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

Tesis de Licenciatura

Caracterización de la evolución de la blockchain Algorand a partir de datos de transacciones

Juan Cruz Miranda

Director: Esteban Mocskos

23 de septiembre de 2024

Agradecimientos

Allá por los ya lejanos 2016 comencé una nueva aventura, más larga de lo que capaz podría haber imaginado en ese momento. Probablemente, como a todos los que hayan pasado por esta facultad los primeros años fueron duros y me di cuenta que esa recta que me había creado y a la cual me apegaba casi obsesivamente no iba a ser tan recta como yo creía. En un principio capaz luché con esa idea, pero hoy, luego de haber transitado ese camino me doy cuenta de que agradezco que así haya sido. A lo largo de los años una frase de la sabiduría ancestral de nuestros hermanos griegos me acompañó como guía: *si uno sube a Delfos en helicóptero, las ninfas no te dan bola.* Así que no me quedó otra, supongo, como a la montaña o a la vida misma, a la física la debí peregrinar.

Es imposible poner en palabras todas las personas que fueron necesarias para poder completar este trayecto. Personas que, capaz sin darse cuenta, volvieron aunque sea un poquito más sencilla esta epopeya, pero haré mi mejor intento.

Agradezco a mis padres Úrsula y Walter por su apoyo incondicional a lo largo de los años, ellos ya sabían que iba a ser un camino duro y estuvieron a la altura desde su lugar, nadie puede pedir más. Siempre me hicieron sentir infinitamente acompañado, dándome fuerzas para seguir dándole para adelante. Me pongo un poquito nervioso por si no sabían... Vos también Tin te portaste bien...

En materia de amigos, la facultad no ha sido un terreno sencillo de recorrer. Múltiples desafíos a lo largo de los años, cientos de fines de semana pasados en el bendito trono resolviendo problemas y muchos días de mal humor por la cercanía de los exámenes. Creo que sería imposible darle dimensión al fantástico trabajo que han hecho dos familias que me escoltan hoy, compuestas por gente tan buena que acompañó y acompaña sin esperar nada a cambio. Personas que han logrado mantenerme a flote cuando más lo necesitaban, y estaban atentos a cambios en mi comportamiento que capaz ni yo captaba. Nada es más valioso que su tiempo, y ellos eligieron y siguen eligiendo pasarlo conmigo. Eso para mi es la muestra más grande de amor que puede brindarte una persona, y por eso voy a estar eternamente agradecido.

A mi gran amigo Tomás Pujolle, una persona fantástica que me ha acompañado SIEMPRE en los días buenos y malos, predispuesto a hacer todo lo posible para que esté un poquito mejor. Su apoyo ha sido un factor enorme a la hora de lograr sortear esta odisea.

A los Macanudes, mis amigos de la facu, los cuales en el día a día lograban despejarme un poco el cráneo. Gracias por los almuerzos cortitos entre clase y laboratorio,

por bancar los momentos insufribles previos a los parciales, por los viajes, las risas, los días de abuelear con juegos de mesa, las noches en la biblioteca y por también lograr entender a piel propia lo que vivía otro simple estudiante. Gente fantástica que espero que me siga acompañando. Gracias Sol, Franco, Leti, Agus, Lu, Lei y Nico. No se imaginan cuánto han hecho para que esto sea posible.

A mis amigos, los que me vienen acompañando hace tantos años ya, que me llenan de orgullo y que me han enseñado y transformado para llegar a ser lo que soy hoy. Por los asados, las juntadas random, los partidos de futbol, los viajes (ojalá se vengan más) y toda la compañia a lo largo de estos años. Han sido otro de los pilares fundamentales para lograr llegar a donde estoy hoy. Gracias Lopi, Santi, Mole, Pedro, Bruno, Fabri, Pietra, Nico, Erik y Cris.

No puedo dejar de nombrar a personas cuyo apoyo probablemente fue más influyente de lo que capaz ellos mismos creen. Gracias Coki, Pauli, Cata, Fran, Juanfe, Sofi, Joaco, Nay, Nahuel, Cami, Sol, Agusanso, Tam, Juli Hana, Pedro, Cani, Uri, Tomi Pagano, Fusa, Dafi, Guada, Miru, Delfi, Belu alta, Belu corta, Luli y Katia.

De alguna forma este camino lo transitamos todos juntos y no podría haber sido posible sin todas estas personas maravillosas que me rodean.

Índice general

Índice de figuras

Resumen

La aparición de Bitcoin marcó, por primera vez, la posibilidad de tener un sistema distribuido que puede llegar al consenso sin necesidad de tener un tercero de confianza que otorgue validez a las operaciones que se realizan. Una de las principales críticas a Bitcoin es el malgasto de recursos computacionales en su mecanismo de consenso *Proof of Work*, que consiste en la resolución de un *puzzle* computacional para agregar nuevos bloques al sistema. Con la explosión en la popularidad de Bitcoin, aparecieron otras alternativas al consenso que se muestran mucho más eficientes.

En esta tesis, nos enfocamos en Algorand, que presenta un nuevo mecanismo de consenso denominado *Pure Proof of Stake*, derivado del llamado *Proof of Stake*. Este mecanismo ha resultado en una mejora en la eficiencia de utilización de recursos, abordando el desafío del trilema de *blockchain* de manera sostenible y ecológica. Además, a nivel de protocolo, Algorand incorpora diferentes tipos de transacciones, posibilitando la *tokenización de activos* y la creación de *contratos inteligentes*.

La motivación para esta tesis fue la de explorar las transacciones realizadas en este sistema a lo largo del tiempo, buscando caracterizar el comportamiento de la red y la generación de sus estructuras. ¿Hay patrones cohesivos que emerjan en las variables del sistema? ¿Los usuarios se comportan y siguen los mismos circuitos de vinculación entre pares desde que comenzó a funcionar este sistema? ¿Hay algunos tipos de transacciones que tengan mayor capacidad descriptiva para analizar la evolución del sistema? ¿Fue variando la forma en que los usuarios usan este sistema?

Para esto, aprovechamos que las transacciones realizadas son públicas y analizamos conjuntos de datos comparando variables como la actividad de los nodos a lo largo del tiempo, la evolución de la centralidad de cada tipo de transacción y la distribución del grado de los nodos mediante el modelo de Barábasi-Albert. En particular, la existencia del llamado *preferential attachment* en nuestro sistema y su evolución a lo largo del tiempo.

Nuestros análisis revelaron tres períodos temporales cualitativamente distintos de la red. Inicialmente, centrada en intercambios par-a-par entre usuarios y luego, con un mayor enfoque en activos tokenizados y aplicaciones automatizadas. Al ajustar el modelo de Barabási-Albert con nuestros datos, inicialmente identificamos una fuerte correspondencia. Sin embargo, observamos una transición hacia una descripción exponencial denominada *sublinear preferential attachment* a partir del segundo período temporal en adelante. Este cambio cualitativo en el comportamiento del sistema coincide con los resultados previamente obtenidos, señalando una evolución significativa en su dinámica a lo largo del tiempo.

Capítulo 1

Introducción

La tecnología *blockchain* o cadena de bloques realizó su primer aparición en el famoso paper de Satoshi Nakamoto [1] donde presentó una red de transferencia de valor entre pares, utilizando un registro digital transparente y descentralizado protegido por tecnología criptográfica permitiendo transferencias sin permiso o *permissionless* de manera par-a-par (*peer-to-peer*) [2] entre los usuarios eliminando la necesidad de un intermediario.

A partir del movimiento iniciado por Nakamoto luego de presentar su visión en la red Bitcoin, se han desarrollado múltiples nuevas interpretaciones de esta visión original la cuál, hasta el día de hoy, sigue manteniendo su centralidad pero donde poco a poco van apareciendo nuevas redes con tecnología desarrollada en los últimos años generando nuevas propuestas alternativas. Sin embargo, hay muchos conceptos considerados fundamentales que se mantienen constantes a lo largo de cualquier interpretación de este tipo de sistemas. En este capítulo, se procederá a explicar los fundamentos del funcionamiento de una cadena de bloques, haciendo especial hincapié en las propiedades que permitieron su auge en estos últimos años y pasando luego al caso particular de la red estudiada en este trabajo llamada Algorand [3], la cual intenta resolver algunas de las características de diseño consideradas más débiles en la idea original de Nakamoto.

1.1. Tecnología Blockchain y sus principales características

Una *blockchain* [4] es un libro contable público y distribuido en donde se escriben transacciones realizadas por usuarios. Estas propiedades son las que le brindan identidad a estas redes y serán descritas con gran detalle intentando mostrar sus consecuencias inmediatas que asientan a estos sistemas como un cambio de paradigma frente a los métodos de intercambio de valor tradicionales preexistentes.

1.1.1. Seguridad criptográfica

Para realizar una descripción de esta tecnología, es necesario explicar dos conceptos fundamentales que soportaron su desarrollo. Ambos tienen su nacimiento en la criptografía moderna y han sido utilizados para brindar seguridad a este tipo redes. El primer concepto es el de **función de hash**. Las funciones de hash se definen en base a una entrada que es de tamaño variable pero su salida tiene tamaño fijo (el resultado habitualmente se lo conoce directamente como *hash*). Para que una función de hash sea considerada *buena*, tiene que cumplir una serie de características necesarias para asegurar su utilidad como herramienta de seguridad criptográfica. Dichas características son:

- La salida debe poder calcularse de forma rápida.
- Dadas dos entradas casi idénticas, la salida debe ser completamente indistinguible.
- Es de una vía, es decir que es *imposible* (o extremadamente trabajoso) obtener la entrada que generó una salida particular.
- Es determinística, dada la misma entrada siempre se obtendrá la misma salida.
- No debe haber colisiones, para dos entradas distintas, las salidas deben ser distintas.

Este tipo de funciones son utilizadas en muchos de los procesos involucrados dentro del sistema de una red *blockchain* para transmitir información de forma segura.

El segundo concepto que se utiliza en estos sistemas es el de **claves asimétricas**. Éste se basa en la existencia de un par de claves, una pública y una privada, las cuales son generadas en conjunto. Su característica principal es que todo mensaje cifrado por la clave privada puede ser descifrado utilizando la clave pública, y de la misma manera podría realizarse el proceso inverso. Este par de claves se utiliza en la llamada firma digital donde utilizando su clave privada un usuario puede enviar mensajes firmados que pueden ser verificados al utilizar su clave pública para descifrarlos. De esta manera se asegura que, efectivamente, el usuario que envió el mensaje fuera el dueño del par de claves pública/privada. Debido a la posibilidad de generar firmas digitales a partir de este par de claves, es de suma importancia que la clave privada sea un secreto del usuario que no debe revelar a nadie más. Por otro lado, su clave pública es perfectamente visible para todo participante de la red y será utilizada para verificar la veracidad de sus firmas.

En la figura 1.1 se muestra un ejemplo de aplicación del concepto de claves asimétricas al intentar transmitir un bloque de datos de manera segura. En primer lugar, se emplea una función de hash para crear un hash único a partir de toda la información contenida en el bloque de datos. Posteriormente, se utiliza el hash junto con la clave privada del usuario que envía el bloque de datos para generar una firma digital, encriptando el hash obtenido. Los datos originales y la firma digital son

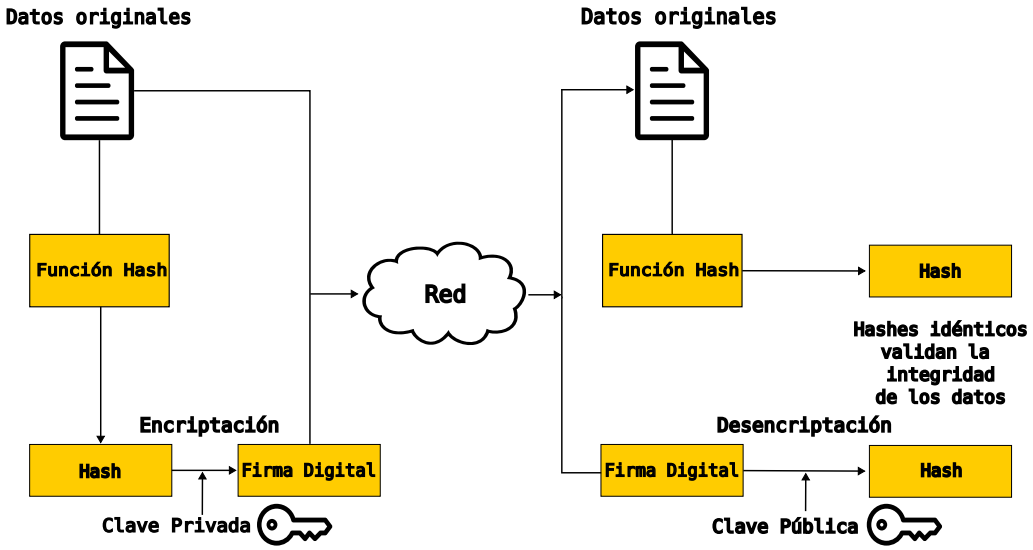


Figura 1.1: Ejemplo de la aplicación del concepto de clave pública y privada.

enviados a través de la red al receptor. Al recibir los datos, el receptor utiliza la firma digital recibida en conjunto con la clave pública asociada a la clave privada original para desencriptar el hash del archivo original. Por último, utilizando nuevamente la función de hash, el receptor genera el hash del bloque de datos recibido y lo compara con el hash desencriptado de la firma digital. Si ambos coinciden, se garantiza la integridad de los archivos transmitidos.

