representados de forma aislada, y lo que hace esta filtración es medir cuánto debe engrosar esos puntos hasta que se toquen, creando así componentes conexas (ya que al principio habrá tantas componentes conexas como elementos haya en la matriz, es decir n^2 en nuestro caso).

Para verlo, pondremos un ejemplo sencillo:

$$\begin{pmatrix} 1 & 1 & 7 \\ 1 & 2 & 6 \\ 2 & 1 & 1 \end{pmatrix}$$

En el ejemplo de la matriz que se propone, si queremos calcular su homología persistente con la filtración Vietoris-Rips y posteriormente su entropía, deberemos transformarla al formato que comentábamos antes, de forma que cada elemento de la matriz defina un punto con sus tres coordenadas:

$$\begin{pmatrix} (0,0,1) & (0,1,1) & (0,2,7) \\ (1,0,1) & (1,1,2) & (1,2,6) \\ (2,0,1) & (2,1,1) & (2,2,1) \end{pmatrix}$$

De forma gráfica, estos puntos quedarían representados como en la figura 5.3.

En la representación, el punto A simboliza el $(0,0,1)$, el punto B $(0,1,1)$ y así sucesivamente. Podemos observar que al principio tendremos 9 componentes conexas, pero si vamos engrosando los puntos, acabaremos teniendo como resultado dos componentes conexas, puesto que los puntos A, B, D, E, G, H e I están relativamente cerca (su altura es la misma o varía muy poco) y por lo tanto habrá que engrosarlos muy poco para que se unan en una sola componente. Por otro lado, los puntos F y C formarán

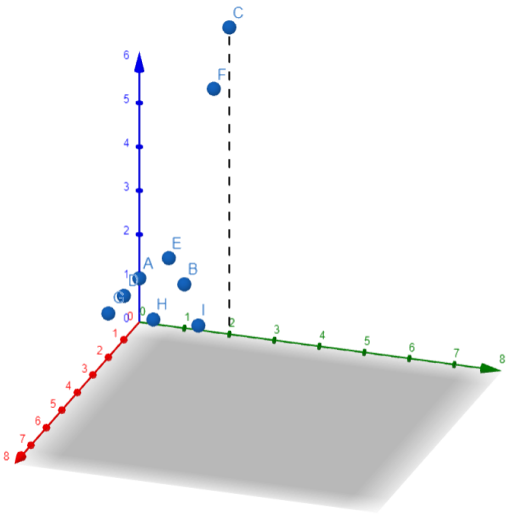

Figura 5.3: Ejemplo de homología persistente con Vietoris-Rips.

otra componente conexa, y estos habrá que engrosarlos más para que lleguen a formar una única con el resto de puntos.

Esto quiere decir, que la matriz del ejemplo es más heterogénea que una en la que todos sus elementos fueran iguales o estuvieran cercanos, puesto que sus alturas serían iguales, y no haría falta engrosar mucho los puntos para poder formar una única componente conexa.

Como hemos podido observar, gracias a esta filtración, podremos distinguir la diferencia de altura entre puntos, detectando anomalías en caso de que las hubiera.

5.5. Filtración Lower-Star

Puesto que una imagen no deja de ser una matriz con valores entre 0 y 255, decidimos hacer las mismas pruebas con Lower-Star, una de las filtraciones más utilizadas en tratamiento de imágenes.

En este caso, los puntos se representan en el espacio de la misma forma, pero esta vez, en lugar de ser puntos independientes, forman parte de una malla cuadrada de vértices $\{(i,j,0)\}_{i=1}^{n}$ que sube en cada coordenada a la altura que indica el elemento m_{ij} en la posición (i,j) de la matriz considerada. Lo que significa que al principio, si hacemos cortes en paralelo al eje z, en el caso de que los elementos de la matriz sean bastante parecidos, uniendo los puntos que queden por debajo de dicho corte, tendremos una sola componente conexa, y conforme vayamos subiendo, si tenemos puntos con alturas diferentes, podremos observar que cada vez hay más (estas componentes se deben a los puntos con mayor altura, formando picos en la malla). Con esta filtración, cuantas más componentes tengamos, más variedad de alturas y por tanto podremos detectar anomalías en los puntos más bajos.

Para aclarar como funciona esta filtración, continuaremos con otro ejemplo, esta

vez, la matriz será de 4×4:

$$\begin{pmatrix} 3 & 3 & 3 & 0 \\ 2 & 1 & 3 & 0 \\ 4 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Los puntos en el espacio quedarían representados en la figura 5.4 (la unión de los puntos con el elemento que crea un cráter, se ha representado de color morado para que se diferencie mejor).

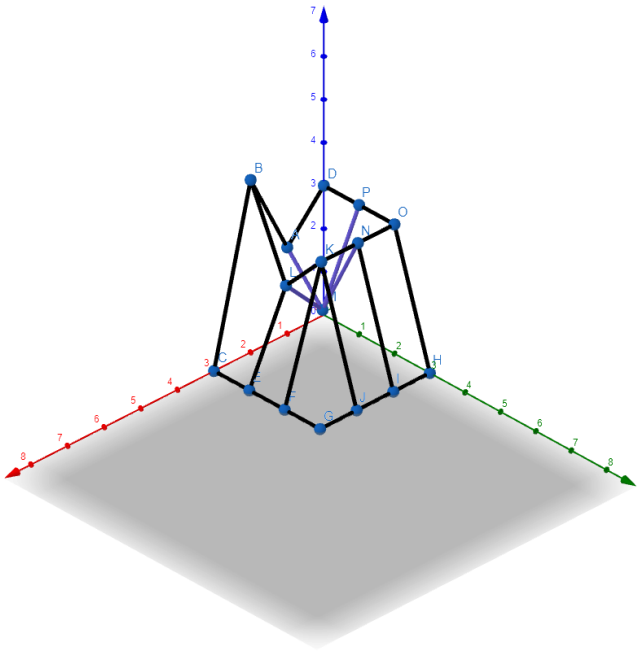

Figura 5.4: Ejemplo de Lower-Star

Para calcular la homología persistente, esta filtración, crea unas barras que representan la persistencia de las componentes conexas. En este caso, para estudiar la persistencia de las componentes conexas, realizaremos cortes en el eje z. De esta forma, podremos ver que comenzamos teniendo una sola componente conexa en z=0 (como se muestra en la figura 5.5).

Para continuar, tendremos que en z=1 nace una nueva componente conexa (representada de color rojo en la figura 5.6).

Y así continuaría la filtración, de tal forma que en z=2 al tener en cuenta el punto A y L se unirían en una sóla componente, por lo que moriría la más joven (figura 5.7).

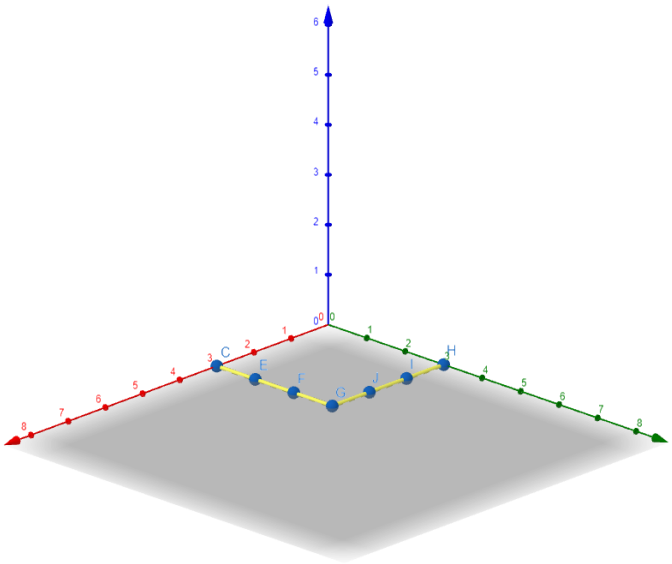

Figura 5.5: Ejemplo de Lower-Star: Primera componente conexa

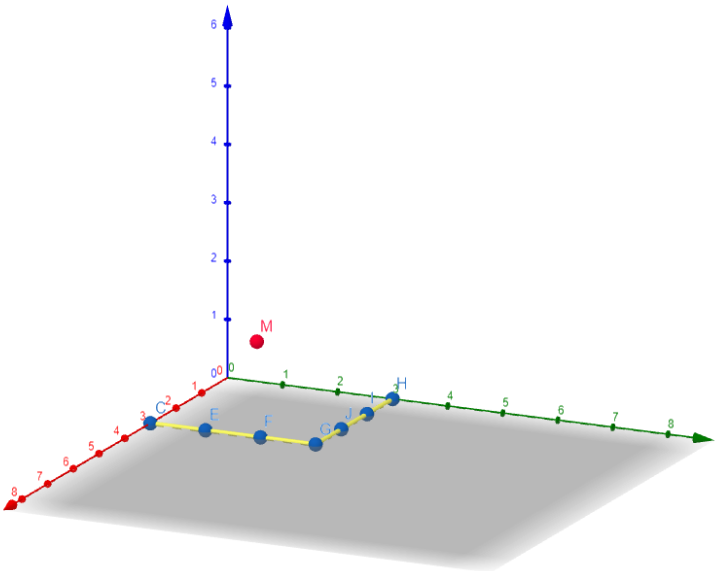

Figura 5.6: Ejemplo de Lower-Star: Dos primeras componentes conexas

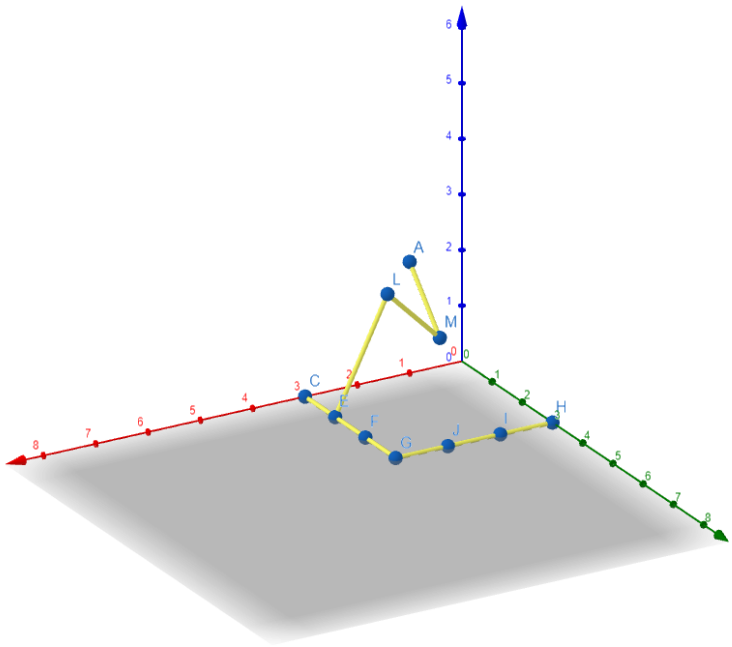

Figura 5.7: Ejemplo de Lower-Star: Una sola componente conexa

A partir de esta altura, por mucho que subamos en el eje z, tendremos una sola componente conexa, por lo que ya sólo quedará viva una barra en el diagrama de persistencia representado en la figura 5.8.

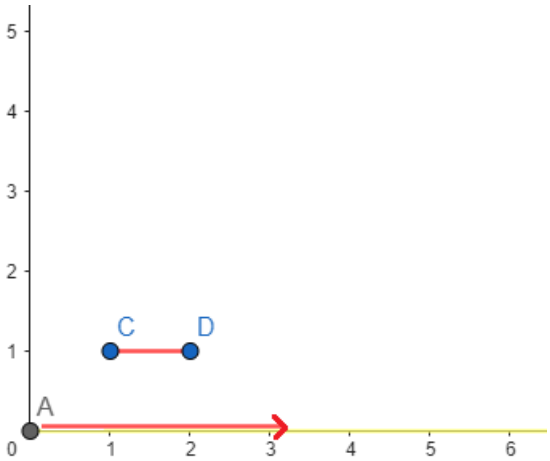

Figura 5.8: Barcode del ejemplo

Gracias a esta filtración, podremos obtener información sobre la entropía de los datos más bajos en el eje z. Aplicado al contexto de las matrices de ocurrencia de Bitcoin, esto nos permitirá detectar los tipos de transacciones que menos se repiten en relación a sus similares.

5.6. Filtración Upper-Star

Tras aplicar la filtración Lower-Star, empezamos a pensar en que al igual que queríamos información sobre la entropía en los puntos más bajos de cada matriz, no queríamos perder información sobre los máximos/picos de cada una de ellas. Es por esto, que pensamos en cambiarle el signo a cada elemento de la matriz, de tal forma que esta vez los máximos, serían los elementos más cercanos al eje y por tanto de los que obtendríamos información.