1.1.2. Transacciones, bloques y propiedades fundamentales

Una *blockchain* es un libro contable donde se anotan transacciones de distintos tipos. Al realizar una transacción, una de las posibilidades es transferir valor utilizando la criptomoneda asociada a dicha cadena de bloques (por ejemplo, la criptomoneda asociada a la red Bitcoin es `bitcoin`). Una transacción involucra un emisor (*sender*) y un receptor (*receiver*) y estos son dueños de sus criptomonedas y las intercambian libremente. Con el paso del tiempo, el desarrollo de nuevas aplicaciones y el cambio de visión frente a los posibles usos de esta tecnología llevaron al surgimiento de nuevos tipos de transacciones que serán discutidos más adelante al centrarnos particularmente en la *blockchain* de Algorand.

Dichas transacciones son agrupadas en lo que es conocido como un **bloque**, que se genera por los participantes de la red, llamados **nodos**, a través de un mecanismo de consenso y verificación para evitar actividades fraudulentas. Todas las transacciones que son realizadas entre usuarios y se alojan en los bloques son públicas, es decir, visibles para cualquier persona que desee auditarlas de manera sencilla. Los *nombres* de los participantes de una transacción están representados por un *address* (también llamado billetera o *wallet*) el cual es el *hash* de su clave pública, la cuál se genera a partir de su clave privada que se utiliza para encriptar la información. Debido a que no podemos trazar la pertenencia de una clave pública a una persona física, este tipo

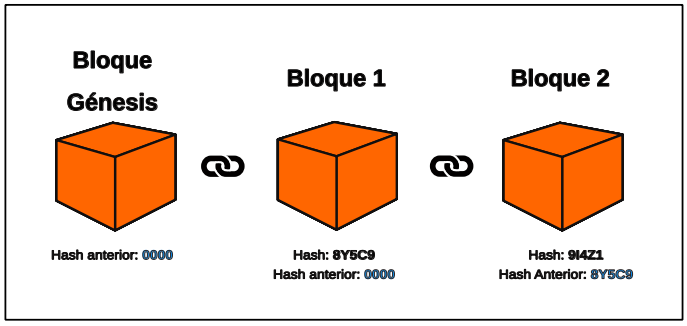


Figura 1.2: La información existente en una *blockchain* es asegurada a través de la generación de una relación criptográfica entre cada bloque consiguiente, donde se utiliza el hash del bloque anterior para obtener el hash del bloque siguiente.

de sistemas son considerados *pseudónimos*, ya que podemos ver las transacciones y direcciones de quienes las realizan, pero no podemos saber quiénes son.

Tras su generación, cada bloque es agregado a la *blockchain* en sí, la cual posee toda la historia de las transacciones realizadas en la red desde su comienzo y está formada por dichos bloques conectados entre sí criptográficamente. Esta conexión se realiza agregando un hash de identificación único a cada bloque generado a partir de la información guardada en el mismo y del hash generado para el bloque que lo antecede en la cadena. De esta manera, se genera una relación bloque padre y bloque hijo ya que el hash generado en cada bloque posterior al primero de la cadena, llamado *bloque génesis*, dependerá del hash obtenido para el bloque anterior. En la figura 1.2 se observan los primeros bloques generados en una *blockchain* con sus respectivos hashes. El hash del bloque génesis es provisto por los creadores del sistema, mientras que los hashes obtenidos para los bloques siguientes son generados utilizando el hash del bloque génesis en el caso del bloque 1 y el hash del bloque 1 en el caso del bloque 2. Es necesario tener en cuenta que, el hash del bloque 2 depende indirectamente del hash del bloque génesis debido a que fue utilizado para el hash del bloque 1. Es decir que dados n y k con $k \leq n$, para una cadena de n bloques, el hash del bloque k, se genera a partir del hash del bloque *k-1* y de la información de las transacciones existentes en el bloque k.

Una de las mayores ventajas de este nuevo paradigma de transferencia de valor es el concepto de **descentralización**. Éste surge a partir de la idea de que todos los nodos poseen una copia idéntica de toda la historia de transacciones realizadas dentro de la red. Es decir que, cualquier persona que desee convertirse en un nodo necesita haber descargado toda la información de los bloques que hayan sucedido previamente a su incorporación a la red. Debido a que la participación en la red es gratuita y abierta, cualquier usuario que desee participar en el sistema de verificación de la red puede montar un nodo, permitiendo una alta descentralización del sistema y por lo tanto un aumento en su nivel de seguridad.

Una conclusión inmediata de esta forma de organizar los datos del sistema y de la descentralización es la propiedad de **incorruptibilidad** de la *blockchain*. En el caso en que un nodo participante del mecanismo de consenso intentara cambiar la

información existente en un bloque anterior al último agregado a la cadena, debido a que los hashes de todos los bloques siguientes al bloque modificado dependen criptográficamente del hash obtenido de dicho bloque, este nodo tendría una versión de la cadena de bloques completamente distinta a la cadena original. Debido a esto, esta desviación de la cadena original producida por dicho nodo sería inmediatamente identificada por otros participantes de la red, discriminando dos cadenas de bloques distintas. De esta manera, se vuelve imposible modificar los datos almacenados en la cadena de bloques. Es decir que, la blockchain posee la propiedad de ser **inmutable**. Una vez que una transacción se encuentra escrita en este libro contable, no puede ser alterada, brindado una mayor transparencia y seguridad a la red.

Sin embargo, en estos sistemas existe la posibilidad de generar cadenas alternativas llamados *forks*. Estos aparecen cuando un participante cuya cadena de bloques no coincide con la aceptada por la mayoría de los nodos decide continuar con su desviación y crear una nueva cadena separada de la original. El sistema no imposibilita este tipo de comportamiento pero éste depende de la capacidad del nodo creador de este *fork* de convencer a otros participantes de que su versión de la cadena es a la que deberían continuar agregándole bloques.

Otra consecuencia del concepto de descentralización fue la **incensurabilidad** de la red, es decir que, la eliminación de un nodo, sea por censura, bloqueo, eliminación forzosa, etc, no afecta el funcionamiento de la red debido a la existencia de múltiples copias de la historia en los demás nodos participantes y la continuidad del mecanismo de consenso y verificación por parte de los nodos remanentes, permitiendo que los usuarios puedan continuar realizando transacciones entre ellos. Esta característica de resiliencia fue uno los componentes necesarios para que esta tecnología se alzara como un competidor frente a los sistemas de transferencia de valor tradicionales, ya que permitía expandir las fronteras de posibilidades al querer transaccionar y nadie podía evitarlo.

Estas características fueron fundamentales para el desarrollo de esta tecnología, ya que, estos sistemas permitían almacenar información de forma segura pero desligándose de cualquier intermediario. En este nuevo paradigma de transferencia de valor, no era necesario que un banco estuviera como intermediario para un intercambio, un usuario ahora podía directamente apoyarse en la seguridad brindada por criptografía y la descentralización de estas redes para realizar sus transacciones.

1.1.3. Mecanismos de consenso y verificación

Los nodos son los participantes del mecanismo de consenso y verificación que aseguran la validez de las transacciones en la red. Se procederá a describir los dos exponentes más importantes de dichos mecanismos de consenso:

- *Proof of Work* [5]: Este mecanismo de consenso fue presentado junto al manuscrito de Satoshi Nakamoto siendo la primer propuesta existente dentro de este nuevo paradigma. El mecanismo consiste en una competencia entre los nodos participantes de la red, generalmente llamados *mineros*. Los *mineros*

luchan para lograr encontrar un hash en particular, limitado a ciertas condiciones manejadas automáticamente por el protocolo. El proceso de generación de un hash debe repetirse una enorme cantidad de veces para lograr arribar a la solución, y como consecuencia la probabilidad de obtener el hash deseado es proporcional al poder de cómputo utilizado.

El nodo que logra resolver dicho problema es el ganador de esta competencia y propone el siguiente bloque de la red y las transacciones dentro del mismo, las cuáles son luego validadas por la red de nodos. Además, por prestar su poder de cómputo en beneficio de la red, es recompensado con nuevas criptomonedas generadas de forma automática por el protocolo y con todas las comisiones pagadas por los usuarios por las transacciones que se incluyeron en dicho bloque. Esto actúa como incentivo para continuar el proceso de generación.

Debido a la existencia de un incentivo por parte de la red a que los mineros provean un poder de cómputo cada vez mayor para poder encontrar el hash ganador antes que otro competidor, el gasto de energía por parte de este mecanismo de consenso crece exponencialmente con el tiempo [6]. Además, el hecho de que toda computación realizada por nodos perdedores durante una ronda en la red no es utilizada para nada que genere valor, llevó a que muchos considerasen este sistema como defectuoso y se buscaran nuevas alternativas que reemplazaran el mecanismo propuesto por Nakamoto.

- *Proof of Stake*: Algunos años después se propuso este mecanismo [7] cuyo principal avance fue el bajo poder de cómputo necesario para su funcionamiento y, por lo tanto, su bajo consumo energético. En este sistema los validadores ponen en juego su capital o *stake* en términos de la criptomoneda asociada a dicha *blockchain* para mostrar que poseen algo de valor en la red que puede ser destruido en caso de que actúen de forma deshonesta. Cada nodo o validador se encarga de que los nuevos bloques se propaguen por la red y, a veces, son elegidos de manera aleatoria para proponer el siguiente bloque. Sin embargo, la probabilidad de ser elegido es proporcional al tamaño del capital puesto en juego por parte del validador. Es decir que, a mayor capital en juego, más grande es la probabilidad de ser seleccionado para proponer el siguiente bloque de la red. Por generar el bloque, al igual que en el mecanismo anterior, el validador recibe una recompensa en la criptomoneda de la cadena de bloques pero en este nuevo sistema no existe la magnitud de desperdicio de recursos de cómputo que tanto se le criticaba a Bitcoin.

  Este sistema tiene en contra la debilidad de que un participante que tuviera una mayoría del *stake*, sería elegido para proponer el siguiente bloque mucho más seguido que un validador con menos *stake*. Dado que el sistema entrega recompensas cada vez que un nodo es elegido para agregar un bloque, llevaría a una paulatina centralización del *stake* total y por lo tanto del proponedor de bloque, lo cuál tendría un impacto en la descentralización y por ende en la seguridad de la red.

1.1.4. Contratos inteligentes

Algunos años después del surgimiento de la *blockchain* de Bitcoin, se generaron nuevas ideas para enriquecer la utilidad de estos sistemas. En particular, como se comentó anteriormente, inicialmente las opciones a la hora de realizar transacciones dentro de una *blockchain* eran muy limitadas. Y, aunque existía la posibilidad de crear *scripts* que brindaban cierta flexibilidad, en la *blockchain* de Bitcoin tenían un alcance limitado dado que no poseía soporte para ciclos, es decir, que no era un lenguaje *Turing completo*. Un grupo de personas se decidió a incorporar un nuevo lenguaje que efectivamente fuera Turing completo y brindara mayores posibilidades a la hora de desarrollar programas y guardarlos directamente en la *blockchain*. Este fue el cambio introducido en la *blockchain* de Ethereum [8] en 2014. *Solidity*, un nuevo lenguaje de computadora, se introdujo para escribir el código que luego era interpretado por la EVM (*Ethereum Virtual Machine*) la cual corría en todos los nodos de la red.

Esta nueva propuesta permitió popularizar la posibilidad de programar en la *blockchain* creando *scripts* complejos, hoy llamados **smart contracts**. Estos nos dan la posibilidad de tener un programa que realice acuerdos entre usuarios sin necesidad de un intermediario y de forma automática. Además, aprovechando las características de estos sistemas, se asegura la integridad del código al ser inmutable una vez que fue incluido en la *blockchain*. Los contratos, al igual que cualquier objeto en este sistema, son públicos y son tratados como una cuenta más, con algunas propiedades alternativas.

La introducción de la posibilidad de almacenar programas de alta complejidad en la *blockchain* fue uno de los avances más fuertes que tuvo esta tecnología y es el motor principal utilizado para el desarrollo de protocolos, proyectos y aplicaciones que a día de hoy se siguen creando de manera continua.

Sin embargo, esta nueva posibilidad también trajo consigo un nuevo desafío. Dado que la capacidad de procesamiento en la red es finita y está vinculada a la capacidad conjunta de los nodos que la respaldan, se hizo necesario establecer un costo asociado a la ejecución de los contratos, conocido como *gas*, que debe ser pagado con la criptomoneda asociada a la red, ALGO en el caso de Algorand. Este costo garantiza que los usuarios solo realicen transacciones necesarias y proporcionales a sus necesidades, ya que, en caso de contar con los fondos necesarios, dicho costo se deducirá automáticamente; de lo contrario, la transacción o ejecución de contrato será rechazado. Esta medida no solo desincentiva la realización de transacciones de *spam* por parte de agentes malintencionados, sino que también previene la ejecución continua de contratos, ya sea debido a un fallo en su diseño o a la decisión de su creador. Así, se evita la sobreutilización de los limitados recursos de la red.