Gracias a esto, las barras de persistencia se van a obtener en función de los picos más altos (antes de cambiarle el signo a los elementos de la matriz). Para verlo más claramente, se continuará con el ejemplo anterior. En el siguiente gráfico, se representan los elementos de la matriz una vez ya se les ha cambiado el signo (figura 5.9).

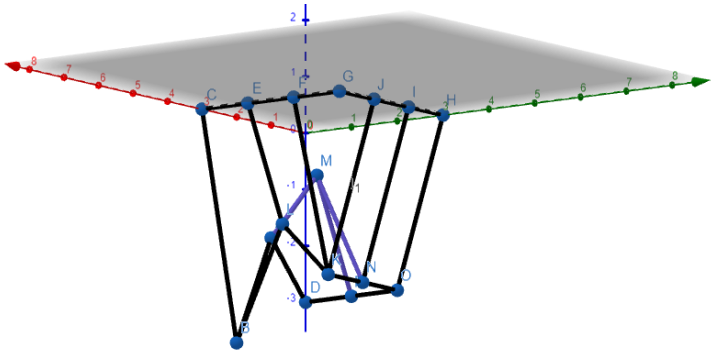

Figura 5.9: Ejemplo de filtración Lower-Star con la matriz cambiada de signo (que se corresponde con al filtración Upper-Star).

Esta vez, al ir haciendo cortes y observando las diferentes componentes conexas, lo primero con lo que nos toparemos será con los puntos que antes eran más altos, en este caso, empezaríamos por el punto B (figura 5.10).

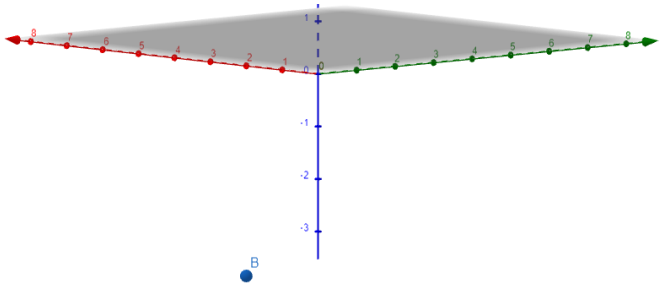

Figura 5.10: Primera componente conexa con Upper-Star

Continuando con los cortes que se realizan, obtendríamos una segunda componente conexa formada por los puntos K, N, O, P, D(representada en naranja en la figura 5.11).

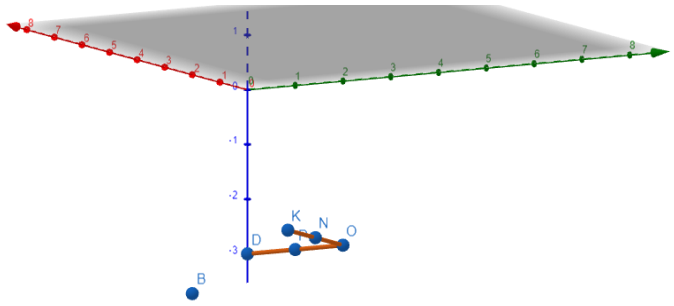

Figura 5.11: Segunda componente conexa con Upper-Star

Tras realizar este tercer corte ya obtendríamos una sola componente conexa, que permanecerá en el tiempo, puesto que por mucho que se vaya subiendo, sólo se añadirán nuevos puntos a la misma componente (figura 5.12).

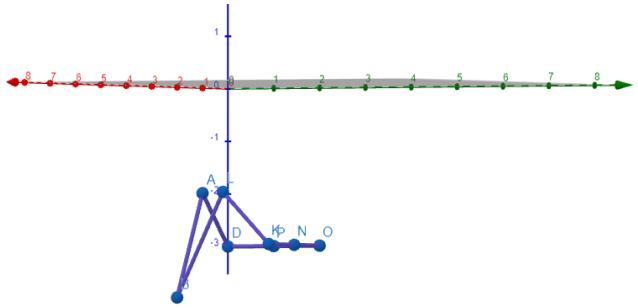

Figura 5.12: Una sola componente conexa con Upper-Star

Gracias a esta filtración podremos obtener información sobre lo heterogéneos que son los puntos más altos, quedándonos las barras representadas en la figura 5.13.

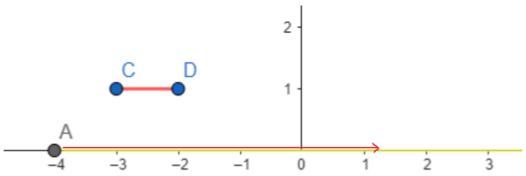

Figura 5.13: Barcode Upper-Star

Esto nos permitirá estudiar los máximos de los que no recogíamos información mediante Lower-Star y ver cómo de diferentes son esos puntos en relación a sus vecinos de la malla.

6. Herramientas de programación y librerías utilizadas

6.1. Introducción

En este capítulo se explicarán las diferentes herramientas que se han utilizado en el proyecto y se detallará cómo deben instalarse para mayor facilidad de uso de cara a futuras personas que vayan a hacer uso de este proyecto.

6.2. Python

En este proyecto utilizaremos Python, un lenguaje de programación de alto nivel, muy utilizado en diversos aspectos como desarrollo web, ciencia de datos, inteligencia artificial y mucho más.

Para esta ocasión se ha utilizado la última versión disponible de Python, la 3.11 que puede descargarse desde la página oficial de la que deberemos descargarnos la versión ejecutable según las características de nuestro equipo. Y simplemente seguiremos la instalación con las opciones por defecto, teniendo en cuenta que debemos marcar la opción "Add python.exe to PATH" en la primera página del instalador.

6.3. Anaconda

Anaconda es una distribución de Python que incluye una serie de paquetes realmente útiles para poder trabajar con datos, además de incluir Jupyter Notebook, herramienta que nos permitirá trabajar de forma más cómoda. Lo instalaremos desde su web oficial, de acuerdo a la versión de Python que hayamos descargado. En este caso la instalación será sencilla, simplemente seguiremos la instalación con las instrucciones por defecto.

6.4. Ripser, Scikit-tda y Persim

Para el cálculo de la homología persistente y entropía persistente se necesita instalar las biblioteca de Python Ripser [20] y scikit-tda [19]. También necesitaremos la biblioteca Persim, puesto que es una dependencia de Ripser. Para usarlas sólo deberemos importar estas bibliotecas en nuestro proyecto una vez las hayamos instalado. La instalación es muy simple, sólo deberemos introducir los siguientes comandos en la consola de nuestro ordenador: pip install Cython pip install Ripser pip install scikit-tda

7. Extracción de los datos

7.1. Introducción

Tras intentar ejecutar el código del trabajo predecesor de Fernando Claros [6] sin éxito (por problemas en la instalación de dependencias), contacté con el autor de los artículos sobre los que se basa este trabajo, Cuneyt Gurcan Akcora, puesto que tiene varios proyectos en GitHub en el que se descarga los datos de bitcoin y forma un grafo con ellos. Al presentarle nuestro proyecto y preguntarle que cuál de ellos sería mejor utilizar, Akcora nos aconsejó leernos los apartados del 3 al 3.5 del artículo Blockchain networks: Data structures of Bitcoin, Monero, Zcash, Ethereum, Ripple, and Iota [10] por lo que vamos a proceder a continuación a explicar dichos puntos del mismo.

Finalmente los datos del repositorio antes comentado se han usado en este proyecto pero de forma indirecta, ya que descubrimos que en el repositorio principal del proyecto de Akcora [11] ya estaban implementadas las matrices de ocurrencia de los chainlets, y por tanto decidimos directamente trabajar con ellas, evitando así tener que tratar el gran volumen de datos que suponía, además de requerir recursos que no disponemos (por potencia computacional). Igualmente, en este capítulo se procederá a explicar la estructura de estos datos dentro del repositorio, tras hablar sobre el artículo aconsejado por el matemático.

7.2. Blockchain networks: Data structures of Bitcoin, Monero, Zcash, Ethereum, Ripple, and Iota

En esta sección hablaremos sobre el artículo recomendado por Akcora [10] centrándonos en Bitcoin puesto que es la parte que más nos interesa del que finalmente sólo utilizaremos los puntos 3 y 3.1 puesto que los puntos 3.2 y 3.3 eran modelos de grafos diferentes al que se pensaba usar en este trabajo (grafo de transacciones en el que no aparecen las direcciones y grafo de direcciones en el que no aparecen las transacciones). En el apartado 3.4 de [10], Akcora habla sobre monedas que no son Bitcoin y en el 3.5 explica los Chainlets, algo que ya hemos tratado en puntos anteriores.

Para empezar, debemos entender que Bitcoin almacena los datos de bloques secuenciales en archivos blk*.dat. Cada uno de estos archivos contiene un número diferente de bloques (puesto que el número y tamaño de las transacciones determinan en tamaño del bloque). Para poder separar los diferentes bloques que se encuentren dentro de este, se añade un byte mágico en estos archivos, en el caso de Bitcoin se utiliza 0xD9B4BEF9.

Como hemos comentado anteriormente a lo largo de este trabajo, cada bloque contiene un conjunto de transacciones, por las que los verificadores pueden o no llevarse una comisión, según decidan los participantes de las transacciones. Además de esta

comisión, el minero se lleva una recompensa por haber minado el bloque, pero, ¿de dónde viene esta recompensa?

Pues bien, al principio de cada bloque existe una transacción de base, aparte de las que se verifiquen de ese bloque, gracias a la cual, el minero recibirá la recompensa por haber minado el bloque. Asimismo, si el minero intentara registrar una recompensa mayor a la establecida, la red rechazaría el bloque automáticamente.

Otro aspecto bastante peculiar sobre esta criptomoneda del que se habla en el artículo [10], es que con estas transacciones, no podremos saber de forma directa la dirección del input de estas ni la cantidad que realmente se está gastando, sino que la única información que tendremos es que tenemos como input el índice de salida i de la transacción x tal y como puede verse en la siguiente figura 7.1 :

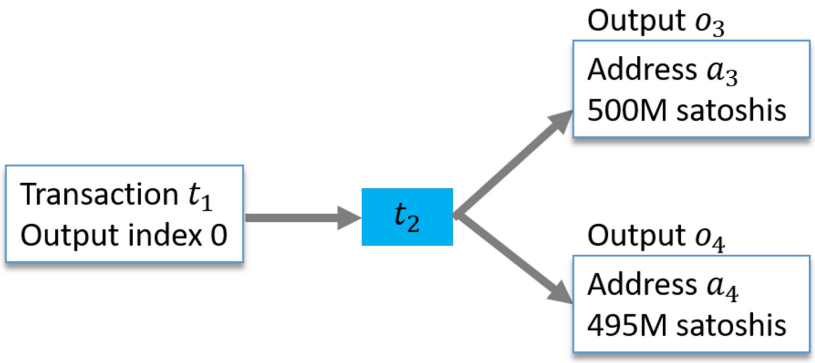

Figura 7.1: Ilustración sobre transacciones con Bitcoin.

También podemos observar que sabiendo que el output con índice 0 de la transacción t_1 es 1B de satoshis, hay una diferencia de 5M de satoshis si sumamos la cantidad transferida a los outputs. Esto se debe a que, tal y como hablábamos en el capítulo 4, los integrantes de una transacción deciden dar o no cierta cantidad de la moneda como concepto de tarifa de transacción. Normalmente, a mayor tarifa de transacción, mayor probabilidad tendrá la misma de ser elegida para ser incluida en el bloque y verificada. En caso de que el minero modificara la transacción para tener una tarifa mayor, o se equivocara a la hora de seleccionar la dirección del input, la red rechazaría el bloque.

7.2.1. Graph rules for UTXO Blockchains

Una vez las transacciones realizadas quedan registradas en la Blockchain, se llevan a cabo de forma efectiva las transferencias de dinero. Una vez recibido el dinero, ya podremos gastarlo en otra transacción si disponemos de la clave privada correspondiente a la dirección desde la que vamos a enviarlo.

En este proceso no encontramos con el término **UTXO** (Unspent Transaction Output), en español significa Salida de Transacción no gastada. Este término, hace referencia a la salida de una transacción que aún no ha sido gastada. Son estas direcciones las que nos permitirán rastrear la cantidad de cada una de las direcciones que aparecen en la Blockchain. Esta mantiene un registro de direcciones, de tal forma que podremos asegurar la integridad de la red, verificando la validez de las transacciones.

Por último, en [10], los autores nos presentan varias reglas sobre la red de Bitcoin que ya se han visto anteriormente 4, pero cabe destacar una de ellas. Una práctica recomendada por la red es esperar seis bloques minados tras recibir una cantidad de la moneda antes de gastarla (por las bifurcaciones que puedan producirse, tal y como se comentó en el capítulo 3). Sin embargo, se observan casos en los que el output recibido se gasta incluso dentro del mismo bloque.

7.3. Bitcoin Transaction Network

Una vez comentadas algunas peculiaridades más sobre Bitcoin, se explicará cómo están estructurados los datos en el repositorio recomendado por este[1]. Gracias a dicho repositorio, es posible descargar los datos de las transacciones de los bloques 0 al bloque 737900, contenidos en un fichero .zip en el que se encuentran los datos por tramos de 73600 en 73600 bloques (hasta llegar al bloque 737900).

Estos datos, están almacenados en dos tipos de archivos, los de input y los de output, es decir, unos tendrán la información de las entradas de cada transacción, y otros tendrán información sobre las salidas de cada transacción. Además, en los ejemplos podremos darnos cuenta de una pequeña particularidad, y es que la unidad monetaria en la que vienen representados los datos en Bitcoin es en Satoshis.

7.3.1. Archivos de input o entrada

Los archivos de input nos permitirán ver la información sobre las entradas de las transacciones con bitcoin que se realicen en un bloque determinado. En el conjunto de datos que ponen a nuestra disposición, los datos estarán ya parseados gracias a la librería Bitcoin-ETL de Google [15] y quedan a nuestra disposición con la estructura que se presenta a continuación:

Nº de bloque	Hash de la transacción	Nº de inputs	Part. i (dirección)
Cantidad de i	Part. $i+1$ (dirección)	Cantidad de $i+1$	...

Cuadro 7.1: Tabla de formato de datos de bitcoin en cada bloque en los archivos input.

En la tabla 7.2 podemos ver un ejemplo de cómo aparecen los datos en estos archivos suponiendo que quisiéramos modelar las transacciones que aparecen en la figura 4.1:

BlockHeightOft1	HashOft1	2	a_1	$1 * 10^8$	a_2	$1 * 10^8$		
BlockHeightOft2	HashOft2	3	a_3	$2 * 10^8$	a_4	$1 * 10^8$	a_5	$1 * 10^8$
BlockHeightOft3	HashOft3	1	a_7	$0{,}8 * 10^8$				
BlockHeightOft4	HashOft4	2	a_{11}	$0{,}3 * 10^8$	a_8	$3{,}8 * 10^8$		

Cuadro 7.2: Tabla de ejemplo de datos en archivos de input.

[1]Bitcoin Transaction Network Files https://chartalist.org/BitcoinData.html

7.3.2. Archivos de output o salida

Estos archivos nos darán información sobre las salidas de las transacciones del bloque que hayamos elegido. Aquí cabe destacar que puede darse el caso de que las direcciones de salida de una transacción, no sigan el estándar por la librería que los parsea, y que aparecerán con una estructura parecida a la siguiente: nonstandard98a1069f93253f2dcf24e989c69c53447fa0870c.

Al igual que con los datos de entrada, los datos de salida tienen un formato específico para que podamos tratarlos (podemos observarlo en la tabla 7.3).

Nº de bloque	Hash de la transacción	Nº de outputs	Part. i (dirección)
Cant. que recibe i	Part. $i+1$ (dirección)	Cant. que recibe $i+1$	...

Cuadro 7.3: Tabla de formato de datos de bitcoin en cada bloque en los archivos output.

Seguidamente, podemos ver a continuación del ejemplo anterior, pero esta vez con los datos de salida.

BlockHeightOft1	HashOft1	2	a_6	10^8	a_7	$0{,}8 * 10^8$		
BlockHeightOft2	HashOft2	1	a_8	$3{,}8 * 10^8$				
BlockHeightOft3	HashOft3	3	a_9	$0{,}2 * 10^8$	a_{10}	$0{,}2 * 10^8$	a_{11}	$0{,}3 * 10^8$
BlockHeightOft4	HashOft4	2	a_{12}	$3{,}7 * 10^8$	a_{13}	$0{,}3 * 10^8$		

Cuadro 7.4: Tabla de ejemplo de datos en archivos de output.

8. Implementación

Aunque al comienzo de esta etapa del proyecto pretendíamos formar el grafo de transacciones, tras no conseguir utilizar el código del repositorio [11] de Akcora, nos decantamos por utilizar las matrices de ocurrencia por días que se encuentran ya en dicho repositorio formadas a partir de los datos explicados en el capítulo anterior.

En este capítulo explicaremos cómo están construidos los datos a partir de los cuales Akcora ha creado las matrices de ocurrencia con las que se van a trabajar, así como el tratamiento que se le ha dado a las mismas en este proyecto y los experimentos que se han llevado a cabo.

8.1. Estructura del repositorio de Akcora

El repositorio de Akcora Coinworks [11] es de donde se han obtenido los datos que se van a utilizar en este proyecto. El repositorio se divide en dos partes principales. En la primera parte se procesan ficheros con los datos de Bitcoin desde 2009 hasta 2018. Una vez procesados, Akcora pudo extraer ficheros con datos procesables más útiles, como por ejemplo, dos ficheros .rar. En el primero, podremos observar matrices de 20x20 que reflejan las ocurrencias de los chainlets(son con las que vamos a trabajar), y en el segundo, ficheros con matrices del mismo tamaño en el que podremos observar la cantidad de la moneda que se mueve en los diferentes tipos de chainlets. La segunda parte de este proyecto, además de procesar los datos, Akcora clasifica los datos de las transacciones por años y meses, y cada año estará asociado con 12 archivos en relación a las entradas de las transacciones y 12 archivos relacionados con las salidas de las transacciones (uno por cada mes del año).

8.2. Matriz de ocurrencia

Como se comenta en el apartado anterior, en el repositorio de Akcora [11], si nos dirigimos al apartado de "data" y dentro de este al archivo de "dailyOccmatrices2009-2018.rar", encontraremos las matrices de ocurrencia, cuya estructura está explicada en el capítulo 4. En este apartado aclararemos con un ejemplo real las partes de esta matriz. Gracias al estudio de las mismas, podremos realizar observaciones como el tipo de chainlets que más se repite (por ejemplo, en el caso de la figura 8.1, el que más se repite es $C_{1\to 2}$).

Otra cosa que también puede llamarnos la atención, es que algunos de los elementos de la última fila y la última columna sean tan altos. Pero como explicábamos en el capítulo 4, la última fila y la última columna representan el resto de chainlets que no caben en la muestra, es decir, representan la suma de todos los chainlets que se desbordan de la matriz.

Por ejemplo, en la siguiente matriz, el elemento (1, 20) representa todos los chainlets con una entrada y 20 o más salidas ($C_{1\to j>20}$). Ocurre igual en el caso del elemento (20, 7), gracias a esta matriz, sabemos que hay 11 chainlets que tienen 20 entradas o más y 7 salidas ($C_{i>20\to 7}$).

30310	140808	2535	1018	575	383	349	283	227	171	150	107	65	66	73	77	70	69	60	1386
2882	20538	1950	407	197	117	76	60	48	30	26	42	17	17	18	22	18	11	11	151
3172	7964	653	95	65	72	26	25	17	20	10	9	7	11	5	4	8	4	6	82
686	3718	175	95	37	34	25	26	12	19	4	7	4	4	1	5	2	2	4	50
556	2188	97	33	17	19	15	8	11	11	9	3	3	3	1	1	1	3	2	32
337	1334	69	24	22	6	5	13	5	9	4	8	2	1	0	0	1	1	1	21
373	900	40	17	15	11	4	8	3	11	4	3	1	1	3	0	0	1	2	10
310	563	23	11	9	8	2	6	3	6	6	1	0	2	1	0	0	0	0	10
155	489	15	3	8	3	6	4	2	1	1	2	0	0	0	0	2	0	0	6
147	397	20	9	2	3	3	6	4	2	6	3	1	0	1	1	0	0	1	4
81	255	6	3	1	6	1	3	2	2	3	1	1	2	0	2	0	0	0	5
66	209	1	2	0	0	2	2	3	1	1	1	1	2	0	0	0	0	2	4
60	170	4	0	1	1	0	1	1	3	2	2	2	0	0	1	1	0	1	6
47	165	1	1	2	1	0	1	1	1	1	0	1	3	0	0	0	0	0	5
34	146	1	5	1	2	0	2	0	0	0	0	1	2	1	0	0	0	0	4
42	104	3	0	2	3	1	1	1	4	0	1	0	0	0	0	1	0	0	2
28	134	8	1	0	0	1	0	2	1	1	1	1	0	1	0	1	0	0	2
26	81	5	1	0	1	1	1	0	2	1	1	2	0	0	1	0	0	0	1
25	85	2	0	1	1	3	3	2	2	1	2	1	0	0	1	0	0	0	5
1161	1300	19	24	7	11	11	4	5	9	4	5	6	5	5	3	0	3	4	48

Figura 8.1: Matriz de ocurrencia del día 154 de 2016, correspondiente al 2 de junio de 2016.

8.3. Importación de los datos

El primer paso que deberemos hacer en el desarrollo este proyecto será importar los datos de las matrices desde cada archivo del repositorio. Para ello se ha desarrollado la función "lee_archivo_dev_matriz". Gracias a esta, podremos leer los datos de una matriz de ocurrencia pasándole como parámetro el nombre del archivo en el que se encuentra (dentro del directorio "Datos/dailyOccmatrices"). En esta función, leeremos el archivo en formato csv en cuestión y meteremos en una lista de listas los datos de la matriz de ocurrencia. Finalmente, transformamos la lista de listas en una matriz gracias a la biblioteca **numpy**.

```
def lee_archivo_dev_matriz(nombre_archivo):
    carpeta = 'Datos/dailyOccmatrices/'
    ruta_archivo = os.path.join(carpeta, nombre_archivo)
    with open(ruta_archivo, 'r') as archivo_csv:
        lector_csv = csv.reader(archivo_csv)
        matriz = []

        for fila in lector_csv:

            fila_enteros = [int(elemento) for elemento in
               fila]
            matriz.append(fila_enteros)

    matrix = np.array(matriz)
```

```
    return(matrix)
```

Extracto de código 8.1: Función de lectura de datos de una matriz de ocurrencia

8.3.1. Normalización y representación en imagen a escala de grises

Ya que una imagen en escala de grises, no es más que una matriz con valores entre 0 y 255, decidimos normalizar los datos de la matriz de ocurrencia transformándola en una que podamos representar como una imagen a escala de grises con idea de obtener una representación gráfica de los datos en forma de imagen, reflejando los tipos de chainlets que más se repiten. Para esto, se ha creado la función "normalizar_a_escala_de_grises", en la cual, cada nuevo elemento (e) vendrá definido por la fórmula que se presenta a continuación, siendo em el elemento de la matriz que vamos a tratar, $minim$ el elemento más pequeño de la matriz y $maxim$ el mayor elemento de la matriz.

$$e = \frac{em - minim}{maxim - minim} * 255$$

La función comentada en el apartado anterior aplica la fórmula descrita para cada elemento de la matriz y devuelve la matriz ya normalizada. Cabe destacar que la función está preparada para matrices de 20×20. Si se quisiera modificar el tamaño de la matriz a normalizar habría que cambiar el rango de los **for** que se encuentran dentro de la función.

```
def normalizar_a_escala_de_grises(matr):
    maxim, minim = matr.max(), matr.min()
    res = []
    for i in range(20):
        fila = []
        for j in range(20):
            elem = matr[i, j]
            nuevo_elem = ((elem-minim)/(maxim-minim))*255
            fila.append(int(nuevo_elem))
        res.append(fila)
    return np.array(res)
```

Extracto de código 8.2: Función para la normalización de los datos

El siguiente paso es representar la imagen a partir de la matriz que acabamos de normalizar, esto será posible gracias a pyplot de la librería **matplotlib**.

```
plt.imshow(resultado, cmap='gray')
plt.show()
```

Extracto de código 8.3: Representación de la imagen a escala de grises

Una vez llegados a este punto, nos encontramos con que la mayoría de los elementos de la matriz son muy pequeños en comparación con el elemento máximo de la misma. Entonces, al normalizar y redondear a entero, la mayoría de elementos toman el valor cero, haciendo que perdamos mucha información. Es por esto que decidimos utilizar los datos sin normalizar para no perder nada de información a la hora de calcular la entropía.

8.4. Experimentaciones

Una vez hemos importado los datos, comenzaremos con los diferentes experimentos que se han llevado a cabo calculando la homología persistente mediante las diferentes filtraciones presentadas en el capítulo 5, de forma que posteriormente calcularemos su entropía observando si hay resultados diferentes para cada una de las filtraciones usadas. Cada una de las filtraciones se ha utilizado de diferentes formas en busca de conclusiones con el objetivo de encontrar propiedades que puedan resultar interesantes en este proyecto.