Incluso así, debido al crecimiento exponencial en la popularidad de esta tecnología se han comenzado a presentar en los últimos años problemas relacionados al *gas*. Esto se debe a que el costo de las transacciones se encuentra indirectamente ligado a la congestión que afronta la red, al aumentar el número de transacciones pendientes a escribirse en la cadena de bloques, los usuarios, para intentar que su transacción sea seleccionada por el minero, están dispuestos a pagar una mayor

comisión, aumentando el costo transaccional general de la red. Además, el costo computacional utilizado al comunicarse con estos contratos es mucho mayor que el de una transacción normal. Al momento de la escritura de esta tesis, una transacción realizada en la blockchain de Ethereum puede costar alrededor de 2,00 USD y la interacción con un contrato puede llegar costar incluso veinte veces más.

Este problema ha motivado el desarrollo de nuevas tecnologías que fueran capaces de manejar una mayor cantidad de transacciones por segundo, sin poner en riesgo su seguridad y descentralización. Entre estas nuevas propuestas se encuentra la blockchain de Algorand, en la cual es posible realizar este mismo tipo de operaciones pero por una fracción del costo. La estructura de estos contratos será discutida más adelante al desarrollar su implementación en la red de Algorand, donde son llamados **aplicaciones**.

1.1.5. Tokenización. Tokens fungibles y no fungibles

La implementación de este concepto permitió que, utilizando un programa, cualquier usuario pudiera generar nuevos objetos circulando en la *blockchain*. Es decir que, a día de hoy, no solo se encuentra circulando la criptomoneda de la red de Ethereum llamada `ether`, sino que además hay miles de nuevas monedas llamadas *tokens*, creadas por usuarios. El sistema no impone ninguna restricción a la hora de agregar código dentro de la *blockchain* debido a que es pública y abierta, por lo que mientras se paguen las comisiones necesarias para mantener la red en funcionamiento, la posibilidad de continuar creando nuevos activos es ilimitada. Esto llevó, en particular, al surgimiento de lo que hoy se conoce como *activos tokenizados*, es decir, un activo de la vida real para el cual se genera una representación digital dentro de la *blockchain*. Una de las primeras aplicaciones de esta idea fue la implementación de monedas estables o *stable coins* que replican el valor del dólar 1 a 1 como USD Tether (USDT) o USD Coin (USDC). Pero también, hoy en día se explora la posibilidad de tokenización de otros activos como el oro, el petróleo, los granos e incluso inmuebles, datos médicos [9] y seguros [10].

Este tipo de nuevos activos se dividen en dos categorías, activos fungibles y no fungibles o NFT (*non-fungible token*). La diferencia entre ellos se encuentra en que en los primeros no es posible diferenciar dos *tokens* del mismo tipo. Por ejemplo, en el caso de dos monedas USDT, ambas representan 1 dólar y son perfectamente intercambiables entre sí y nunca podría diferenciarlas. Mientras que en el caso de los NFTs, cada *token* es acompañado por un número de identificación único y público el cuál lo distingue de otros *tokens* de forma determinante.

Inicialmente, uno de los rubros que ganó gran popularidad por la utilización de estos *tokens* no fungibles fue el del arte, especialmente del digital. Éste utilizó esta tecnología para poder brindarle mayores posibilidades a los artistas a la hora de obtener regalías por los intercambios de sus piezas. Esto es factible gracias a que, debido a su identificación única y su existencia dentro de la *blockchain*, es posible mantener el rastro de la pieza al intercambiarse entre diferentes personas y, a través de un *smart contract*, ir obteniendo regalías cada vez que es intercambiada.

Todos los conceptos introducidos hasta ahora, serán de particular relevancia a la hora de realizar el tratamiento de los datos utilizados durante este trabajo. A continuación, se procederá a realizar una introducción de la *blockchain* de Algorand y también una descripción más detallada de algunas de sus propiedades particulares. Entre ellas, los tipos de transacciones existentes en la misma, los cuales serán un elemento central en este trabajo.

1.2. Blockchain de Algorand

A partir de un trabajo teórico publicado por Silvio Micali y Jing Chen en 2017 [11] se creó la *blockchain* de Algorand. Silvio Micali, el fundador de Algorand, es reconocido como uno de los padres de la criptografía moderna. Uno de sus aportes es el cifrado probabilístico, las *verifiable random functions* (VRF) y las pruebas de conocimiento cero (*zero knowledge proofs*, ZKP). Estos descubrimientos fueron puestos en práctica en la *blockchain* de Algorand, de código abierto y cuya criptomoneda nativa es el ALGO. Esta *blockchain* fue lanzada con el objetivo de resolver el conocido trilema de *blockchain* donde se asegura que un sistema no puede tener escalabilidad, seguridad y descentralización de forma simultánea, sino que tiene que elegir dos de esas cualidades. Algorand fue diseñada para poder cumplir estas tres condiciones utilizando su propio sistema de consenso llamado *Pure Proof of Stake* (PPoS) basado en el problema de acuerdos bizantinos.

1.2.1. Verifiable Random Functions (VRF)

El mecanismo de consenso propuesto por Algorand utiliza la VRF [12], una función de código abierto, la cual toma la clave de participación del nodo y una semilla o *seed* de selección, y la utiliza para generar un valor pseudoaleatorio firmado y una prueba, por lo que el resultado puede ser verificado utilizando la clave pública. Usando este mecanismo, que se realiza de manera *offline*, luego envía el mensaje a la red con el resultado obtenido y, dependiendo su valor, el nodo es elegido como participante del comité para seleccionar el bloque ganador de ese ciclo (esto será discutido con más detalle en la sección de PPoS). Este comportamiento es muy relevante a la hora de disminuir los tiempos que toma la red en realizar el proceso para llegar al consenso y, por lo tanto, agiliza el proceso de generación de bloques. Debido a que la función que utilizan todos los nodos es la misma, es fácil verificar el resultado obtenido por uno de los nodos sin necesidad de información adicional más allá del mismo mensaje recibido.

1.2.2. Pure Proof of Stake (PPoS)

El mecanismo de consenso de Algorand llamado *Pure Proof of Stake* (de aquí en adelante PPoS) está inspirado en el mecanismo de *Proof of Stake* pero con cualidades particulares que lo diferencian de los demás. Consiste en varias etapas en las cuales se rotan sus participantes para que los individuos con un menor *stake* tengan también

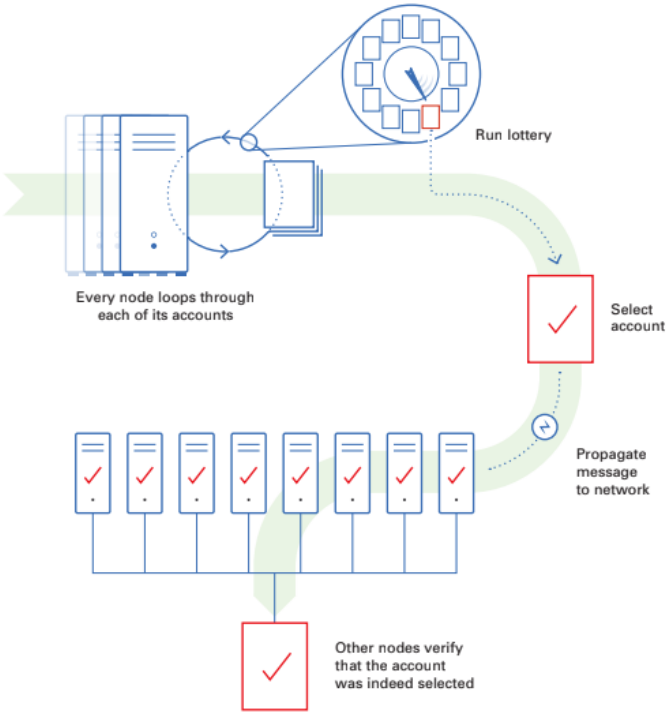


Figura 1.3: En esta primera etapa, todos los nodos corren las VRF para cada una de las cuentas que se encuentran registradas en dicho nodo que posean *stake* habilitado para participar en el consenso. En caso de resultar ganador, envían su propuesta de bloque junto con la verificación de que efectivamente fueron ganadores.

la chance de participar en el mecanismo de consenso. Dichas etapas serán descritas a continuación.

Primera etapa Se observa en la figura 1.3. Todos los nodos ejecutan su función VRF para cada cuenta que se encuentra en el nodo de manera *offline*. Utilizando como *seed* un parámetro que se encuentra en el bloque anterior al que se está por proponer, imposibilitando hacer un cálculo previo ya que dicha *seed* no es conocida hasta un momento antes de comenzar a ejecutar la función. La VRF se ejecuta tantas veces como unidades de la moneda Algo posea la cuenta. Es decir, que una cuenta con 1 Algo ejecuta la función una sola vez mientras que una cuenta con 1000 Algo la ejecuta mil veces, teniendo una posibilidad mucho mayor de ser elegida como participante del mecanismo de consenso pero manteniendo la chance de ser elegido aún teniendo un muy bajo *stake*.

Dependiendo de su resultado, el nodo es seleccionado para proponer un posible próximo bloque utilizando sus transacciones conocidas y, además, envía la prueba de que efectivamente ha sido seleccionado. Dado que muchas cuentas serán ganadoras, múltiples bloques serán nominados como posibles próximos bloques a ser agregados en la cadena.

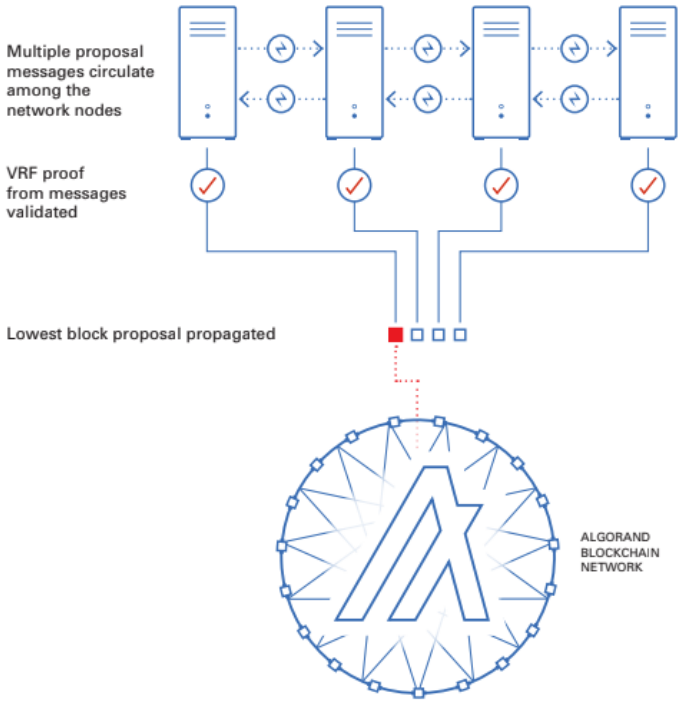


Figura 1.4: En esta segunda etapa los nodos se ponen de acuerdo para propagar a los demás los bloques propuestos con menor hash. Luego de un tiempo, terminan de filtrar todos hasta que solo queda uno el cuál pasa a la siguiente etapa del consenso

Segunda etapa Se ejemplifica en la figura 1.4. Los bloques propuestos comienzan a transmitirse hacia toda la red de nodos, pero presentándose a ellos un criterio de eliminación completamente arbitrario elegido para el protocolo el cual consiste en solo transmitir el bloque con el *hash* de VRF más pequeño. La red mantiene esta transmisión de manera continua por un tiempo determinado por el protocolo. Sin embargo, no necesariamente se asegura de que todos los nodos se encuentren de acuerdo con respecto a qué bloque es el que posee el menor hash, debido al tiempo de propagación de la red.

Posterior a este proceso, se genera un comité utilizando de nuevo la función VRF para seleccionar a sus miembros y estos votan de forma pesada el bloque seleccionado según la cantidad de moneda que poseen. Luego de llegar a un quorum de votos necesarios, se avanza a la siguiente etapa.

Tercera etapa Se muestra en la figura 1.5. En esta etapa cada nodo evaluará la VRF sobre cada cuenta participante que administran para ver si quedan seleccionados para participar en el llamado comité de voto suave o *soft vote*. En caso de ser elegida la cuenta enviará un voto pesado dependiendo de la cantidad de moneda que posea. Esos votos apoyarán el bloque de menor hash que ha recibido dicho nodo en particular. Un quorum de votos es necesario para pasar al siguiente paso y debe superar un determinado porcentaje del tamaño total del comité. Los votos serán recibidos por otros nodos en la red y cada nodo validará la selección de los

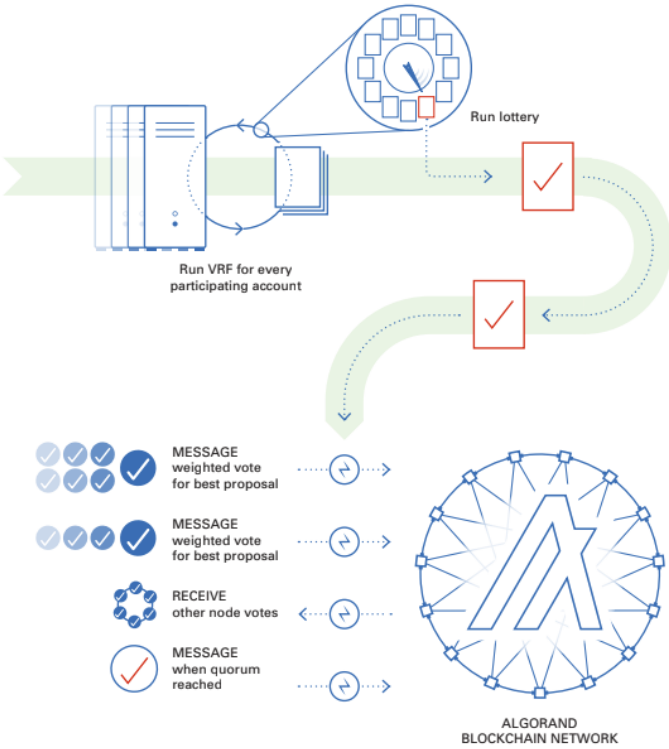


Figura 1.5: Durante esta tercer etapa, múltiples comités son formados los cuales contabilizan la validez del bloque propuesto antes de agregar el nuevo bloque a la cadena.