8.4.1. Vietoris-Rips

En este apartado, vamos a proceder a explicar los diferentes experimentos que he realizado utilizando la filtración de Vietoris-Rips.

Primer experimento

Para aplicar esta filtración a las matrices de ocurrencia anteriormente explicadas, se ha creado la función calcula_rips que nos calculará la entropía de la matriz de ocurrencia de tamaño 20×20 que le pasamos como parámetro (extracto de código 8.4).

```
def calcula_rips(matriz):
    ind=20
    data = np.zeros((ind*ind, 3))
    ##Transformamos la matriz al formato que necesitamos para
        calcular la homologia con Rips##
    for i in range(ind):
        for j in range(ind):
            data[i*ind+j][0] = i
            data[i*ind+j][1] = j
            data[i*ind+j][2] = matriz[i][j]

    rips = Rips(maxdim=0)
    diagrams = rips.fit_transform(data)
    entropia = persim.persistent_entropy.persistent_entropy(
       diagrams, keep_inf=False, val_inf=None, normalize=
       False)
```

```
    return entropia
```
Extracto de código 8.4: Función para calcular la entropía mediante Vietoris-Rips

A partir de esta función, podremos calcular la entropía persistente asociada a cada matriz de ocurrencia de 2017 (se ha decidido utilizar este rango de tiempo para todos los experimentos puesto que es el último año del que hay datos de forma que estos sean lo más reciente posible y no sean demasiados), guardando en un diccionario las entropías asociadas a cada día:

```
carpeta = 'Datos/dailyOccmatrices'
mat2017 = [mat for mat in os.listdir(carpeta) if "occ2017" in
    mat]
dicc_entropia_rips = {}

for nombre_archivo in mat2017:
    if nombre_archivo.endswith('.csv'):
        matrix = lee_archivo_dev_matriz(nombre_archivo)
        dicc_entropia_rips[nombre_archivo] = calcula_rips(
           matrix)
```
Extracto de código 8.5: Cálculo de entropías mediante Vietoris-Rips para todos los días de 2017

Gracias al siguiente extracto de código, podremos observar una gráfica lineal de los datos anteriormente calculados (figura 8.2). En dicha gráfica, en el eje x estarán representados cada uno de los días de 2017 y en el eje y su entropía asociada.

```
claves_rips = list(dicc_entropia_rips.keys())
valores_rips = list(dicc_entropia_rips.values())

fig, ax = plt.subplots()
ax.plot(claves_rips, valores_rips)

ax.set_title('Entropia por dias')
ax.set_xlabel('Dia')
ax.set_ylabel('Entropia')
plt.show()
```
Extracto de código 8.6: Representación de entropías mediante Vietoris-Rips de 2017

Para poder observar las matrices cuya entropías sean menor y mayor al haber usado esta filtración, utilizaremos el siguiente extracto de código:

```
clave_max = max(dicc_entropia_rips, key=dicc_entropia_rips.
   get)
clave_min = min(dicc_entropia_rips, key=dicc_entropia_rips.
   get)

print("La clave asociada al valor maximo mediante Rips es:",
   clave_max, " con entropia igual a: ", dicc_entropia_rips[
   clave_max])
```

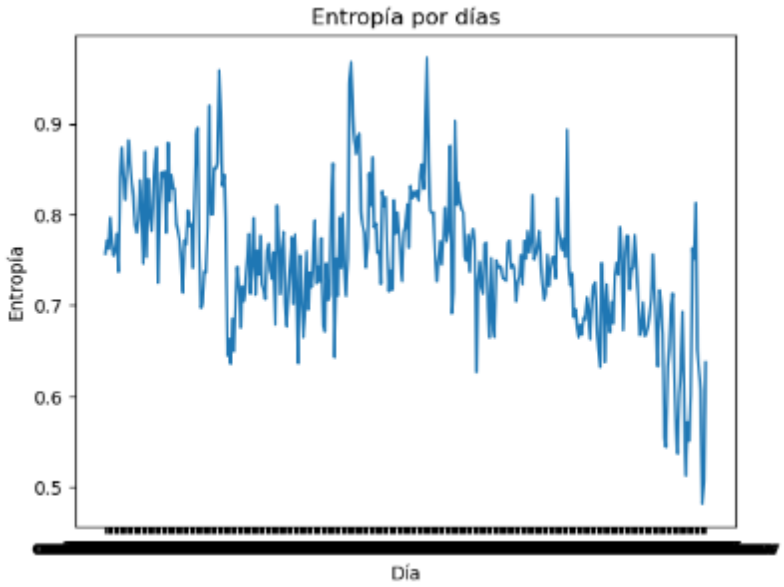

Figura 8.2: Representación de entropía persistente con Vietoris-Rips.

```
print("La clave asociada al valor minimo mediante Rips es:",
   clave_min, " con entropia igual a: ", dicc_entropia_rips[
   clave_min])
matriz_maxima = lee_archivo_dev_matriz(clave_max)
matriz_minima = lee_archivo_dev_matriz(clave_min)
print("La matriz con mayor entropia mediante Rips es:")
print(matriz_maxima)
print("=====================================================")
print("La matriz con menor entropia mediante Rips es:")
print(matriz_minima)
```

Extracto de código 8.7: Matrices de entropía máxima y mínima con Vietoris-Rips

Una vez hecho esto, podremos observar los resultados de las matrices correspondientes a los días con mayor y menor entropía de 2017 en las figuras 8.3, 8.5 y 8.5. Puesto que no encontramos mucha diferencia entre ambas, procederemos a realizar otro experimento en busca de obtener resultados más claros.

```
La clave asociada al valor máximo mediante Rips es: occ2017196.csv  con entropía igual a:  [0.97258664]
La clave asociada al valor mínimo mediante Rips es: occ2017363.csv  con entropía igual a:  [0.48143364]
```

Figura 8.3: Días de entropía máxima y mínima mediante Vietoris-Rips

```
La matriz con mayor entropía mediante Rips es:
[[ 16433 121686   3872   1384    603    495    324    336    306    201
     526    138    121     83     84     64     56     48     46    626]
 [  3007  27524   1168    317    105     83     74     45     43     28
      29     26     29     16     13     21     15     13     26    146]
 [  1299   7818    445    244     64     61     51     52     26     19
      18     15     18     15      9     15     12     12      8     62]
 [   741   3685    160     99    101     64     29     18     14     14
      20      8     13      9      7     11      7      4      4     65]
 [   449   2045    104     32     39     72     22      8     13      7
       5      8      6      6      1      2      4      6      7     48]
 [   401   1345     65     26     27     43     20     14     13     12
       7      8      4      4      3      2      5      7      5     29]
 [   223    789     31     26     20     14     12     17     14     11
       7      5      2      5      2      2      2      4      1     20]
 [   157    581     52     12     21     12      9     11     18     12
       8      3      4      1      2      2      4      2      2     16]
 [   144    475     25      9     11     18      2     11      6      5
       4      3      1      3      2      6      5      2      0      6]
 [   105    431     32      5     11     14     11     10      8      9
       6      3      8      2      4      1      1      3      1     12]
 [   102    312     23      5      8     11      7      8      5      2
       3      2      1      2      1      0      1      1      0      6]
 [    68    350      6      7     17      3      1      2      1      2
       0      1      1      1      2      0      0      3      0      4]
 [    60    229     10      5      6      3      4      2      4      0
       3      4      1      0      1      1      1      0      0      4]
 [    59    165      3      5      4      6      1      1      2      2
       1      1      1      0      0      1      0      1      0      2]
 [    54    140      5      5      4      9      3      3      2      0
       1      1      0      0      0      0      1      1      0      0]
 [    38    134      4      1      6      5      3      0      3      1
       1      2      0      2      0      1      1      1      0      5]
 [    37    134      5      6      1      4      2      1      0      1
       1      2      0      1      0      0      1      1      0      2]
 [    28    106      2      1      2      0      4      1      1      0
       0      0      0      2      0      0      0      1      0      6]
 [    26    109      3      1      4      1      0      3      0      1
       1      0      0      0      1      0      0      0      0      1]
 [  2048   1950     42     15     15     31     15     25     17     13
      13     10     13      7     10      4     10      8      6     98]]
```

Figura 8.4: Matriz de entropía máxima mediante Vietoris-Rips

```
La matriz con menor entropía mediante Rips es:
[[ 21426 256264   7685   1467   1194    790    468    353    326    272
     321    208    225    159    148    126    126    103    101   1940]
 [  5113  22649   1792    218    158    218     82     66     57     27
      42     31     31     20     15     29      9     18      6    320]
 [  2095   6555    271    105    477    172     65     59     24     21
      32     28     28     21     25     26     17      7     11    211]
 [   946   2968    104     67     69     83     29     25     28     15
      25     32     18     16     12      7     10      5      7    150]
 [   552   1970     58     43     34     39     28     15     11     19
      23     12     11      9     11      5      8      4      1    129]
 [   322    984     37     26     19     34     19      4      8      4
      14     19     12      7      8      6      1      2      4     93]
 [   204    774     22     11     20     24     12      7      5      4
       7     12     12     17     15      3      6      1      2     68]
 [   151    549     16     11     11     12      8     13      5      7
      10      7      8     10      4      9      3      2      4     72]
 [   112    471      6     12      8      8      5      7      2      3
      11      5      6      5      3      2      5      5      0     45]
 [    99    426     77      3      3      8      4      2      2      4
      36     28     25     17     23     14      8     11     10    121]
 [    58    312     10      5      4      9      0      1      6      2
      10      5      5      1      1      2      0      1      1     29]
 [    57    282      3      6      3      2      3      1      3      3
       8      9      1      3      2      1      1      1      1     16]
 [    39    831      3      3      3      5      3      2      0      0
       3      7      8      2      1      1      0      1      0     17]
 [    30    192      2      3      5      4      2      0      0      1
       1      3      2      2      0      1      4      2      0     20]
 [    28    196      1      1      3      4      5      1      1      1
       0      3      0      4      3      0      1      0      0     16]
 [    30    119      5      2      2      3      2      0      3      0
       1      1      3      2      2      0      0      1      1     15]
 [    23    102      3      2      1      2      5      1      2      1
       1      0      2      2      0      1      0      1      0      7]
 [    19     95      4      1      1      2      3      3      1      2
       1      1      0      0      0      0      1      1      0     16]
 [    12     78      1      4      3      0      1      1      2      0
       1      1      0      1      2      0      1      0      1      7]
 [   818   1482     33     29     22     34     23      9     12     15
      16     16     11     17      9      7     10      3      6    479]]
```

Figura 8.5: Matriz de entropía mínima mediante Vietoris-Rips

Segundo experimento

Al no observar cambios significativos entre las matrices con entropía máxima y mínima en Vietoris-Rips, nos dimos cuenta de que la matriz correspondiente a la entropía más baja tenía barras muy largas y pequeñas, lo que hacía que la entropía resultara pequeña también. Es por esto que decidimos realizar los mismos experimentos, pero normalizando la matriz con valores entre 0 y 1 para que las anomalías fueran más evidentes.

Para ello simplemente hemos tenido que normalizar cada matriz antes de pasarla a cada una de las filtraciones. Esto se ha hecho gracias a la siguiente función:

```
def normalizar(matr):
    maxim, minim = matr.max(), matr.min()
    res = []
    for i in range(20):
        fila = []
        for j in range(20):
```

```
            elem = matr[i, j]
            nuevo_elem = ((elem-minim)/(maxim-minim))
            fila.append(nuevo_elem)
        res.append(fila)
    return np.array(res)
```

Extracto de código 8.8: Normalización de matrices

Esta es muy similar a la que utilizamos para normalizar las matrices y poder representarlas mediante imágenes, sólo que no hará falta que multipliquemos por 255.

$$e = \frac{em - minim}{maxim - minim}$$

Gracias a esta normalización, hemos obtenido la gráfica de la figura 8.6.

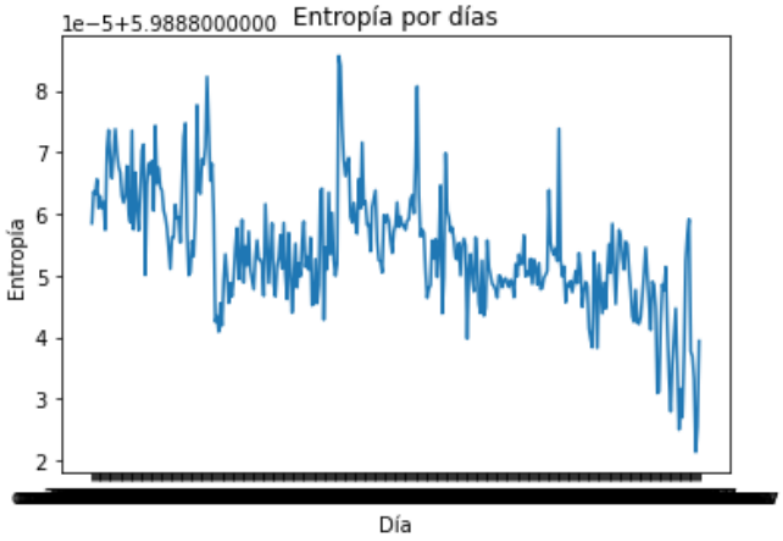

Figura 8.6: Gráfica de entropía con Vietoris-Rips con las matrices normalizadas

Como puede observarse en las figuras 8.6, 8.7, 8.8 y 8.9 mediante este experimento todas las entropías tienen unos valores muy cercanos al 5,9888, la razón se explicará en el siguiente capítulo.

```
La clave asociada al valor máximo mediante Rips es: occ2017149.csv  con entropía igual a:  [5.98888563]
La clave asociada al valor mínimo mediante Rips es: occ2017363.csv  con entropía igual a:  [5.98882135]
```

Figura 8.7: Valores de entropía máxima y mínima con Vietoris-Rips normalizando la matriz

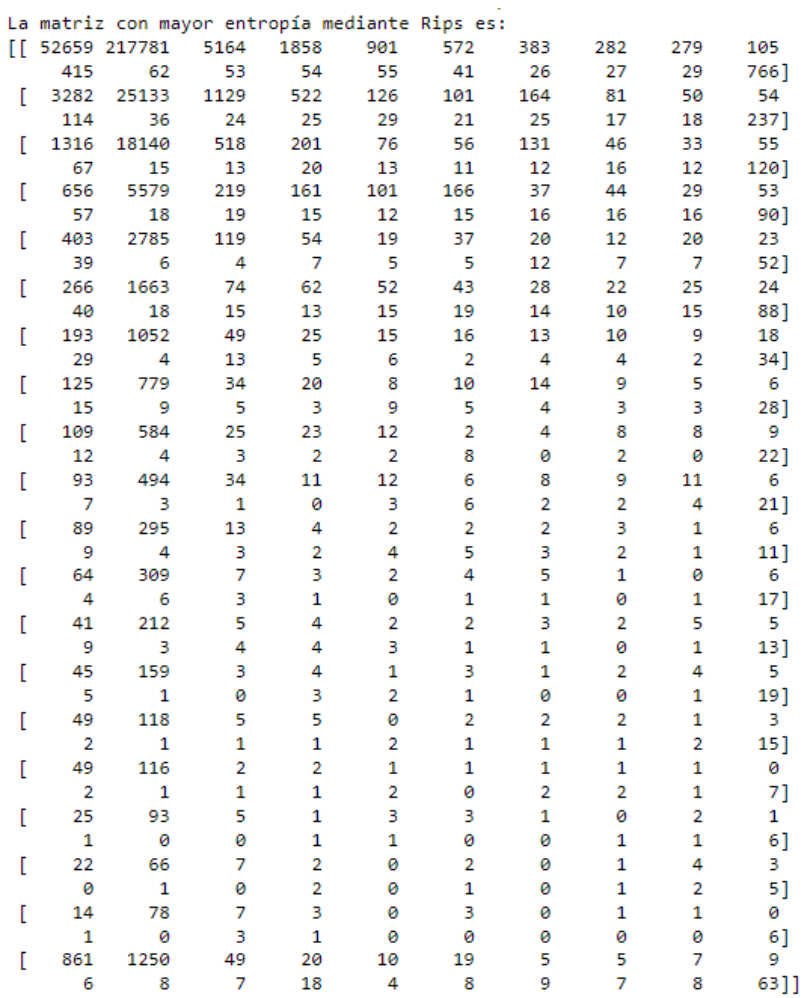

Figura 8.8: Matriz de entropía máxima con Vietoris-Rips normalizando la matriz

```
La matriz con menor entropía mediante Rips es:
[[ 21426 256264   7685   1467   1194    790    468    353    326    272
     321    208    225    159    148    126    126    103    101   1940]
 [  5113  22649   1792    218    158    218     82     66     57     27
      42     31     31     20     15     29      9     18      6    320]
 [  2095   6555    271    105    477    172     65     59     24     21
      32     28     28     21     25     26     17      7     11    211]
 [   946   2968    104     67     69     83     29     25     28     15
      25     32     18     16     12      7     10      5      7    150]
 [   552   1970     58     43     34     39     28     15     11     19
      23     12     11      9     11      5      8      4      1    129]
 [   322    984     37     26     19     34     19      4      8      4
      14     19     12      7      8      6      1      2      4     93]
 [   204    774     22     11     20     24     12      7      5      4
       7     12     12     17     15      3      6      1      2     68]
 [   151    549     16     11     11     12      8     13      5      7
      10      7      8     10      4      9      3      2      4     72]
 [   112    471      6     12      8      8      5      7      2      3
      11      5      6      5      3      2      5      5      0     45]
 [    99    426     77      3      3      8      4      2      2      4
      36     28     25     17     23     14      8     11     10    121]
 [    58    312     10      5      4      9      0      1      6      2
      10      5      5      1      1      2      0      1      1     29]
 [    57    282      3      6      3      2      3      1      3      3
       8      9      1      3      2      1      1      1      1     16]
 [    39    831      3      3      3      5      3      2      0      0
       3      7      8      2      1      1      0      1      0     17]
 [    30    192      2      3      5      4      2      0      0      1
       1      3      2      2      0      1      4      2      0     20]
 [    28    196      1      1      3      4      5      1      1      1
       0      3      0      4      3      0      1      0      0     16]
 [    30    119      5      2      2      3      2      0      3      0
       1      1      3      2      2      0      0      1      1     15]
 [    23    102      3      2      1      2      5      1      2      1
       1      0      2      2      0      1      0      1      0      7]
 [    19     95      4      1      1      2      3      3      1      2
       1      1      0      0      0      0      1      1      0     16]
 [    12     78      1      4      3      0      1      1      2      0
       1      1      0      1      2      0      1      0      1      7]
 [   818   1482     33     29     22     34     23      9     12     15
      16     16     11     17      9      7     10      3      6    479]]
```

Figura 8.9: Matriz de entropía mínima con Vietoris-Rips normalizando la matriz

Tercer experimento

Esta vez decidimos replicar el segundo experimento, pero esta vez multiplicando por una constante. Podríamos preguntarnos entonces, qué diferencia habría entre esta normalización y en la que normalizábamos las matrices para representarlas en imágenes a escala de grises. Pues bien, la diferencia es que en este caso no se van a perder los decimales, es decir, que no sólo tendremos números enteros a la hora de calcular la homología y entropía persistentes.

Para poder llevar a cabo este experimento, simplemente hemos utilizado una función similar a la anteriormente presentada "normalizar"pero multiplicando cada elemento por 1000. Gracias a este experimento hemos podido encontrar una diferencia más clara entre las matrices con mayor y menor entropía. Podremos observar estas matrices así como sus entropías en las figuras 8.10, 8.11 y 8.12.

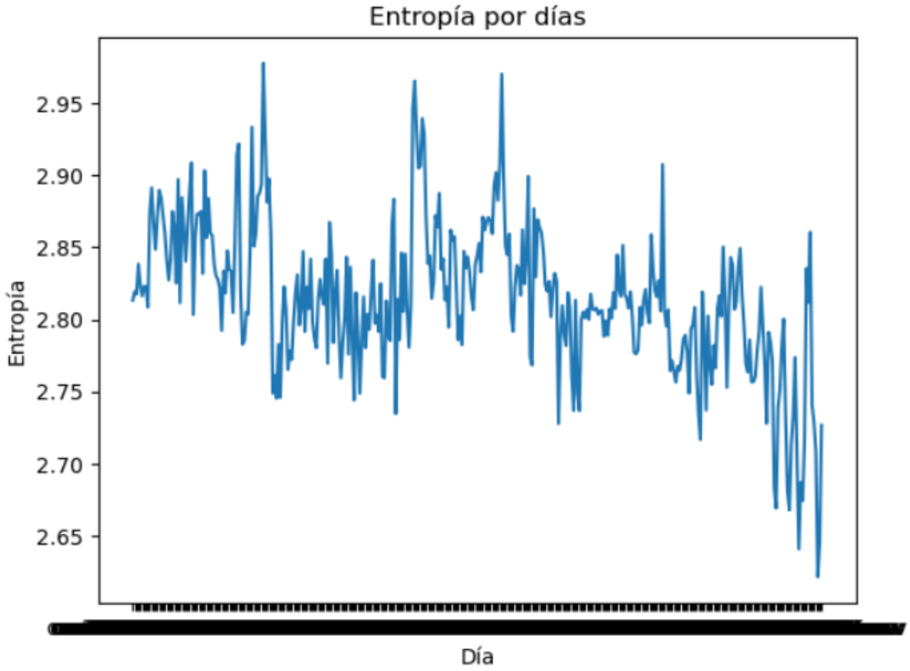

Figura 8.10: Gráfica de entropía mediante el tercer experimento

```
La matriz con mayor entropía mediante Rips es:
232.0 1000.0 28.0 9.0 4.0 2.0 2.0 3.0 2.0 2.0 7.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 5.0
20.0 144.0 6.0 3.0 1.0 1.0 0.0 0.0 0.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
9.0 57.0 3.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
5.0 28.0 1.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
3.0 16.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
3.0 10.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
1.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
10.0 11.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0
```

Figura 8.11: Matriz de entropía máxima mediante el tercer experimento

```
La matriz con menor entropía mediante Rips es:
84.0 1000.0 30.0 6.0 5.0 3.0 2.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 8.0
20.0 88.0 7.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
8.0 26.0 1.0 0.0 2.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
4.0 12.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
2.0 8.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
1.0 4.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
3.0 6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2.0
```

Figura 8.12: Matriz de entropía mínima mediante el tercer experimento

8.4.2. Lower-Star

En este apartado se explicarán los diferentes experimentos que se han realizado utilizando la filtración de Lower-Star.

Primer experimento

En el caso de los experimentos con Lower-Star, se ha tenido que realizar un pequeño paso previo que consiste en elegir cuántos puntos del diagrama de persistencia elegiremos para calcular la entropía. En Vietoris-Rips, no importaba el número de puntos que se fueran a elegir puesto que coincide con el tamaño de la matriz, sin embargo, Lower-Star puede crear un número de barras diferentes para cada matriz (dependiendo de los cráteres o picos que se formen). Ya que para que la comparativa entre entropías tenga significado, éstas deben calcularse sobre el mismo número de barras

Para que todas las entropías sean calculadas con la misma cantidad de barras, deberemos calcular cuantas nos quedan para cada una de las matrices, y saber cuál es el número más pequeño de barras que obtenemos. Para ello, se aplica la filtración Lower-Star para cada una de las matrices, buscando el mínimo número de barras, una vez hecho esto, ya sabremos cuantas barras deberemos seleccionar de cada matriz. Seleccionaremos las barras que más largas (aquellas cuyo $x - y$ sea mayor) puesto que estas barras son las que más información nos proporcionarán, ya que representan componentes conexas que sobreviven más tiempo. Una vez hecho esto, ya podremos proceder a calcular la entropía persistente.

```
carpeta = 'Datos/dailyOccmatrices'
mat2017 = [mat for mat in os.listdir(carpeta) if "occ2017" in
    mat]
dicc_entropia_lwrstar1 = {}
minimo = float('inf')
for nombre_archivo in mat2017:
```

```
    if nombre_archivo.endswith('.csv'):
        matrix = lee_archivo_dev_matriz(nombre_archivo)
        diagrams = lower_star_img(matrix)
        if len(diagrams)<minimo:
            minimo=len(diagrams)
minimo=minimo-1
for nombre_archivo in mat2017:
    if nombre_archivo.endswith('.csv'):
        matrix = lee_archivo_dev_matriz(nombre_archivo)
        dgms_resultado = lower_star_img(matrix)
        dst_puntos = {}
        for punto in dgms_resultado:
            dst_puntos[tuple(punto)] = abs(punto[0]-punto[1])
        puntos_mas_alejados = sorted(dst_puntos, key=
           dst_puntos.get, reverse=True)[1:minimo+1]
        puntos_mas_alejados_para_entropia = []
        for i in range(len(puntos_mas_alejados)):
            for j in range(len(dgms_resultado)):
                if puntos_mas_alejados[i][0]==dgms_resultado[
                   j][0] and puntos_mas_alejados[i][1]==
                   dgms_resultado[j][1]:
                    puntos_mas_alejados_para_entropia.append(
                       dgms_resultado[j])
        puntos_mas_alejados_para_entropia = np.array(
           puntos_mas_alejados_para_entropia)
        entropia = persim.persistent_entropy.
           persistent_entropy(
           puntos_mas_alejados_para_entropia, keep_inf=False,
            val_inf=None, normalize=False)
        dicc_entropia_lwrstar1[nombre_archivo] = entropia
```

Extracto de código 8.9: Cálculo de entropía mediante Lower-Star

Para ver los resultados representados, se utilizara un código similar al utilizado para representar los resultados de Vietoris-Rips.