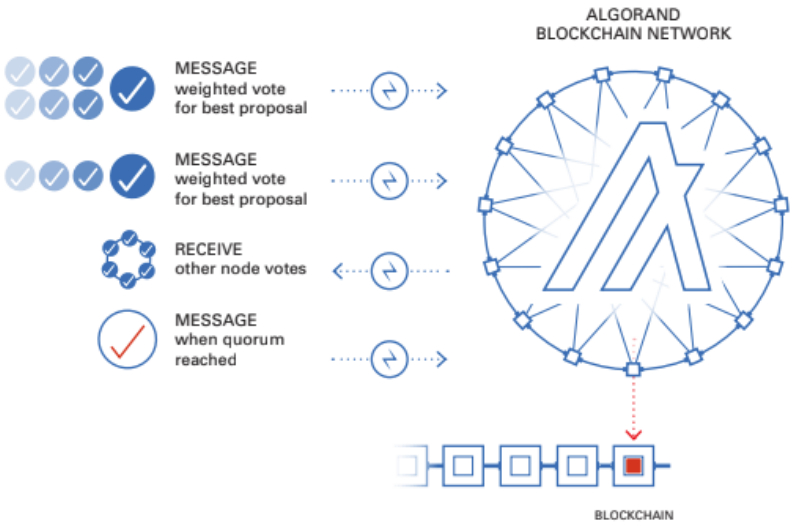


Figura 1.6: En esta última etapa, luego de un proceso de verificación y certificación, el bloque es añadido a la cadena y se reinicia el proceso.

miembros del comité resultante de la VRF antes de agregar sus votos al total. Una vez logrado el quorum necesario, se avanza a la siguiente etapa.

Cuarta Etapa Se muestra en la figura 1.6. Un nuevo comité chequea el bloque propuesto para casos de *overspending* (gastar de más), *double spending* (gastar dos veces la misma criptomoneda) o cualquier otro problema que pudiera llegar a surgir en el bloque. Si el bloque es considerado válido un nuevo comité vuelve a votar para certificar el bloque y agregarlo a la cadena. Pasado ese momento una nueva ronda inicia y el proceso comienza de nuevo. En el caso de que no se llegue a un acuerdo en el tiempo estipulado para el comité de certificación, el sistema entrará en un modo de recuperación.

1.2.3. Red principal y red de prueba

A la hora de interactuar con el sistema de Algorand, se encuentran dos opciones disponibles. La red principal, llamada *MainNet* con número de identificación *ID* = `mainnet-v1.0`, y una red de prueba conocida como *TestNet* con *ID* = `testnet-v1.0`. Aunque ambas redes funcionan de manera similar, la diferencia fundamental entre ambas se halla en que, en el caso de la red de prueba, es posible obtener ALGO para realizar el pago de los costos de transacción de manera gratuita a través de los llamados *Faucets* como Algorand Dispenser, por lo que provee a los desarrolladores un ambiente seguro para poder probar sus productos, identificar bugs y probar nuevas características de manera rigurosa pero sin consecuencias en el mundo real. Por otro lado, en la *MainNet* es donde se realizan las transferencias reales de activos digitales, la criptomoneda utilizada para pagar el costo de dichas transacciones tiene valor y estas quedan almacenadas en la blockchain de Algorand.

Los datos obtenidos y utilizados a lo largo de este trabajo para hallar información sobre la evolución de la red, fueron tomados de los bloques existentes en la *MainNet* de la blockchain de Algorand.

1.2.4. Algorand Standard Assets (ASAs)

La *blockchain* de Algorand soporta la creación de activos *on-chain* llamados *Algorand Standard Assets* o ASAs los cuales se benefician de la misma seguridad, compatibilidad, velocidad y facilidad de uso que la criptomoneda Algo. En Algorand no hay una distinción estricta entre un *token* fungible y no fungible, sino que ambos son ASAs con diferentes especificaciones en el momento de su creación. Estos activos puede ser utilizados para representar monedas estables, puntos de lealtad de una empresa, puntos de un juego pero también las escrituras de casas, objetos coleccionables, partes únicas de una cadena de suministros, etc. A la hora de crear un activo en Algorand, se deberán brindar una serie de parámetros los cuales serán importantes para determinar qué tipo de activo se está buscando crear. Dichos parámetros son:

- *Creator*: Se define el creador de los activos.
- *AssetName*: Nombre del activo, opcional pero recomendado.
- *UnitName*: Nombre de la unidad, opcional pero recomendado.
- *Total*: Total de unidades a crear.
- *Decimals*: Número de decimales.
- *DefaultFrozen*: Verdadero para congelar las tenencias de este activo por default.
- *URL*: URL del activo, completamente opcional.
- *MetaDataHash*: Hash de la *metadata*, opcional.

Además hay cuatro parámetros que, en comparación a los anteriores, podrán ser cambiados posteriormente a la creación del activo y le darán un mayor control del activo al creador. Estos parámetros son:

- *Manager Address*: La cuenta *manager* puede autorizar transacciones para reconfigurar o destruir un activo.
- *Reserve Account*: La cuenta reserva guarda las unidades del activo llamadas de reserva, o que aún no han sido puestas en circulación dentro de la *blockchain*.
- *Freeze Account*: Esta cuenta es capaz de congelar o descongelar el activo para otra cuenta. Una cuenta congelada no puede enviar o recibir el activo congelado.

- *Clawback*: La cuenta *clawback* es capaz de transferir activos desde y hacia cualquier tenedor del activo. En general se utiliza para quitar los activos de una cuenta particular.

Para poder comenzar a utilizar un activo es necesario, previamente, haber realizado una transacción de *opt-in*, dándole acceso a esa *wallet* a determinado activo. Dicha transacción será discutida con mayor detalle en la siguiente sección.

1.2.5. Bloques en Algorand

Los bloques en Algorand, al igual que en todas las *blockchains*, son la estructura a partir de la cual se organizan los datos de las transacciones. Cada bloque se ordena dentro de la cadena referenciando al bloque anterior y es el resultado del mecanismo de consenso descrito anteriormente. Dentro de cada bloque se especifican múltiples parámetros con numerosa información, que será detallada y discutida a continuación.

- *earn*: Especifica cuantas recompensas se han entregado hasta ese momento.
- *fees*: Aquí se encuentra el *address* a donde todas las comisiones o *fees* de transacción son enviadas.
- *frac*: Número de microALGO (µAlgo) sobrantes luego de la distribución de *RewardsRate*/*rewardUnits*.
- *gen*: Se define el *genesis Id*, en caso de encontrarse en la *mainnet* lo esperado es `mainnet-v1.0`.
- *gh*: El hash del *genesis Id*.
- *prev*: Hash del bloque anterior.
- *proto*: Protocolo utilizado actualmente.
- *rnd*: Número de ronda, equivalente a número de bloque. Este parámetro es utilizado a lo largo de toda la tesis para determinar el número de bloque.
- *rwcalr*: La ronda en la cual el *RewardsRate* será recalculado.
- *rwd*: Contiene el *address* del pool de *rewards* el cual se utiliza para distribuir las recompensas.
- *seed*: Esta semilla es utilizada en el mecanismo de consenso junto con las VRF.
- *spt*: Se mantiene un seguimiento de la próxima ronda en la que se hará una transacción de *state proof*. Esta transacción aún no ha sido descrita en esta tesis pero se verá en la sección de transacciones.
- *tc*: Contador de transacciones desde el principio de la *blockchain*

- *ts*: *Timestamp*, utilizado para poder saber el tiempo del bloque.
- *txn* y *txn256*: Ambos son hashes los cuales son utilizados para asegurar la integridad de la información de transacciones dentro del bloque usando *merkle trees* [13].
- *txns*: Lista de transacciones dentro del bloque.

1.2.6. Transacciones en Algorand

Uno de los temas centrales de esta tesis fue el análisis de la distribución de las transacciones en la red. Por lo tanto, en esta sección se describirá con detalle los tipos de transacciones posibles dentro de la red de Algorand y su respectiva *metadata*, la cual fue utilizada para identificar cada transacción, poder clasificarla y, posteriormente, analizarla. A lo largo de esta tesis se encontraron algunas particularidades dentro de cada tipo de transacción que también serán descritas. Para poder diferenciar cada tipo se utilizó un parámetro llamado `type` el cuál define qué tipo de transacción se está realizando.

Payment

Este tipo de transacción es utilizada para transferir `ALGO` de una cuenta a otra y en el caso de que se quisiera eliminar una cuenta. La manera de distinguir entre estos dos tipos de transacciones es a través de la *metadata* que la acompaña.

- *amt*: Cantidad de moneda a enviar, en millonésima de Algo (µAlgo). Es decir, si quisiera enviar 1 Algo pondría 1 000 000.
- *fee*: Cantidad de comisión cobrada por la transacción en millonésima de Algo (µAlgo).
- *fv*: Cota inferior del número de ronda donde la transacción es considerada válida.
- *gen*: *Genesis Id*, en el caso de que la transacción se encuentre en la *mainnet* debería ser `mainnet-v1.0`.
- *gh*: Hash del *genesis Id*.
- *lv*: Cota superior del número de ronda donde la transacción es considerada válida.
- *note*: Nota opcional, representado por *base64-encoded bytes* del texto deseado.
- *rcv*: *Address* receptora de la transacción o *receiver* que recibe los Algo envíados.
- *snd*: *Address* emisor de la transacción o *sender* que envía los Algo de la transacción.

- *type*: Tipo de transacción realizada. En este caso se utiliza *pay*.

Durante el desarrollo de esta tesis se encontraron transacciones de tipo `pay` donde el parámetro `rcv` no se encontraba dentro de la transacción. Consultando con *developer relations* de Algorand, se confirmó que eso sucede al seleccionar la cuenta receptora como el `address` cero global, una cuenta utilizada para "quemar" *tokens*. Es decir, que no se encuentra bajo el control de nadie y las criptomonedas enviadas a dicha cuenta son consideradas perdidas, atrapadas o inutilizadas.

El segundo tipo de transacción posible dentro del tipo `pay` es el utilizado para cerrar una cuenta. Debido a que para que una cuenta dentro de Algorand sea considerada activa necesita tener un balance mínimo, es necesario realizar una transacción de `close` para poder desactivarla completamente. La *metadata* que acompaña a esta transacción es igual a la anterior pero con un parámetro extra.

- *close*: Se agrega este parámetro para identificar que la transacción es de tipo `close` que coincide con el `address` de la cuenta receptora de la transacción.

Debido al bajo número de transacciones de tipo `close` observadas durante esta tesis, este tipo de transacción no fue de carácter central en los análisis realizados.

Key Registration

Este tipo de transacción es utilizada para cambiar el estado de una cuenta con respecto a su participación en el consenso de la red de Algorand. Una cuenta puede encontrarse *online* u *offline*. Previo al envío de este tipo de transacción la cuenta debe generar una clave de participación. Al igual que para las transacciones de tipo `pay`, también se encuentra acompañada de *metadata*.

La información que acompaña esta transacción es la misma que el tipo anterior, agregando cuatro parámetros nuevos en las transacciones que desean cambiar el estado a *online*. Los parámetros `selkey`, `votefst`, `votekd` y `votekey` son completados a partir de la información obtenida de la clave de participación en el nodo donde se encuentra registrada dicha cuenta. Además el tipo de transacción que se utiliza en el parámetro `type` es `keyreg`. En el caso de desear marcar una cuenta como *offline* es necesario simplemente enviar este tipo de transacción, pero sin utilizar los cuatro parámetros específicos mencionados anteriormente. Utilizando el parámetro `type` pudimos obtener la cantidad de transacciones de tipo `keyreg` que se realizaron en los bloques analizados.

Asset Configuration

Este tipo de transacción es utilizada al crear, modificar algún parámetro de un activo o directamente destruirlo. Debido a que la *metadata* que identifica cada uno de estos subtipos de transacción es distinta, será descrita a continuación con gran detalle, ya que estas diferencias son las que posibilitan su posterior identificación y clasificación. En los tres casos de este tipo de transacción, el `type` se identifica con `acfg`.

La transacción de creación tiene la particularidad de llevar toda la información para un nuevo activo dentro de un parámetro llamado `apar`, de *asset parameters*, donde se encuentran todos los parámetros mencionados anteriormente en la sección de ASAs. Entre ellos se encuentran las cuentas de `manager`, `clawback`, `freeze` y `reserve`. Otra particularidad de este tipo de transacciones es que no se encuentra el parámetro de `receiver`. Al enviar esta transacción se genera un nuevo activo en la red identificado con un `Id` creado en ese momento. A partir de su creación, será necesario utilizar ese `Id` para poder realizar cambios en ese activo. A partir de la aparición del parámetro `apar` y la falta del parámetro `Id` se puede categorizar esta transacción como una de creación.

En los otros dos tipos de transacciones habrá un parámetro nuevo el cual se encargará de identificar el activo con el que se esta interactuando, este parámetro es `caid` donde se encontrará el `Id` del activo creado. En el caso de una reconfiguración, el parámetro `apar` se encontrará debido a que los cambios se realizan sobre las cuentas `manager`, `clawback`, `freeze` y `reserve` las cuales son las únicas que puede ser modificadas posterior a la creación del activo. Por lo tanto para identificar una transacción como una de modificación de parámetros, basta con que se encuentren los parámetros `apar` y `caid` dentro de la transacción.

El último caso, la transacción de destrucción de un activo, solo se puede realizar por la cuenta `manager` y siempre y cuando todos los activos se encuentren bajo la propiedad de esta cuenta. En ese caso, la transacción es acompañada por el parámetro `caid`, pero la falta del parámetro `apar` vuelve fácil su posterior identificación.

Asset Transfer

Este tipo de transacción es utilizada para transferir o revocar algún tipo de *Algorand Standard Asset* y para realizar *opt-in*, un tipo de transacción particular que será explicada a continuación. Al igual que en las transacciones anteriores, es posible diferenciar cada tipo de transacción a partir de la *metadata* que la acompaña pero en todos los casos el parámetro `type` será `axfer`.

Debido a los bajos costos de operación dentro de la red de Algorand, menores al centavo de dólar al momento de la escritura de esta tesis, una vulnerabilidad existente en el sistema se halla en la posibilidad de realizar múltiples transacciones enviando "activos basura" hacia cuentas existentes en la red. Para defenderse frente a este tipo de ataque, se creó un tipo de transacción llamada *opt-in*, la cual es la primer transacción que debe realizarse para poder comenzar a interactuar con un tipo de activo que no sea la criptomoneda nativa de la red Algo. Es decir, excepto por el Algo, todos los activos dentro de la red se encuentran "no disponibles" para una cuenta que no haya realizado *opt-in* en ningún activo y no será capaz de recibirlos. Posterior a haber realizado esta transacción, dicha cuenta podrá comenzar a realizar las otras transacciones existentes del tipo *asset transfer* con el activo aceptado. La transacción de *opt-in* tendrá la particularidad de llevar los parámetros `xaid` para identificar el `Id` del activo al que se desea hacer *opt-in* y `arcv` para determinar la cuenta que está realizando el *opt-in*, la cual coincidirá con la cuenta `snd` o `sender`.

Al realizar la transferencia de un activo, será necesario especificar la cantidad a

través del parámetro `aamt`, de *asset amount*, el `xaid` y el `arcv`. Debido a la aparición de este nuevo parámetro `aamt`, es posible diferenciar ambos tipos de transacciones para su posterior clasificación.

Por último, en caso de querer utilizar la función de `clawback` para retirarle a una cuenta la posesión de algún activo, será necesario especificar quién será la cuenta que enviará el activo, es decir la cuenta a la cual le estarán revocando el acceso a dicho ASA, y también especificar que cuenta será la que recibirá dichos activos revocados. Esto será especificado en los parámetros `asnd` para el participante cuyos activos son revocados, y `arcv` para el que los recibe. Utilizando esta distinción logramos identificar inequívocamente este tipo de transacción.

Asset Freeze

Este tipo de transacción solo puede ser realizada por la cuenta especificada como la cuenta `freeze` de un activo. El receptor de esta transacción pierde la posibilidad de enviar o recibir el activo congelado. Debido a que es la única transacción cuyo parámetro `type` es `afrz`, es fácilmente identificada. La cuenta que luego de la transacción tiene sus activos congelados es identificada a través del parámetro `fadd`, de *frozen address*.

Application Call

Este tipo de transacción es utilizada al crear o interactuar con una aplicación dentro de la *blockchain* de Algorand. Una aplicación en Algorand, es lo que anteriormente había sido definido en general como un contrato inteligente o *smart contract*. Las transacciones de este tipo serán identificadas en el parámetro `type` con `appl`. Existen varios tipos de transacciones posibles, en esta tesis fueron divididas en dos grupos. El primero, solo con la transacción de creación de aplicación caracterizada por la falta de un parámetro llamado `apid`. Y por otro lado, los demás casos, incluyendo actualización y borrado de aplicación, `opt-in, close-out` y `clear state`. En estos casos, este parámetro se encuentra dentro de la *metadata* asociada a la transacción pudiendo entonces diferenciar estos tipos de transacción con el grupo. Cada uno, a la vez, es identificado unívocamente por un número de 0 a 5 en el parámetro `apan`.

State Proof

Este tipo de transacción es realizada de forma automática por el protocolo cada una cierta cantidad de bloques predeterminada luego del proceso para llegar al consenso. Al no poder ser realizada ni por aplicaciones ni por individuos, no fue tenida en cuenta a lo largo de esta tesis como un dato del comportamiento general de la red.

1.3. Marco teórico

Habiendo discutido los distintos tipos de transacciones posibles, se procederá a introducir los elementos teóricos necesarios utilizados para describir algunas variables dentro de una red *blockchain*, y los modelos posteriormente utilizados.

Para lograr representar las relaciones formadas entre los participantes de la red a lo largo del tiempo se utilizaron *grafos*. Un *grafo* G es un par ordenado $G = (N, A)$, donde N es un conjunto de *nodos* o vértices, que se encuentran unidos por un conjunto de enlaces A llamados *aristas*. Esta herramienta matemática facilita el estudio de las relaciones entre unidades que interactúan unas con otras y sus propiedades matemáticas se encuentran profundamente estudiadas. Adicionalmente, poseen la característica de ser fácilmente visualizados de una manera intuitiva, lo que permite la exploración de los datos.

En el sistema estudiado en esta tesis, cada *address* de un usuario de la red fue representada con un nodo en un grafo y cada transacción realizada por dicha cuenta con una arista hacia otro nodo. De esta manera se generó un grafo G que incluía ambos conjuntos dentro de la red, los participantes y las transacciones realizadas entre los mismos. La utilización de dicha herramienta matemática nos permitió utilizar sus propiedades para realizar una descripción profunda de las relaciones entre los participantes de la red. Algunas de sus propiedades fundamentales que fueron aprovechadas a lo largo de esta investigación se describirán a continuación. Los grafos permiten no solo representar qué nodos se encontraron involucrados en una transacción, sino también en qué sentido se realizó la misma.

En un grafo direccionado es posible discriminar el nodo de origen y el de destino de un vértice. En nuestro caso, en todas las transacciones realizadas en la red se encontraban discriminados el emisor y el receptor, por lo que consideramos un grafo dirigido para representarla.

Una segunda característica deriva de la interconexión entre los nodos de la red. Un grafo es considerado **conexo** si para dos nodos cualesquiera existe un camino que los conecte, pasando por otros nodos. Cada conjunto de nodos conexos dentro de un grafo es considerado un **componente**. En el caso de que existan múltiples componentes dentro de un mismo grafo, este es llamado *no conexo*. En la figura 1.7 se encuentra un ejemplo de un grafo conexo y uno no conexo. En el caso del conexo se puede observar que para todos los nodos dentro del grafo existe un camino que los conecte entre sí, mientras que en el caso del no conexo, es posible encontrar un camino para conectar algunos de los nodos pero no todos los que se encuentran dentro del conjunto. Debido a que tenemos dos subgrupos o componentes que conviven dentro del mismo podemos decir que es no conexo. Esta característica de dividir al grafo en componentes será crucial a la hora de definir qué participantes de la red son importantes dentro de nuestra investigación y cuáles deben ser eliminados en un proceso de filtrado. Esto será discutido más adelante al hablar del algoritmo de filtrado.

Dentro de un grafo con N nodos, el nodo *i-ésimo* se dice que tiene grado k_i si posee k aristas hacia otros nodos de la red. Por lo tanto, el número total de aristas

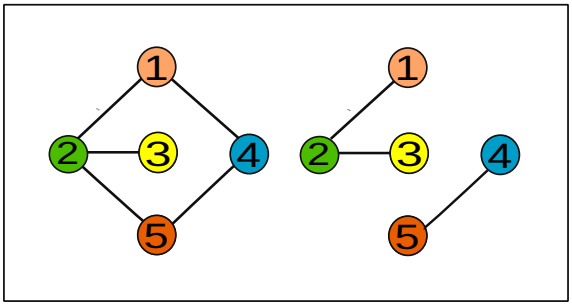


Figura 1.7: Del lado izquierdo, un grafo completamente conexo, del lado derecho uno no conexo con múltiples componentes.

L debe cumplir:

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i \tag{1.1}$$

donde se agregó el $\frac{1}{2}$ debido a que las aristas serán contadas dos veces (desde los dos extremos de la misma).

Utilizando el valor de L se puede calcular una propiedad importante de la red que es el grado promedio, el cual se calcula a partir de:

$$\langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2L}{N}. \tag{1.2}$$

Una parte importante de este trabajo consistió en calcular la distribución de probabilidad p_k, la cual provee la probabilidad de que un nodo seleccionado al azar de la red tenga grado k. Ésta se calcula a partir de:

$$p_k = \frac{N_k}{N} \tag{1.3}$$

siendo N_k el número de nodos con grado k, y N el número total de nodos.

1.3.1. Redes aleatorias y libres de escala

Una posibilidad a la hora de reproducir redes de la vida real es utilizar la teoría de redes aleatorias o *random networks* [14], teoría desarrollada originalmente por Erdös y Rényi. La contrucción de dichas redes teóricas se realizan de la siguiente manera:

- Comienzo con N nodos aislados.
- Selecciono un par de nodos y genero un número aleatorio de 0 a 1. Si el número excede el valor de un parámetro p, conecto ambos nodos con una arista, sino los dejo desconectados.

- Se repite este proceso para los $\frac{N(N-1)}{2}$ pares de nodos.

En este tipo de redes, el número total de aristas L es aleatorio. La probabilidad de que una realización de la red posea L aristas está dada por

$$p_L = \binom{\frac{N(N-1)}{2}}{L} p^L (1-p)^{\frac{N(N-1)}{2}-L} \tag{1.4}$$

una distribución binomial. El número de aristas esperadas es

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L P_L = p\frac{N(N-1)}{2}, \tag{1.5}$$

obtenida a partir de la probabilidad p y del número de pares que intentamos conectar. Usando la ecuación (1.2), obtenemos el valor medio de k para este tipo de sistemas dado por los valores de p y de *N-1* el cuál es el máximo número de aristas que puede tener un nodo en una red de N nodos.

$$\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N-1) \tag{1.6}$$

La distribución del grado de los nodos para este tipo de redes está dada por la binomial

$$p_k = \binom{N-1}{k} p^k (1-p)^{N-1-k} \tag{1.7}$$

donde la forma de esta distribución depende de los valores de p y de N.

Debido a que en la mayoría de las redes reales se cumple $\langle k \rangle \ll N$ este tipo de distribución puede ser bien aproximada por la distribución de Poisson

$$p_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \tag{1.8}$$

cuyo único parámetro es $\langle k \rangle$ en vez de p y N como sucedía para la binomial.

Sin embargo, en múltiples sistemas, este tipo de aproximación a redes reales no suele ser la que se ajusta mejor a los datos. Otro de los modelos utilizados que suele aproximar una red real, como puede ser el ejemplo de Internet, es la llamada distribución en **ley de potencias** o *power law distribution*

$$p_k \sim k^{-\gamma} \tag{1.9}$$

cuyo parámetro γ es el grado del exponente. Tomando el logaritmo se obtiene

$$\log p_k \sim -\gamma \log k \tag{1.10}$$

donde se consigue una relación lineal para el logaritmo de k con su pendiente dada por el parámetro γ. Esto es importante debido a que los gráficos que se utilizarán a lo largo de esta tesis se encontrarán en escala logarítmica. En el caso de que la distribución de grados de una red se encuentre dada por una ley de potencias y que

el parámetro γ se encuentre entre los valores de 2 a 3, esta red es llamada *libre de escala* o *scale free*.

La diferencia con la distribución de Poisson presentada anteriormente se encuentra en que, para un valor de k pequeño, la ley de potencias supera el valor de la de Poisson, indicando un mayor número de nodos de grado bajo. También, para los k en la vecindad de $\langle k \rangle$, la distribución de Poisson se encuentra arriba de la ley de potencias, sugiriendo un exceso de nodos con grado $k \sim \langle k \rangle$. Por último, una diferencia particularmente visible para valores de k grandes, la ley de potencias vuelve a encontrarse por arriba de Poisson, indicando que la probabilidad de observar *hubs*, es decir, nodos con un valor de k muy alto, es múltiples órdenes de magnitud mayor en una red libre de escala en comparación con una red aleatoria.