En este caso, podemos observar que la entropía se mueve en un rango mayor en comparación con Vietoris-Rips. Esto se debe a que en Lower-Star, observamos sobre todo la heterogeneidad de los cráteres que se forman, es decir, de los puntos más bajos (como se observa en el ejemplo del capítulo 5). Esto hace que obtengamos un resultado diferente al obtenido gracias a la filtración anterior. Para obtener las matrices correspondientes a los días de mayor y menor entropía mediante Lower-Star realizaremos los mismos pasos que con Vietoris-Rips, obteniendo los siguientes resultados:

En la matriz correspondiente a la entropía máxima se observa que hay menor cantidad de ceros que en la matriz correspondiente a la entropía mínima. Esta es la forma más clara en la que podemos verlo, pero realmente, la diferencia entre los elementos más bajos de la matriz será mayor en la matriz cuya entropía corresponde a la máxima.

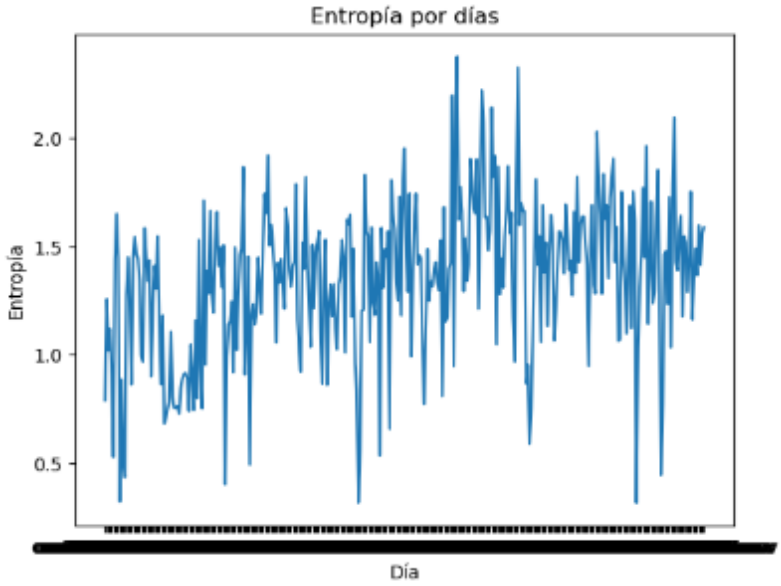

Figura 8.13: Resultados de entropía mediante Lower-Star

```
La clave asociada al valor máximo mediante Lower Star es: occ2017215.csv  con entropía igual a:  [2.37714433]
La clave asociada al valor mínimo mediante Lower Star es: occ2017324.csv  con entropía igual a:  [0.31606166]
```

Figura 8.14: Días de entropía máxima y mínima mediante Lower-Star

Segundo y tercer experimento

Como en Vietoris-Rips, pensamos en normalizar los elementos de las matrices tal y como se realiza en el apartado anterior. Sin embargo, al observar los resultados normalizando las matrices y normalizando y multiplicando por una constante, obtuvimos los mismos resultados que en el primer experimento.

8.4.3. Upper-Star

Como se comenta en el capítulo 5, ahora se procederá a realizar los mismos experimentos que en el apartado anterior, sólo que esta vez, cambiaremos el signo a los elementos de cada matriz para estudiar la entropía persistente de los puntos máximos.

Primer experimento

Para este experimento, se ha utilizado el mismo código que en para Lower-Star, pero esta vez, se les pasa como parámetro la matriz negativa, pero salvo por eso, el proceso es exactamente el mismo.

Tras realizar este proceso, hemos obtenido la siguiente gráfica en la que se representa la entropía de cada día calculada mediante Upper-Star:

```
La matriz con mayor entropía mediante Lower Star es:
[[ 19556 128308   4514   1211    547    509    328    309    277    215
     436    132    134    119    113     91     70     69     57    985]
 [  6491  21913   1286    312    108     97     44     43     31     39
      36     19     17      7     17     12     15     15      9    183]
 [  2879   7759   1881    213     69     79     65    211     91     39
      29     12     14      8     13      3      5      5      4     96]
 [  1507   3933    130     53     79     95     33     32     14     18
       6     14      9      4      5      4      4      4      2     89]
 [   848   2168     48     31     33     96     23     22      8     18
       8      4      2      2      6      4      4      3      3     45]
 [   583   1321     65     26     21     65     29     25     12      7
       9      9      6      8      3      1      2      2      2     54]
 [   377    848     38     14     20     47     10     15     13      7
       6      3      2      3      3      1      3      0      1     38]
 [   300    569     21     10     19     34      7     10     14     10
       3      5      5      0      4      3      2      2      3     35]
 [   210    452     18     10      7     35     14      8     13      7
       9      6      6      3      2      0      1      2      4     20]
 [   140    394      8     37      5     25      4     10      9     14
       6      4      4      2      3      2      2      2      2     17]
 [   143    198     10      3      3     14      4      5      1      5
       1      4      5      0      0      0      0      2      2      9]
 [    98    311      6      5      3      7      1      3      5      3
       3      3      1      4      0      3      1      0      1     10]
 [    90    199      4      4     17      6     10      5      6      5
       1      0      0      3      2      0      0      0      2      6]
 [    91    118      1      3      2      1      7      1      2      0
       2      2      1      1      0      1      0      0      2      3]
 [    68     95      4      0      4      7      2      1      1      1
       2      4      4      0      2      0      1      1      0      3]
 [    55     98      5      1      3     23      3      4      1      3
       2      4      1      3      1      0      1      0      0      4]
 [    50     83      5      3      4      9      1      0      1      3
       2      2      0      3      1      2      0      0      1      3]
 [    38     87      4      0      0      2      0      3      1      1
       1      1      2      0      0      2      1      1      1      3]
 [    44     59      5      2      0      4      1      0      0      1
       0      2      0      0      1      0      0      0      0      4]
 [  1009   2123     28     42      9     38     19     12     16     24
      10     17     12      8      7     12      6      8      8    139]]
```

Figura 8.15: Matriz de entropía máxima mediante Lower-Star

```
La matriz con menor entropía mediante Lower Star es:
[[ 23942 223864   5878   1533    800    554    339    386    370    200
     470    154    167    160    125    126     99    108    107   1786]
 [  4947  32116   1047    419    128    106     76     64     43     37
      36     20     23     18     11     20     24     15     13    290]
 [  2146  11479    363    168    481     56     58     54     33     27
      51     17     45     30     21     20     11     16     15    192]
 [  1163   5569    106     56     31     36     21     30     26     20
      50     21     19     15      4     10      4      7      6    136]
 [   709   3116     70     29     33     26     11     12     17     25
      48     12     12      7      9      4      4      5      5     93]
 [  3246   1728     33     17     17     24     17      7     11     10
      38     10     10     10     13      5      6     14      4     62]
 [   331   1052     34     22      7      9     24      7      9     12
      25      5      4      4      7      5      2      6      2     64]
 [   234    864     26     15      8     12      7     28      6      5
      12      4      5      6      2      6      7      3      3     58]
 [   220    742     23     10     12     10      1      7     14      6
      15      2      0      2      1      4      2      5      2     37]
 [   137    499     15     10      8     10      5      5     12      4
       9      1      1      4      3      3      1      6      2     44]
 [   140    528      5      9      3      2      6      4      2     14
       4      0      4      2      2      0      0      0      1     16]
 [    93    357      8      2      3      3      2      2      2      2
      15      4      1      0      3      0      1      0      1     11]
 [    85    263      6      1      0      2      0      1      0      1
       8      7      2      1      1      1      0      2      1      9]
 [    83    182      4      2      0      0      1      0      4      2
       5      1      0      2      0      0      1      0      0     11]
 [    75    164      3      2      2      3      1      1      2      2
       6      0      2      1      3      1      2      0      0      8]
 [    67    127      5      2      1      2      1      1      1      1
       2      1      1      4      0      2      0      1      0      6]
 [    60    122      2      0      1      1      1      0      0      0
       1      2      1      2      5      0      1      0      1     12]
 [    47     97      3      0      0      0      3      0      1      0
       4      0      0      0      2      1      1      1      2     12]
 [    54     98      2      0      2      0      0      0      0      1
       2      0      0      0      0      0      0      0      0      9]
 [  1240   1642     73     20     24     19     20     21     13     52
      66     13     26     12      6     15     16      7     12    433]]
```

Figura 8.16: Matriz de entropía mínima mediante Lower-Star

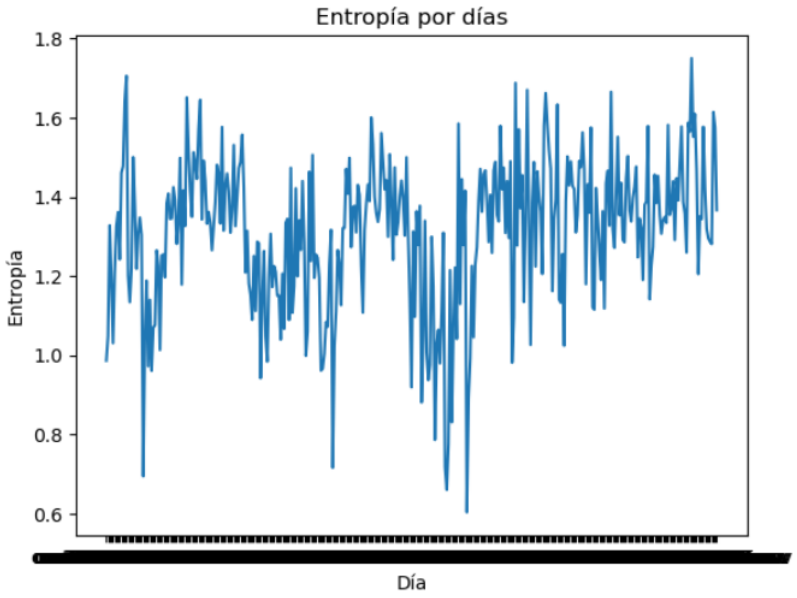

Figura 8.17: Resultados de entropía mediante Upper-Star

Y para poder observar los días con mayor y menor entropía, teniendo en cuenta esta vez los datos más altos, realizaremos el mismo proceso que en apartados anteriores, obteniendo los siguientes resultados:

```
La clave asociada al valor máximo mediante Upper Star es: occ2017350.csv  con entropía igual a:  [1.75061545]
La clave asociada al valor mínimo mediante Upper Star es: occ2017216.csv  con entropía igual a:  [0.60328698]
```

Figura 8.18: Días de entropía máxima y mínima mediante Upper-Star

```
La matriz con mayor entropía mediante Upper Star cambiando el signo de la matriz es:
[[ 28309 252806  11172   2067    998    783    449    448    397    396
     478    190    165    160    156    122    117     96     84   1976]
 [  5409  32067   1869    295    134    110     47     61     47     30
      32     24     21     30     22     22     19     16     17    325]
 [  2040  10245    436    169   1104     79     46     34     24     25
      19     26     22     14      9      8      7      7     11    118]
 [   974   4337    419    133    101     95     36     34     25     18
      13     11     13     14     13     11     10     10      4    103]
 [   599   2650     82     47     43    125     20     22     19     16
      16     11      5     10      8      5      2      7      3     69]
 [   382   1298     46     36     36    134     36      8     10     11
      10     10      4      7      4      3      3      6      3     59]
 [   254    928     29     19      6     36     14     18     10     12
       3     11      2      6      1      3      0      2      2     48]
 [   178    657     29     13     11     25      8      7     27      9
       6      8      1      5      2      2      4      1      4     35]
 [   131    453     13      7      9     16     11      8      6     17
       4      9      2      2      3      0      0      0      2     24]
 [   162    420      8      4      7     15      1      7      6      7
       8      7      2      2      2      1      3      3      1     23]
 [    94    579      8      4      6     12      4      5      1      5
       4     13      1      3      0      0      1      3      0     18]
 [    80    170      7      5      4      9      2      3      3      2
       3      2      2      1      0      0      0      1      0     16]
 [    69    709      2      4      8      6      4      4      3      3
       1      5      2      5      1      0      0      1      0     11]
 [    48    144      9      4      2      4      0      3      3      4
       3      2      1      0      5      0      0      0      1     15]
 [    47     94      6      0      1     10      4      5      1      4
       0      0      1      1      2      0      0      1      0     16]
 [    33     87      4      0      4      3      1      2      1      4
       1      1      0      1      0      0      1      0      0     11]
 [    29     75      6      2      1      3      6      3      1      2
       3      1      0      3      1      0      1      0      1     10]
 [    32     78      2      2      2      1      1      2      3      2
       0      1      2      1      0      0      1      0      0     14]
 [    22     57      4      0      2      5      2      2      1      1
       0      0      1      0      1      0      0      0      0     13]
 [   869   1360     38     18     26     68     33     23     24     23
      23     21     24     12     10     15      7     10      8    557]]
```