En las redes descritas por el modelo de ley de potencias el parámetro γ controla el comportamiento de la red. Se observan tres regímenes posibles dependiendo de este valor. Para $\gamma < 2$ se considera que el sistema se encuentra en un *régimen anómalo*, donde no pueden existir redes grandes, para valores de $2 < \gamma < 3$ un *régimen de escala libre* y un *régimen aleatorio* para valores de $\gamma > 3$ donde la red es indistinguible de una red aleatoria.

1.3.2. El modelo de Barabási-Albert

Una de las razones por la cual el modelo de *random networks* no es bueno explicando redes de la vida real se debe a la falta de generación de grandes *hubs* o nodos con valor alto de k. En particular, esto sucede debido a que el modelo no tiene en cuenta dos factores muy importantes para una red a la hora de desarrollarse. Estos son el apego preferencial o *preferential attachment* y el crecimiento o *growth* [15].

El *preferential attachment* implica que un nuevo nodo que aparece en la red tiene mucha más probabilidad de interactuar con un nodo que tiene múltiples aristas, es decir cuando su grado k es mayor, que con uno que no tiene tantas. Esto es bastante aparente, por ejemplo, al hablar de Internet, donde todos conocemos Google o Facebook e interactuamos con ellos, pero no con muchas de las otras millones de páginas web que existen. Por lo tanto, un mayor número de aristas en un nodo, genera una tendencia a ir paulatinamente obteniendo una cantidad de aristas aún mayor.

Por otro lado, tenemos el *growth*, el cual implica que el número total de nodos de la red N, va aumentando a medida que la red se desarrolla. Esto genera una diferencia muy importante a la hora de modelar la red, debido a que anteriormente al utilizar el modelo de *random networks*, este crecimiento gradual no era tenido en cuenta, sino que lo único que cambiaba con cada paso era el número de interconexiones entre los nodos del sistema.

Utilizando este modelo, es posible generar redes libres de escala de la siguiente manera: se comienza con un número inicial de nodos m_0, donde las aristas entre ellos son elegidas de forma arbitraria, pero todos los nodos deben tener por lo menos una arista. En cada paso se agrega un nuevo nodo con m aristas que conectan este nuevo nodo con los ya existentes en la red. Teniendo en cuenta la existencia de *preferential*

attachment, la probabilidad de conectarse con el nodo i depende de su grado k_i de manera

$$\prod(k_i) = \frac{k_i^\alpha}{\sum_j k_j}. \tag{1.11}$$

con $\alpha = 1$,

Debido a la existencia de *preferential attachment*, se observa un *rich-gets-richer phenomenon*, es decir, que los nodos que tienen una mayor cantidad de nodos conectados tienden a obtener una mayor cantidad de conexiones nuevas, por lo que se vuelven rápidamente nodos de gran centralidad en la red, dejando marginados a los nodos mas pequeños. Estos nodos, como se ha descrito anteriormente, son llamados *hubs* y su existencia en el modelo de Barabási-Albert se debe a la incorporación de *preferential attachment* y *growth*.

La evolución del grado de un nodo se puede modelar a partir de

$$k_i = m(\frac{t}{t_i})^\beta \tag{1.12}$$

siendo m el parámetro que determina la cantidad de aristas con las que inicia cada nodo nuevo añadido al sistema, t_i el momento en que ingresa a la red el nodo i y un parámetro β llamado exponente dinámico, el cual tiene el valor de $\beta = \frac{1}{2}$.

En este caso, la expresión analítica de la distribución de grado de los nodos, es decir, la probabilidad de que al tomar un nodo al azar de la red este tenga grado k, está dada por

$$p(k) \sim 2m^{\frac{1}{\beta}} k^{-\gamma} \tag{1.13}$$

donde el valor del parámetro gamma es $\gamma = \frac{1}{\beta} + 1 = 3$. Es decir, que la distribución de grado sigue una ley de potencias con exponente negativo y $\gamma = 3$.

En el caso del modelo de Barabási-Albert la ecuación propuesta para la probabilidad de generar una nueva arista al incorporarse un nuevo nodo al sistema está dada por la ecuación (1.11). Sin embargo, esta propuesta puede modificarse para dar lugar a lo que se conoce como *non-linear preferential attachment* o apego preferencial no lineal. En este nuevo modelo, la relación con k_i puede ser no lineal, es decir, un valor del exponente α de k_i que sea distinto de 1. Dependiendo de dicho valor nos encontraremos con regímenes de comportamiento distintos los cuáles se detallarán a continuación.

Sublinear preferential attachment $(0 < \alpha < 1)$

En el caso que $\alpha = 0$ no existe un *preferential attachment*, por lo que la descripción coincidiría con el de una red que crece pero sin esta característica. Todos los nodos tendrán la misma posibilidad de conectarse a cualquier nodo independientemente de su grado. En el caso $0 < \alpha < 1$ nos encontramos en un caso donde los nuevos nodos tienen una tendencia a favorecer nodos que se encuentran más conectados en comparación con los menos conectados. Sin embargo, en este caso esta preferencia es débil, y no es lo suficientemente fuerte para generar una distribución

de escala libre cómo fue descrita anteriormente. En este régimen la distribución de grados sigue una distribución de exponencial estirada dada por

$$p_k \sim k^{-\alpha} e^{\left(\frac{-2\mu(\alpha)}{\langle k \rangle (1-\alpha)} k^{1-\alpha}\right)} \tag{1.14}$$

donde $\mu(\alpha)$ depende débilmente de α. Este tipo de *preferential attachment* limita el tamaño y el número de *hubs* existentes en la red.

Superlineal preferential attachment ($\alpha > 1$)

En este régimen la tendencia a relacionarse con nodos altamente conectados es muy alta. En particular, para $\alpha > 2$, se observa un *winner-takes-all phenomenon* donde casi todos los nodos se encuentran conectados a unos pocos nodos llamados *super hubs*. La situación para $1 < \alpha < 2$ es menos extrema pero muy similar. En este caso no encontramos una descripción analítica para la probabilidad p_k. Sin embargo, en este régimen los primeros nodos se convierten en estos *super hubs* y todos los nodos que aparecen posteriormente se conectan a ellos.

Linear preferential attachment ($\alpha = 1$)

Este régimen coincide con el modelo de Barabási-Albert, por lo que la distribución p_k, como fue descrito anteriormente, sigue la ley de potencias de (1.9).

1.3.3. Trabajo relacionado

Para la realización de este trabajo, dado el auge en el interés por este tipo de tecnología [16] nos encontramos con numerosa bibliografía que resultó valiosa a la hora de validar nuestro proceso de decisión. En los últimos años se han desarrollado múltiples enfoques dentro del mismo paradigma de computación descentralizada los cuáles nos permitieron atacar el problema en cuestión desde ángulos distintos y de esa manera facilitar la obtención de conclusiones.

Se han realizado trabajos que estudian la escabilidad del sistema [17], donde en el caso particular de Algorand es uno de sus puntos más fuertes. Por otro lado, un tema de gran importancia a tener en cuenta al analizar este tipo de sistemas es el de la seguridad y su resistencia a ataques, estudiados en Saha et. al [18], Gemeliarana y Fitri Sari [19] y Singh et. al [20]. Este punto es uno de los más estudiados en este tipo de sistemas.

Debido a que el surgimiento de la red de Algorand no se ha dado hasta los últimos tres años la bibliografía e investigación existente para esta red en particular es acotada. Sin embargo, se han realizado múltiples estudios de carácter relacionado al trabajo que se pretende mostrar en este tesis. En particular se podría recalcar el análisis del *preferential attachment* sobre activos en la red de Ethereum [21] o un análisis realizado de la red de Ethereum a partir de sus transacciones [22] centrado en la predicción de conexiones entre los nodos participantes de la red, el cual será un tema central durante este trabajo. Otro ejemplo también en la red de IOTA [23]

y en la *Lightning Network* de Bitcoin [24] donde se ha intentado también analizar la dinámica de las conexiones entre los nodos participantes. En múltiples de estos trabajos fue utilizado el modelo de Barabási-Albert como un posible contendiente a la hora de describir la evolución de una red *blockchain* el cuál ha sido explicado anteriormente en esta tesis y tendrá gran centralidad al analizar los resultados obtenidos en este trabajo.

Capítulo 2

Metodología y tratamiento de datos

En este capítulo se describirá la metodología empleada para la obtención de los conjuntos de datos utilizados a lo largo de esta tesis. También se describirán las decisiones tomadas a la hora de filtrar y organizar dichos datos y durante el desarrollo de los múltiples *scripts* creados a lo largo del trabajo. Esta sección buscará esclarecer la organización general del proyecto para que pueda ser replicado por un tercero en el futuro.

2.1. Obtención de datos

El primer problema que se tuvo al intentar obtener los datos deseados fue la fuente. En un principio se esperaba poder realizar la descarga de la *blockchain* completa, involucrando más de 20 000 000 de bloques. Sin embargo, debido a que el espacio de almacenamiento necesario de aproximadamente 1,2 TB superaba el espacio total de almacenamiento de la computadora utilizada para esta tesis, dicha propuesta inicial fue descartada.

Como segunda posibilidad se propuso también el montado de un nodo, por el cual pasan todas las transacciones de la red. Pero por lo investigado unos días después de proponer este método para la obtención de los datos, se determinó ineficiente debido a que era necesario un período de aceptación del nodo en la red de alrededor de un mes, lo que provocaría un atraso en el comienzo del trabajo en esta tesis.

La solución encontrada fue la de utilizar una API de un proveedor tercerizado. Utilizando este proveedor llamado PureStake y una llave API provista por el mismo, se pudo realizar la descarga de los datos a través de un cliente de `Algod` (la API de Algorand) con una limitación de 30 000 bloques por día. La particularidad que poseía este método de obtención de datos es que era sumamente lento. Esto se debía a que dentro de las posibilidades de descarga, la única opción era descargar un bloque a la vez por cada llamado de la API. Esto presentó algunos problemas a la hora de realizar las descargas de los conjuntos de datos más grandes ya que tardaba varios días en descargar la información si la cantidad de bloques deseada era elevada,

llevando a problemas en casos de cortes internet, necesidad de traslado del equipo utilizado, etc.

Habiendo solucionado la fuente de los datos, se desarrolló un *script* en `Python` para realizar la descarga de los mismos, en el archivo `getBlocks.py` se generó la función `GetBlockInfo` tomando como parámetros el cliente de `Algod` generado a partir de la llave API entregada por el proveedor, el número inicial a partir del cual pretendíamos descargar los bloques y por último la cantidad de bloques deseada. Los datos obtenidos para un bloque con sus respectivos parámetros descritos en la sección 1.2.5 se pueden observar a continuación en la figura 2.1

Como segundo paso, se generó un segundo *script* llamado `acquireTxns.py` con su función `acquireTxns` la cual se encargó de tomar todos los datos dentro de cada uno de los bloques y guardar la fracción deseada. En particular, las transacciones dentro de cada bloque, el número de bloque, el cual provenía del parámetro `round` que se encontraba dentro de la *metadata* del bloque, y por último el *timestamp*, también proveniente del parámetro `ts`, mediante el cual pudimos obtener una fecha para cada transacción. Para ello se generó una lista de transacciones donde en cada elemento se guardó un diccionario con todos los parámetros de dicha transacción y se le agregó dos nuevos pares clave valor para los parámetros de fecha y número de bloque.

Por último, en este proceso de obtención de datos se eligió como manera de organizarlos un `dataframe` utilizando la biblioteca `Pandas` de `Python`. Se ordenaron los datos en ocho columnas en total. La fracción seleccionada de datos que se encontraba en las transacciones fue:

- `Sender Address`: el *address* del usuario que envió la transacción.
- `Receiver Address`: el *address* del usuario que recibió la transacción.
- `Transaction Type`: el tipo de transacción realizada entre el *sender* y el *receiver*.
- `Transaction Block`: el número de bloque en que se realizó la transacción.
- `Transaction Amount`: En el caso en que se enviaran criptomonedas o *tokens*, era discriminada la cantidad en este parámetro.
- `Asset Id`: En el caso de que se enviara algún ASA, se discriminaba su *Id* en este parámetro.
- `Transaction Date`: La fecha en la que fue realizada la transacción. Debido a que nuestra investigación abarcaba un período de años, se decidió solo quedarnos con las fechas de los bloques y no de las horas, las cuáles se encontraban incluidas en el *timestamp* de cada bloque.
- `Transaction Note`: En el caso de que la transacción incluyera alguna nota, esta era discriminada en esta columna.

```
{
"block": {
    "earn": 218288,
    "fees": "Y76M3MSY6DKBRHBL7C3NNDXGS5IIMQVQVUAB6MP4XEMMGVF2QWNPL226CA",
    "frac": 6845750026,
    "gen": "mainnet-v1.0",
    "gh": "wGHE2Pwdvd7S12BL5FaOP20EGYesN73ktiC1qzkkit8=",
    "prev": "blk-ANZ5DIZWY235VI6BDGUMTEQIKA37OIFYBKZBXNXKMNFD4MXFWPSQ",
    "proto": "https://github.com/algorandfoundation/specs/tree/433d8e
    9a7274b6fca703d91213e05c7e6a589e69",
    "rnd": 23621938,
    "rwcalr": 24000000,
    "rwd": "737777777777777777777777777777777777777777777777UFEJ2CI",
    "seed": "XDjkql0xKTdnHDGaXbEzCD/wJPeJz0gmD4Y/nWIg79A=",
    "spt": {
        "0": {
            "n": 23621888}},
    "tc": 881160413,
    "ts": 1663881720,
    "txn": "NnSa5P622FGwZQ96bpNmPpkZX/WIbHcwHnZXBbkXXag=",
    "txn256": "2bral/DgZ/srghKjzj60i6sX3XVx6JrzJ2bYZd2hpoc=",
    "txns": [
        {
            "hgi": true,
            "sig": "7851Yq21StpaeDQQ7NWuaWCRACGebLhALkm9so1EoPiCfXE
                    4ZrYNEVMuUs7cE9Q53b8/FWkJriYzRUTSJjOdBQ==",
            "txn": {
                "aamt": 1,
                "arcv": "C7RYOGEWDT7HZM3HKPSMU7QGWTRWR3EPOQTJ2OHXGY
                         LARD3X62DNWELS34",
                "fee": 1000,
                "fv": 23621935,
                "lv": 23622935,
                "note": "eyJpIjoiMzU1MTg4YTg4ZGM0ZTNiNjM3Mj",
                "snd": "C7RYOGEWDT7HZM3HKPSMU7QGWTRWR3EPOQTJ2OHXGYL
                        ARD3X62DNWELS34",
                "type": "axfer",
                "xaid": 127745593}}
```


Figura 2.1: Datos obtenidos al observar el contenido de uno de los bloques utilizados durante esta investigación.