Figura 8.19: Matriz de entropía máxima mediante Upper-Star

```
La matriz con menor entropía mediante Upper Star cambiando el signo de la matriz es:
[[ 19543 136957   5019   1444    641    454    339    294    266    193
     224    164    153    132    146    109     88     88     79    926]
 [  4354  22652   1212    268    113    127     66     56     45     33
      25     12     27     16     19     15     16     15     10    140]
 [  1834   8230   2229    229     59    125     80    223     97     29
      20     12     11      5     16      7      8      6      5     91]
 [  1020   4190    114     57     62    105     47     29     19      8
      12     12      9      5      4      6      2      4      6     76]
 [   676   2389     71     39     31     81     27     22     23     11
       6      8     11      4      3      2      5      3      4     53]
 [   477   1441     58     26     25     60     18     16     14     12
       4      6      6      3      4      4      2      1      0     34]
 [   307    899     43     24     19     26     16     18      7      7
       6      8      1      4      0      2      0      6      2     30]
 [   224    638     27     15     19     20      7     18     15     13
       4      7      4      5      3      2      1      0      3     24]
 [   172    544     21      8     12     12     11     10      5     15
       7      5      4      5      2      2      2      3      1     25]
 [   185    452     24     28      9     12      8      8      7      7
       7      4      5      2      3      3      0      1      1     24]
 [   126    235     19      5      6     11      9      6      5      6
      10      4      2      1      2      0      0      1      2     14]
 [    72    253      7      7      5     10      4      6      4      3
       0      4      3      1      1      1      3      0      0     10]
 [    68    178     28      8     14      7      6      5      6      4
       5      1      3      2      5      0      1      0      0     11]
 [    76    141      8      4      8      3      2      1      2      3
       4      1      1      0      1      1      0      0      0      7]
 [    51    137      5      6      3      4      2      2      1      1
       5      0      0      0      0      1      0      0      1      7]
 [    53    131      6      2      7     29      1      0      3      1
       2      1      2      0      1      0      1      0      0      2]
 [    44     88      3      2      1      1      4      1      1      1
       2      2      1      0      0      0      0      0      0      3]
 [    34     81      3      1      2      1      0      1      4      1
       0      0      0      0      0      0      0      0      0      2]
 [    38     64      1     15      4      4      3      1      1      0
       1      0      1      0      1      1      0      0      0      3]
 [  1083   7716     37     53     20     39     32     19     17     19
      15     12     18     11     14      7      9     11      4    152]]
```

Figura 8.20: Matriz de entropía mínima mediante Upper-Star

En este caso, es más complicado observar la diferencia entre ambas matrices, puesto que estamos estudiando la diferencia entre los mayores elementos de la matriz. Donde sí podemos observar una pequeña diferencia es entre últimos elementos de la primera fila por ejemplo. En la matriz correspondiente a la entropía máxima, estos elementos son bastante más dispares que en la matriz cuya entropía es mínima.

Segundo y tercer experimento

Al igual que con Lower-Star, en estos experimentos, hemos obtenido los mismos resultados que si no hubiéramos normalizado ni multiplicado por una constante.

8.4.4. Resultados teóricos novedosos derivados del análisis realizado

Gracias a los experimentos realizados hemos podido deducir dos proposiciones que se explicarán en este apartado.

Primer resultado - Normalización con Vietoris-Rips

Si observamos los resultados de las entropías calculadas con Vietoris-Rips normalizando las matrices en las figuras 8.6, 8.7, 8.8 y 8.9, nos damos cuenta de que todas las entropías tienen valores muy cercanos al 5,9888... En un principio no sabíamos si podía tratarse de un error en el código, pero estudiando el caso más detenidamente pudimos averiguar la razón del resultado anteriormente mencionado.

Tras comprobar los resultados manualmente descubrimos la razón de los resultados obtenidos. Al normalizar, los elementos de la matriz (de tamaño 20×20) están entre 0 y 1, como sólo algunos de los valores son realmente altos y diferentes al resto, la mayoría de elementos tendrán valores muy cercanos a cero. También debemos tener en cuenta que tal y como está desarrollado el proyecto, con este experimento, sólo normalizamos la última componente de cada elemento de la matriz (ya que convertíamos cada elemento de la matriz al siguiente formato [número de fila, número de columna, elemento]). Es por esto, que la distancia aproximada que hay entre los puntos es de 1 (debido su posición en el espacio por el número de fila y columna). Esto hace que sólo tengamos que engrosar los puntos en 0,5 unidades aproximadamente y por tanto la longitud de las barras estará en torno a esa medida, obteniendo como resultado que su entropía tiene un valor muy cercano a 5,99...

Obtenemos el siguiente resultado teórico novedoso que enunciamos y demostramos a continuación.

Proposición 1: Consideremos una matriz de tamaño $n \times n$ normalizada con valores todos nulos y supongamos que le aplicamos la filtración Vietoris-Rips. Entonces,su entropía persistente es $\ln(n \times n)$.

Demostración: Si aplicamos la fórmula de la entropía persistente que podemos encontrar en el capítulo 5 y sustituimos con los valores adecuados para este caso:

$$E(F) = -\sum_{i=1}^{n\times n} \frac{0,5}{0,5(n\times n)} \ln\left(\frac{0,5}{0,5(n\times n)}\right) = -\sum_{i=1}^{n\times n} \frac{\cancel{0,5}}{\cancel{0,5}(n\times n)} \ln\left(\frac{\cancel{0,5}}{\cancel{0,5}(n\times n)}\right).$$

Puesto que el sumatorio se llevará a cabo tantas veces como elementos tenga la matriz y para todos será igual, obtenemos que:

$$E(F) = -(n\times n)\frac{1}{(n\times n)} \ln\left(\frac{1}{(n\times n)}\right) = -\cancel{(n\times n)}\frac{1}{\cancel{(n\times n)}} \ln\left(\frac{1}{(n\times n)}\right) = \ln(n\times n).$$

Conclusiones: Dado que las matrices tratadas en este proyecto son de 20×20, la entropía tendrá un valor de $\ln(20 \times 20) = 5,99...$ Por lo que el resultado obtenido tiene sentido.Al haber obtenido este valor fue cuando decidimos multiplicar por una constante en el proceso de normalizar la matriz (tercer experimento). Gracias a esto, obtuvimos los resultados reflejados en las figuras 8.10, 8.11 y 8.12 donde ya se puede apreciar diferencia entre las matrices con mayor y menor entropía. En la matriz correspondiente a la mayor entropía podemos ver que hay una menor cantidad de ceros que en la matriz con menor entropía, sobre todo en las primeras columnas.

Segundo resultado - Normalización con Lower-Star y Upper-Star

Al aplicar Lower-Star y Upper-Star, decidimos normalizar puesto que es lo común a la hora de aplicar estas metodologías en el tratamiento de imágenes. Es por eso que se llevaron a cabo el segundo y tercer experimento mediante ambas filtraciones.

Una vez realizados, nos extrañó mucho que obtuviéramos los mismos resultados en los tres experimentos que se habían llevado a cabo mediante estas tres metodologías. Tras la experiencia con Vietoris-Rips decidimos hacer alguna prueba a mano, para ver qué valores obteníamos de la entropía persistente mediante la fórmula descrita en el capítulo 5. Obtuvimos el siguiente resultado teórico novedoso.

Proposición 2: Si aplicamos la filtración Lower-Star o Upper-Star a una matriz de tamaño $n \times n$ habiendo normalizado los valores de dicha matriz entre 0 y 1 y multiplicando por una constante, obtendremos exactamente los mismos resultados que si no normalizamos.

Demostración: Antes de normalizar, los intervalos del barcode serán de la forma $[a_i, b_i]$, pero aplicando la fórmula para normalizar los valores de la matriz, el nuevo intervalo será el siguiente:

$$\left[\frac{a_i - x_{min}}{x_{max} - x_{min}} * c \ , \ \frac{b_i - x_{min}}{x_{max} - x_{min}} * c \right].$$

Para este caso utilizaremos la siguiente fórmula de la entropía (es la misma que la comentada en el capítulo 5 pero expresada de otra forma):

$$E(F) = -\sum_{i=1}^{n \times n} p_i log(p_i) \text{ donde } p_i = \frac{l_i}{S_L} \ , \ l_i = y_i - x_i, \text{ y } S_L = \sum_{j=1}^{n \times n} l_j.$$

Para ir por partes, en primer lugar calcularemos l_i quedándonos lo siguiente:

$$l_i = y_i - x_i = \frac{b_i - x_{min}}{x_{max} - x_{min}} * c - \frac{a_i - x_{min}}{x_{max} - x_{min}} * c = \frac{b_i - a_i}{x_{max} - x_{min}} * c.$$

Y por lo tanto, para calcular S_L:

$$S_L = \sum_{j=1}^{n \times n} l_j = \sum_{j=1}^{n \times n} \frac{b_j - a_j}{x_{max} - x_{min}} * c.$$

Entonces, a la hora de calcular cada uno de los p_i, observamos que obtendremos los mismos p_i normalizando que sin normalizar:

$$p_i = \frac{l_i}{S_L} = \frac{\frac{b_i - a_i}{\cancel{x_{max} - x_{min}}} * \cancel{c}}{\sum_{j=1}^{n \times n} \frac{b_i - a_i}{\cancel{x_{max} - x_{min}}} * \cancel{c}} = \frac{b_i - a_i}{\sum_{j=1}^{n \times n} b_j - a_j}.$$

Conclusión: Gracias a esta demostración evitaremos perder tiempo normalizando los valores de la matriz a tratar.

9. Conclusiones y trabajo futuro

En este capítulo, se aclararán las conclusiones a las que he llegado tras el estudio de las matrices de ocurrencia de Bitcoin, así como de las técnicas que se han utilizado a lo largo del proyecto, además de los resultados del mismo. Otros aspectos que se tratarán en este capítulo serán las desviaciones del planteamiento inicial así como las posibles mejoras para futuras investigaciones.

9.1. Conclusión sobre la investigación realizada

Esta sección expondrá las conclusiones obtenidas en este estudio a partir de los resultados de del mismo, así como las reflexiones personales en relación al proyecto.

9.1.1. Complejidad del proyecto

En primer lugar, con respecto a la complejidad del proyecto, no ha sido muy diferente a mis pensamientos iniciales, puesto que a pesar de la gran cantidad de información que he encontrado sobre Blockchain y Bitcoin y las explicaciones de mis tutoras, hay muchos términos o mecanismos que me resultaron bastante difíciles de comprender. También cabe destacar la sorpresa que me he llevado con la gran cantidad de aplicaciones de Blockchain mas allá de Bitcoin, así como la meticulosidad con la que se ha tenido que implementar esta criptomoneda para que sea perfectamente funcional y segura.

Gracias a las recomendaciones de las tutoras, el principio del proyecto no fue demasiado agobiante, puesto que al tratarse de un área totalmente desconocida para mí, me asustaba perderme entre tanta cantidad de información, pero gracias a sus recomendaciones y a ir trabajando los términos poco a poco no fue tan abrumador como tenía pensado.

Otra de las dificultades al comenzar la etapa de desarrollo del proyecto fue ver de qué forma podía unificar los datos que tenía, para posteriormente utilizarlos ya que los datos de cada día estaban distribuidos en dos archivos diferentes, resultaba bastante complejo. Sin embargo, al percatarnos de que las matrices de ocurrencia ya estaban previamente generadas y disponibles en uno de los repositorios, experimentamos un considerable alivio en relación a esta etapa del proyecto, a pesar de que implicara un cambio de enfoque en nuestra investigación.

Además de estas, una de las principales dificultades para mí fue el comprender los términos de topología que se tratan en este proyecto ya que no había trabajado nunca con ellos. Gracias al curso impartido por Akcora me quedaron claros los conceptos que utilizaba y con los que posteriormente comencé a trabajar.

Como conclusión final, cabe destacar que mi inmersión en este tema ha resultado sumamente enriquecedora desde el punto de vista académico. Además, me gustaría resaltar un aspecto que considero de gran relevancia en esta investigación: la flexibilidad inherente al proceso de investigación. Aunque hayamos tenido que apartarnos del camino inicialmente trazado, esto no ha impedido la obtención de resultados interesantes. Esta experiencia me ha enseñado la importancia de mantener la mente abierta y adaptarse a los cambios para lograr avances significativos en el ámbito de la investigación.