El *script* desarrollado para poder organizar esta información de manera programática fue `txnsDataframe.py`, utilizando la función *make_df*. Esta función toma la lista de transacciones generada a partir de *acquireTxns*, donde cada elemento de la lista es un diccionario y los clasifica a partir de varias condiciones. En particular, lo considerado más importante a la hora de clasificar las transacciones fue poder definir qué tipo de transacción se estaba realizando. Este *script* requirió múltiples iteraciones debido a que a medida que se analizaron distintos bloques surgieron errores a la hora de desglosar la información que no se encontraban en la página de Algorand Developer. Sin embargo, se logró obtener resultados favorables de forma consistente y se generaron *dataframes* bien organizados para todas las transacciones descargadas.

Por último, se utilizó la biblioteca `NetworkX` donde, a partir de los *dataframes* formados y tomando las columnas de *Sender Addresses* y *Receiver Addresses*, se generaron grafos para cada uno de los *data sets* descargados. Luego de exportarlos, para su posterior visualización se utilizó un programa llamado `Gephi`. Se decidió utilizar este *software* ya que está centrado en la fácil visualización de grafos, y además posee varias herramientas útiles a la hora de responder preguntas acerca de los datos con los que contábamos. En particular, en la figura 2.2 se puede observar uno de los grafos generados elegido aleatoriamente con carácter demostrativo.

El procedimiento completo se organizó en el archivo `generateFullData.py` donde se generó el cliente para utilizar la API y luego entregando el número inicial de bloque, el número final y la cantidad de bloques que se deseaba descargar generó el archivo de *data* de bloques, transacciones, *dataframe* y `Gephi` de dicho *data set*.

2.2. Conjuntos de datos

A lo largo de esta tesis se utilizaron cinco conjuntos de datos en total. Cada uno tuvo un propósito particular que será discutido en esta sección de la tesis. Cada conjunto de datos consistió en múltiples conjuntos de bloques consecutivos los cuáles de ahora en más llamaremos *chunks* de bloques. Cada *chunk* fue utilizado como una forma de notar las características del sistema en un momento particular. Por lo que si buscábamos generar un *data set* que mostrara la evolución del sistema a lo largo del tiempo, era necesario múltiples de dichos *chunks* por *data set*.

Debido al bajo tiempo de generación de bloque que posee esta *blockchain*, cada bloque tarda aproximadamente de 4 a 5 segundos en generarse llevando a una blockchain de aproximadamente 24 000 000 de bloques al momento de comenzar esta tesis. Debido a la lenta descarga de bloques, hubo que tomar una decisión con respecto a cuántos *chunks* debían generarse por *data set* y de cuántos bloques cada uno, para mantener un tiempo razonable de descarga. Una de las primeras discusiones que se generaron a partir de esto fue cuán representativos podían ser estos *chunks*, en general de entre 100 y 20 000 bloques, de lo que realmente estaba pasando en el sistema.

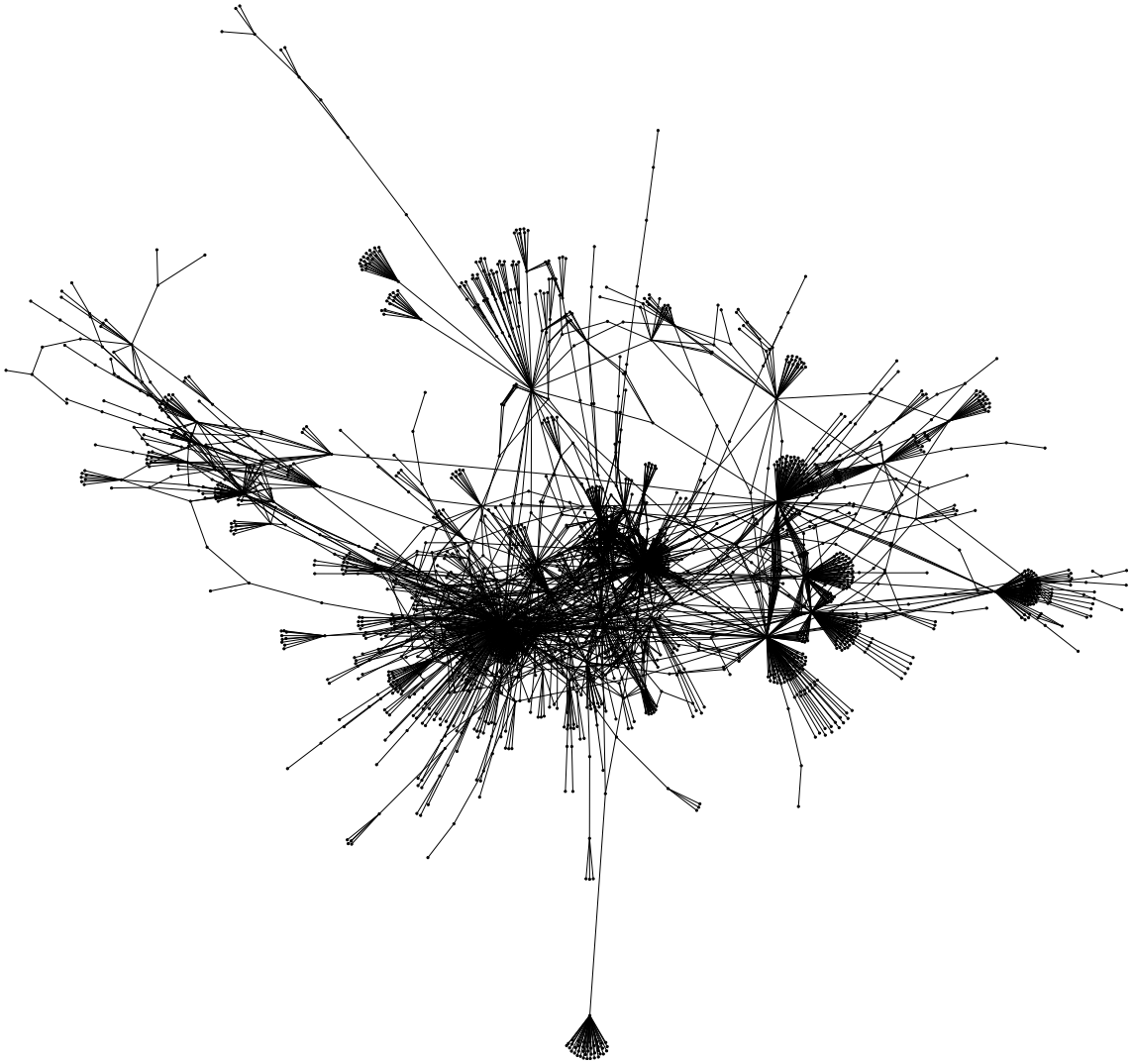


Figura 2.2: Grafo generado a partir de uno de los *Data sets*. Se observa la existencia de nodos con mayor centralidad en la red y la formación de diversas estructuras a partir de las interacciones entre los participantes.

2.2.1. Conjuntos de datos preliminares

De forma preliminar, para realizar pruebas del correcto funcionamiento de los *scripts* descritos anteriormente, se generaron dos *chunks*, uno de 300 bloques y uno de 500 bloques. Se prestó especial atención en el correcto armado del *data frame*, debido a que, como fue descrito anteriormente, fue necesario realizar múltiples iteraciones del *script* debido a inconsistencias en su correcto funcionamiento por la complejidad de los diferentes casos abarcados. A partir de estos dos *chunks* se comenzó a idear qué preguntas se buscaría responder al analizar este sistema. En este momento fue introducida la biblioteca `NetworkX`, la cual fue esencial a la hora de poder exportar nuestros datos a `Gephi` para su posterior visualización y además simplificó en gran medida la obtención de varias medidas estadísticas, en particular durante el proceso de filtrado que se discutirá más adelante.

2.2.2. Nodos spam y aislados

En estos dos sets de datos preliminares se observaron dos fenómenos que fueron centrales a lo largo de esta tesis que serán descritos a continuación. En particular, la forma de su tratamiento fue una de las discusiones centrales que se realizaron a lo largo de estos meses de investigación por lo que es necesario definir ambos fenómenos en profundidad.

Nodos Aislados: A lo largo de todos los *data sets* estudiados se observó que un gran porcentaje de los nodos existentes en la red no interactuaban con ningún otro. Por el contrario, enviaban transacciones a sí mismos y siempre se encontraban desconectados del componente principal de la red. La existencia de estos nodos nos pareció completamente natural e inevitable dentro de un sistema de este tipo. Podría darse por múltiples razones, entre ellas personas probando realizar transacciones por primera vez, aplicaciones realizando tareas que involucraban interacción con si mismas, etc. Sin embargo, debido a su aislamiento, no fue un tipo de nodo que nos aportara información deseada para nuestro análisis. Por lo tanto, se decidió que este tipo de nodo fuera eliminado durante el proceso de filtrado que se describirá más adelante.

Nodos Spam: La presentación usual de este tipo de nodos fue de un nodo central rodeado por múltiples otros, en general desconectados de cualquier otro componente del sistema excepto del nodo central. Dicho nodo enviaba transacciones de forma unilateral a sus vecinos. En particular, este tipo de nodos generó bastantes problemas a lo largo de todo el proceso de investigación, debido a que, por ejemplo, al ser tan denso y numeroso no permitía el correcto funcionamiento del programa de visualización `Gephi`. Además, todas las estadísticas del sistema se veían muy afectadas por la presencia de este tipo de nodos, pero estos no aportaban información relevante para la descripción de la evolución del sistema. Por lo tanto, se decidió eliminar dichos nodos durante el proceso de filtrado de los datos. A posteriori, creemos que haber tomado esta decisión fue necesaria para poder lograr un análisis correcto del sistema.

En la figura 2.3 se puede observar claramente la aparición de ambos de estos

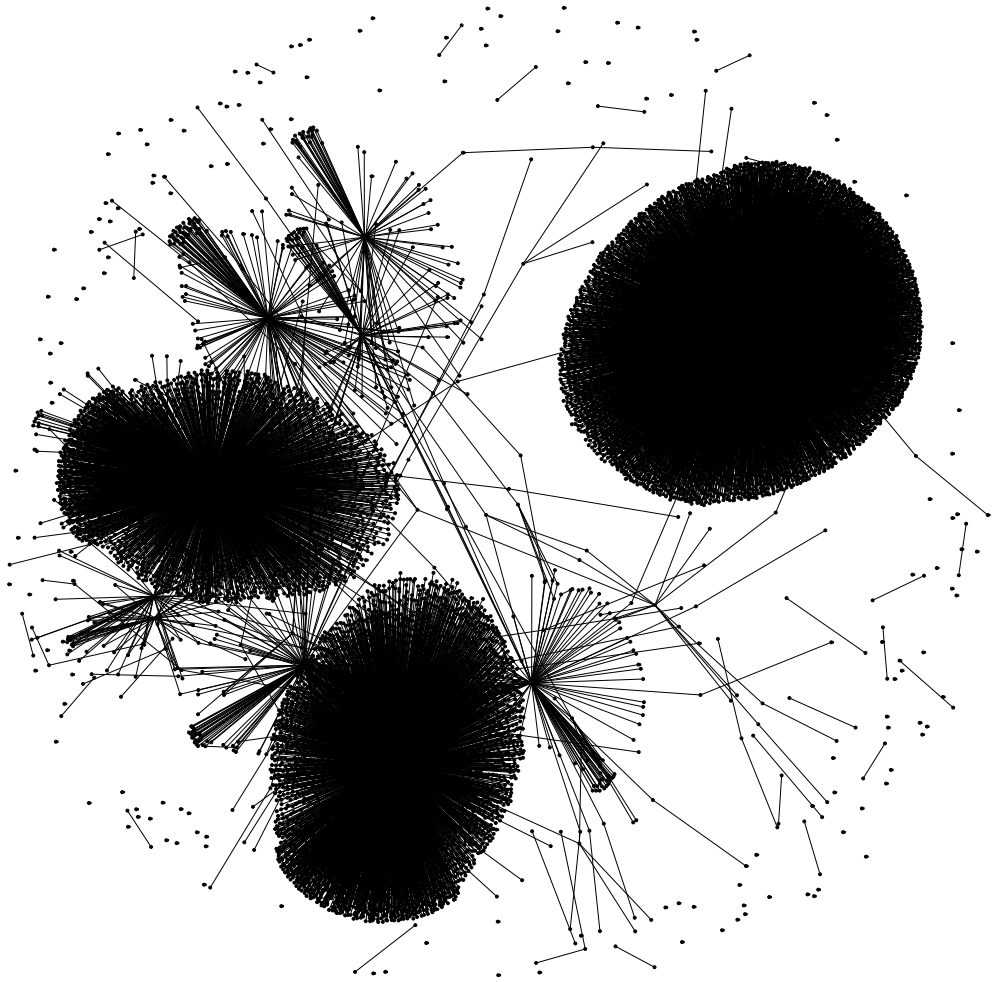


Figura 2.3: Tres nodos *spam* y múltiples nodos solitarios descubiertos durante el análisis de los dos *Data sets* preliminares.

fenómenos. Estos tipos de nodos fueron eliminados durante el proceso de filtrado para lograr obtener una descripción más acertada del estado actual del sistema en cada *chunk*.