9.1.2. Conclusiones obtenidas a partir del estudio realizado

En este proyecto, se ha investigado la aplicación de ciertas herramientas de topología a las matrices de ocurrencia formadas a partir de los tipos de chainlets de la red de Bitcoin. Con esto buscábamos comprender la estructura topológica de los datos recogidos sobre la red de Bitcoin.

En primer lugar, se llevó a cabo un estudio para comprender los términos involucrados en el mismo, así como el funcionamiento de Bitcoin y la blockchain de Bitcoin. El siguiente paso fue investigar de qué forma podríamos descargarnos los datos de las transacciones de Bitcoin para poder estudiarlos, y es en este punto del proyecto donde nos encontramos con las matrices de ocurrencia de los chainlets de Bitcoin. Siguiendo las recomendaciones de las tutoras del proyecto, nos adentramos en el estudio de estas mediante la homología y entropía persistentes, conceptos que introducidos previamente en el capítulo 5 .

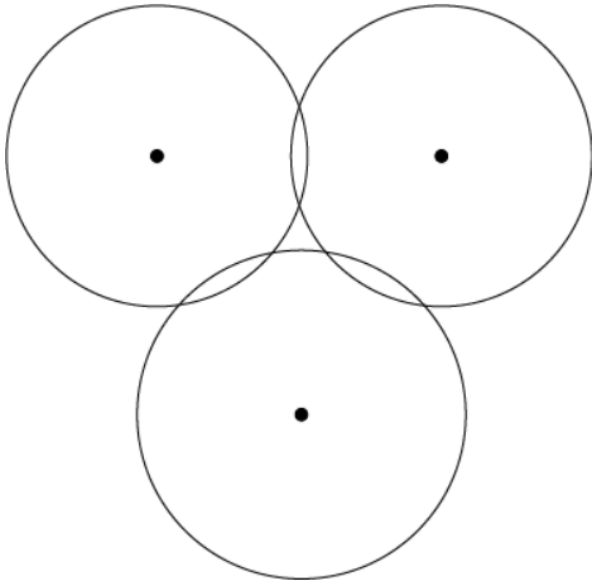

Figura 9.1: Engrosamiento de puntos con Vietoris-Rips

Al haber aplicado tres tipos de filtraciones diferentes para estudiar las matrices de ocurrencia, hemos obtenido información sobre las matrices en las que hay menor número de transacciones de un tipo (gracias a Vietoris-Rips normalizando las matrices y multiplicando por una constante), además de descubrir que a pesar de normalizar

o normalizar y multiplicar por una constante mediante las filtraciones Lower-Star y Upper-Star no habrá diferencia con la entropía sin llevar a cabo este último proceso. Aunque no sean aspectos que podamos relacionar con sucesos en la práctica de Bitcoin, son propiedades interesantes que ser útiles para trabajos futuros.

Gracias a haber aplicado las filtraciones antes mencionadas en los diferentes experimentos, hemos conseguido encontrar dos resultados teóricos novedosos. En primer lugar, hemos averiguado que al normalizar y aplicar la filtración Vietoris Rips deberemos multiplicar por una constante en el proceso de normalización para que la diferencia entre las entropías de las matrices sea apreciable. Finalmente, hemos observado que al utilizar la filtración Lower Star (y, por ende, Upper Star), hemos descubierto que no importa si normalizamos los datos, ya que obtenemos resultados idénticos tanto si lo hacemos como si no.

En conclusión, este estudio ha demostrado que la combinación de la homología persistente y entropía persistente son herramientas útiles para estudiar dichas matrices de ocurrencia aunque aún no hayamos encontrado una aplicación directa en la práctica.

Este proyecto nos ha permitido cubrir una rama que hasta ahora no se había investigado y a partir de ella han surgido otras que se comentarán en la siguiente sección. Además de las que se proponen, existen otras posibles investigaciones que permitirán aplicar otras técnicas topológicas que nos den una visión diferente de las transacciones de Bitcoin.

9.2. Desvíos del planteamiento inicial

Como bien se refleja en el capítulo 2, los objetivos en un principio fueron la creación de un modelo de grafo concreto a partir de de las transacciones de Bitcoin para posteriormente buscar propiedades dentro de él, pero tras toparnos con la dificultad del tratamiento de los datos nos encontramos con la posibilidad de estudiar las matrices de ocurrencia de los chainlets ya formadas.

Este desvío ha provocado que las algunas de las tareas sean diferentes, por lo que se muestra a continuación la lista de tareas que se han realizado finalmente:

1. Seguimiento y comunicación con las tutoras del proyecto.
2. Lectura de los artículos sobre los que basa en gran parte el proyecto.
3. Contacto con el autor principal de los artículos en los que se basa el proyecto y asistencia al curso impartido por él.
4. Investigación sobre Bitcoin, Blockchain y transacciones con criptomonedas.
5. Investigación sobre cómo obtener los datos.
6. Investigación sobre las librerías a usar para el tratamiento de los datos.
7. Extracción de los datos y comprensión de la estructura de los mismos
8. Investigación sobre normalización de las matrices para transformarlas en imágenes.

9. Investigación sobre las posibles técnicas de topología a aplicar a dichas matrices y experimentación con las mismas.
10. Analizar los resultados obtenidos.
11. Redacción de la memoria del proyecto.
12. Realización y preparación de la presentación del proyecto.

Tarea	Tarea(s) predecesora(s)	Aproximación en horas
1	-	15
2	1	10
3	-	10
4	-	20
5	-	20
6	-	20
7	2,4,5	35
8	6,7	35
9	8	40
10	9	45
11	2,10	40
12	11	10

Cuadro 9.1: Tabla de planificación inicial modificada. Se preveen alrededor de 300 horas en total.

Seguidamente, procedemos a presentar una tabla en la que se apreciarán las horas previstas en comparación con las horas que realmente se le han dedicado a cada tarea.

Tarea	Horas estimadas	Horas reales	Porcentaje de desviación
1	15	19	26,67 %
2	10	13	30 %
3	10	7	-30 %
4	20	20	0 %
5	20	25	25 %
6	20	15	-25 %
7	35	35	0 %
8	35	20	-42,86 %
9	40	60	50 %
10	45	40	-11,11 %
11	40	50	25 %
12	10	10	0 %

Cuadro 9.2: Tabla comparativa de cómputo de horas.

En la tabla anterior se puede observar que finalmente ha habido una redistribución de las horas que se iban a dedicar inicialmente a las tareas del proyecto. Puede observarse que finalmente se han dedicado 314 horas a este proyecto que aunque son más de las inicialmente estipuladas, no sobrepasa demasiado las horas que se le iban a dedicar inicialmente.

Como última conclusión añadiría que las tareas de mayor complejidad han sido la extracción de datos hasta que se decidió utilizar las matrices y el desarrollo del código del proyecto así como los experimentos que se han realizado.

9.3. Posibles mejoras para futuras investigaciones

En este apartado se desarrollaran posibles mejoras o continuaciones para futuras investigaciones a partir de ideas que han surgido a lo largo de este proyecto.

En primer lugar, una posible rama de investigación que puede resultar interesante es el estudio de las matrices de cantidad del repositorio de Akcora [11] y propiedades topológicas que pudieran observarse en ellas. Sobre todo, se aconseja el estudio de dichas matrices mediante la filtración Vietoris-Rips, puesto que nos dimos cuenta de que no solo es importante el número de transacciones de cada tipo sino también la cantidad total que transfieren. Se aconseja esta filtración en concreto, puesto que puede ayudarnos a encontrar patrones o agrupaciones en las matrices de cantidad.

En segundo lugar, otra investigación posible podría ser el objetivo inicial de este proyecto, es decir, formar el grafo de transacciones que se propone en el capítulo 4 si se dispone de suficientes recursos y conocimientos como para poder unificar los archivos que contienen la información de entradas y salidas de transacciones de Bitcoin. En el caso contrario, se podría intentar mantener un contacto más contínuo con Akcora para formar el grafo a partir de su código y estudiarlo en busca de propiedades.

Y para finalizar, se podría realizar un análisis de la eficiencia de los algoritmos de minería de Bitcoin, evaluando y comparando posibles implementaciones para determinar cual ofrece un mejor rendimiento.

10. Bibliografía

[1] Cuneyt G. Akcora, Asim Kumer Dey, Yulia R. Gel, and Murat Kantarcioglu. Forecasting bitcoin price with graph chainlets, 2018. URL https://doi.org/10.1007/978-3-319-93040-4_60.

[2] Cuneyt G. Akcora, Yitao Li, Yulia R. Gel, and Murat Kantarcioglu. Topological data analysis for ransomware prediction on the bitcoin blockchain, 2020. URL https://par.nsf.gov/biblio/10189145.

[3] Cuneyt Gurcan Akcora, Yulia R. Gel, and Murat Kantarcioglu. Blockchain: A graph primer, 2017. URL https://doi.org/10.48550/arXiv.1708.08749.

[4] Cuneyt Gurcan Akcora, Nazmiye ceren Abay, Yulia R Gel, Murat Kantarcioglu, Umar Islambekov, Yahui Tian, and Bhavani M. Thuraisingham. Chainnet: Learning on blockchain graphs with topological features, 2019. URL https://doi.org/10.1109/ICDM.2019.00105.

[5] Rocio Gonzalez-Diaz Atienza, Nieves and Matteo Rucco. Persistent entropy for separating topological features from noise in vietoris-rips complexes, 2019. URL https://doi.org/10.48550/arXiv.1701.07857.

[6] Fernando Claros Barrero. Análisis y estudio del grafo de transacciones de bitcoin, 2022. URL https://idus.us.es/bitstream/handle/11441/137229/TFG_Fernando_Claros_Barrero%20(1).pdf?sequence=1.

[7] A. Baumann, B. Fabian, and M. Lischke. Exploring the bitcoin network, 2014. URL https://www.academia.edu/78913661/Exploring_the_Bitcoin_Network.

[8] Martín Heredia Campos. Filtraciones en homología persistente mediante estimadores kernel de densidad, 2018. URL https://ddd.uab.cat/pub/tfg/2018/196556/TFG_Campos_Heredia_Martin_H.pdf.

[9] Dot CSV. Hoy sÍ vas a entender quÉ es el blockchain - (bitcoin, cryptos, nfts y más). *Youtube*, 2021. URL https://www.youtube.com/watch?v=V9Kr2SujqHw.

[10] Murat Kantarcioglu Cuneyt Gurcan Akcora, Yulia R. Gel. Blockchain networks: Data structures of bitcoin, monero, zcash, ethereum, ripple, and iota, 2021. URL https://doi.org/10.1002/widm.1436.

[11] Nazmiye Ceren Abay Cuneyt Gurcan Akcora. Coinworks, 2021. URL https://github.com/cakcora/CoinWorks.git.

[12] Sofokles Dans. Criptografia bitcoin ¿qué es y como funciona?, 2018. URL https://www.cafeconcriptos.com/criptografia-bitcoin/.

[13] Jake Frankenfield. What is botcoin? how to mine, buy and use it, 2023. URL https://www.investopedia.com/terms/b/bitcoin.asp.

[14] Robert Ghrist. Barcodes: The persistent topology of data. *BULLETIN (New Series) OF THE AMERICAN MATHEMATICAL SOCIETY*, 45, 02 2008. doi: 10.1090/S0273-0979-07-01191-3.

[15] Google. Bitcoin-etl library of google, 2022. URL https://github.com/blockchain-etl/bitcoin-etl.

[16] n.d. Web Jobted. Analista programador - salario., 2023. URL https://www.jobted.es/salario/analista-programador.

[17] M. Lischke and B. Fabian. Analyzing the bitcoin network: The first four years, 2016. URL https://doi.org/10.3390/fi8010007.

[18] Margaret Rouse. Computerweekly.es, 2015. URL https://www.computerweekly.com/es/definicion/Planificacion-de-proyectos.

[19] Nathaniel Saul and Chris Tralie. Scikit-tda: Topological data analysis for python, 2019. URL https://doi.org/10.5281/zenodo.2533369.

[20] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, 3(29):925, Sep 2018. doi: 10.21105/joss.00925. URL https://doi.org/10.21105/joss.00925.

[21] Agencia Tributaria. Manual de actividades económicas. obligaciones fiscales de empresarios y profesionales residentes en territorio español, 2023. URL https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/folleto-actividades-economicas/3-impuesto-sobre-renta-personas-fisicas/3_5-estimacion-directa-simplificada/3_5_4-tabla-amortizacion-simplificada.html.