2.2.3. Data sets exploratorios

Luego de la obtención de los sets de datos preliminares y la posterior verificación del correcto funcionamiento de los programas desarrollados hasta ahora, se pasó a un segundo set de datos que involucraba 16 *chunks* de 500 bloques cada 1 millón de bloques de la red, comenzando desde el número de bloque 9 000 000 y terminando en el 24 000 000. Este fue llamado *Data Set 1*.

Una de las primeras conclusiones a la que llegamos a partir de lo observado en este nuevo set de datos fue que la información existente en la red previa al bloque 11 000 000 no sería relevante en nuestro análisis dado que durante períodos anteriores a este, la actividad de la red era tan baja que no nos pareció que correspondía

incluirlos dentro del análisis. Este criterio fue elegido de forma arbitraria como punto de inflexión del comienzo efectivo de la actividad de la red.

Conociendo ya la existencia de los nodos aislados y *spam* se decidió realizar un filtrado, en este caso de manera manual. Dicho proceso de filtrado se realizó observando en el *software* `Gephi` cada uno de los *chunks* y determinando qué nodos eran considerados *spam* de forma completamente arbitraria, pero intentando seguir las directivas:

- El nodo poseía cientos de nodos conectados a él.
- Las transacciones eran realizadas de forma unilateral. Es decir que el nodo central solo enviaba transacciones y no recibía o viceversa.
- Dicho nodo no estaba conectado con ningún otro componente de la red, excepto en algunos casos donde se hallaba conectado a traves de una sola arista.

Una por una se filtraron las que llamamos *Blacklisted Accounts* generando un set de datos más limpio y con una estadística menos sesgada por dichos nodos. Además, se filtraron todas las transacciones de nodos que enviaban transacciones a sí mismos, eliminando entonces ambos tipos de nodos problemáticos. En este momento de la investigación teníamos claro que dicho modo de filtrar los datos debería cambiar en el futuro pero recién estábamos buscando desarrollar el código para realizar estadística descriptiva del sistema que fuera programática, por lo que dicho proceso de filtrado sería implementado más adelante. Por ahora nos mantuvimos en una etapa de exploración.

Luego del análisis de dicho *data set* inicial, se decidió incorporar uno nuevo bastante más extenso, donde se tomaron 61 *chunks* de 1000 bloques cada uno cada 200 000 bloques de la red, entre los bloques 11 000 000 y 23 000 000. Este fue llamado *Data Set 2*. Con este nuevo set de datos se buscó obtener una mayor cantidad de puntos en las variables analizadas dentro de la estadística descriptiva que será comentada con detalle a continuación.

2.2.4. Estadística descriptiva

En esta sección se describirá con gran detalle las preguntas detrás de las estadísticas que fueron obtenidas de cada set. En particular, cuáles fueron esenciales a la hora de realizar una descripción general de la red. En un principio se propusieron múltiples preguntas a responder pero a lo largo de la investigación se agregaron muchas que no habían sido tenidas en cuenta en un principio.

El desarrollo programático de la estadística generada fue plasmado en el *script* `describeData.py`. Cada una de las estadísticas generadas fue considerada por *chunk*, es decir, se obtuvo un punto para cada conjunto de bloques dentro del set de datos. En este se incluían:

- *Created Apps*: Número de aplicaciones creadas.
- *Total_transaction_amount*: Número total de transacciones.

- *Total_sender_number*: Número total de emisores.
- *Total_receiver_number*: Número total de receptores.
- *Total_active_accounts*: Número total de cuentas activas.
- *mean_transaction_amount_per_sender*: Número promedio de transacciones que enviaba cada emisor.
- *mean_transaction_amount_per_receiver*: Número promedio de transacciones que recibía cada receptor.
- *mean_amount_of_unique_receiver_for_sender*: Número promedio de receptores únicos por emisor.
- *mean_amount_of_unique_sender_for_receiver*: Número promedio de emisores únicos por receptor.
- *only_sender_accounts*: Número de cuentas que solo eran emisoras.
- *only_receiver_accounts*: Número de cuentas que solo eran receptoras.
- *percent_of_senders_only_senders*: Porcentaje de los emisores que eran solo emisores.
- *percent_of_receivers_only_receivers*: Porcentaje de los receptores que eran solo receptores.
- *percent_of_accounts_only_senders*: Porcentaje de todas las cuentas que se encontraban incluidas en el *chunk* que eran solo emisoras.
- *percent_of_accounts_only_receivers*: Porcentaje de todas las cuentas que se encontraban incluidas en el *chunk* que eran solo receptoras.
- *sender_average_transacted_with_same_accounts*: Número de veces que, en promedio, un emisor transaccionaba con una misma cuenta.
- *receiver_average_transacted_with_same_accounts*: Número de veces que, en promedio, un receptor transaccionaba con una misma cuenta.
- *involved_accounts_per_type*: Número de cuentas únicas involucradas en transacciones de cada tipo.
- *involved_senders_per_type*: Número de emisores involucrados en transacciones de cada tipo.
- *involved_receivers_per_type*: Número de receptores involucrados en transacciones de cada tipo.
- *percentage_of_total_accounts_per_type*: Porcentaje de cuentas respecto del total involucradas en cada tipo.

- *transaction_amount_in_microalgo*: Cantidad de criptomoneda Algo enviadas en una transacción en la unidad µAlgo, lo cual representa 10^{-6} Algo.
- *closing_transactions_count*: Número de transacciones de cierre.
- *more_than_one_algo*: Número de transacciones que enviaron una cantidad superior a 1 Algo. Se tomó como límite esta cantidad para distinguir una transacción real de una *spam*. Dicha cantidad fue determinada de forma puramente arbitraria ya que se observó un gran número de transacciones con valores menores a los 0,01 Algo por transacción.
- *more_than_one_algo_percentage*: Dentro del número total de transacciones para ese *chunk*, el porcentaje de transacciones mayores a 1 Algo.
- *mean_amount_of_algo_sent*: Porcentaje promedio de Algo enviado en cada transacción de tipo *pay*.
- *transaction_type_percentages_of_total_transactions*: Para cada tipo de transacción posible, obtuvimos el porcentaje total de ese tipo de transacción con respecto al total de las mismas. Este fue uno de los principales actores dentro de lo que fue el desarrollo de esta tesis. En particular los resultados obtenidos a partir del estudio de la evolución de estos porcentajes constituyeron una gran parte del trabajo realizado durante esta investigación.
- *total_activity*: Dentro de todas las cuentas activas en todos los *chunks* involucrados en el set de datos, en cuántos de estos *chunks* apareció cada cuenta. Se buscó determinar si la actividad de cada uno de los nodos se mantenía a lo largo del tiempo. Un punto a tener el cuenta en este caso es que, debido al muestreo utilizado, existía la posibilidad de que cada nodo que hubiera estado activo en un *chunk* anterior, no fuera captado en los siguientes y viceversa.
- *max_degree_list*: Grado del nodo más grande de mi red.
- *number_of_nodes*: Número total de nodos dentro de mi red.
- *algo_price_correlation*: Se desarrolló un *script* llamado `algopricecorrelation.py` donde se buscó la correlación entre el precio de mercado en USD de la criptomoneda Algo y el número total de Algo enviados en promedio por *chunk*.

2.2.5. Proceso de filtrado

Para los sets de datos siguientes, sabiendo ya qué cosas buscábamos saber sobre los datos, nos decidimos a desarrollar ahora sí un sistema programático de filtrado de los datos para no tener que seguir realizando el proceso de forma manual en futuros *data sets* y además para estandarizar el proceso.

La mayor dificultad al intentar generar un filtrado efectivo fue poder elegir de forma correcta el criterio a partir del cual un nodo debía ser considerado filtrable.

En el caso de los nodos solitarios, no presentaron ningún problema a la hora de ser filtrados, pero no puede decirse lo mismo en el caso de los nodos *spam*.

Como un primer acercamiento al problema se desarrolló el *script*

- filterTxns.py

donde se intentó realizar el filtrado de los nodos no deseados a partir de sus transacciones. Es decir que se tomó el promedio de las transacciones realizadas por cada nodo y a partir del número promedio total se determinó un límite de filtrado a partir de

$$limite = threshold * mean\_txns \tag{2.1}$$

siendo el *threshold* un número arbitrario tomado a partir de los resultados empíricos obtenidos de los datos, en este caso se tomó $threshold = 700$. Lo que se buscaba hacer con este proceso de filtrado fue, dado que un nodo *spam* contaba con múltiples transacciones, debía entonces superar el promedio de transacciones de la red múltiples veces. Sin embargo, se encontraron dos problemas que este tipo de enfoque no lograba resolver.

El primero era que múltiples componentes pequeños, de entre uno y diez nodos, quedaban sin filtrarse. Idealmente buscaríamos eliminarlos ya que no nos pareció que aportaran a la estadística general de la red. El segundo fue que en múltiples ocasiones los nodos *spam* no eran eliminados. Por el contrario, debido a que lo único que tenía en cuenta este algoritmo de filtrado era la cantidad de transacciones varias veces obtuvimos que el resultado final posterior al filtrado fue la supervivencia de nodos *spam* y la eliminación de los nodos principales del componente principal de la red. Esto se debía que algunos nodos importantes realizaban muchísimas transacciones, pero no lo hacían de manera unilateral como los nodos *spam*. En conclusión, este método de filtrado se consideró defectuoso y se pasó a un segundo intento que se explicará a continuación.

El segundo enfoque dado a este problema fue muy parecido pero en vez de tomar como parámetro de filtrado la cantidad de transacciones promedio realizadas por un nodo en la red, se tomó el grado promedio de los nodos de la red y se mantuvo el sistema de *threshold*, tomando en este caso $threshold = 300$. Es decir que, cualquier nodo que superara el grado de $k = avg\_degree * threshold$ sería filtrado. El valor dado para el *threshold* fue obtenido de los datos empíricos analizados. En particular, se observó de forma consistente que los nodos *spam* solían superar estos valores, y que los nodos pertenecientes al componente principal de la red no estaban cerca de dichos valores, solucionando de esta manera el problema que se había encontrado en el enfoque anterior. Es decir, utilizar el grado promedio como parámetro, sí era algo característico que describía el comportamiento de los nodos. En general, como un ejemplo, los nodos principales de la red podían llegar a tener un grado de k = 100 mientras que un nodo *spam* podía encontrarse en el grado de k = 2000, habiendo una diferencia muy significativa entre ambos tipos de nodo. Este nuevo enfoque fue plasmado en los *scripts* `filterDegree.py` y `filterDegreeImproved.py` donde se realizaron dos iteraciones sucesivas del mismo proceso de filtrado. Sin embargo, en el segundo también se tuvo en cuenta que, si un nodo era filtrado en uno de los

chunks, dicho nodo también fuera filtrado en los demás, cosa que antes no se estaba teniendo en cuenta, sino que para cada *chunk* se realizaba un filtrado aislado del realizado para el *chunk* anterior. Con este nuevo enfoque, se obtuvieron resultados mucho mejores que con el anterior, por lo que se decidió que era el adecuado para este tipo de análisis. Sin embargo, se encontraban todavía algunos detalles a mejorar. En algunas iteraciones se obtuvieron nodos *spam* pero de menor escala, los cuáles al no superar el *threshold* no eran filtrados por el algoritmo. Y, por otro lado, al igual que en el enfoque anterior, la existencia de grupos de nodos pequeños aislados de la red principal seguía manteniéndose como una molestia que buscaríamos resolver en la última iteración de este problema.

Como tercer y última iteración de este problema, se decidió un cambio drástico en el proceso de filtrado, el cual fue el de eliminar todos los nodos que no se encontraran conectados al componente principal de la red. De esta manera, solucionamos los dos problemas presentados anteriormente de manera eficiente. El *script* de este proceso de filtrado se plasmó en `filterFinal.py` donde se incorporó un nuevo filtrado a partir del componente luego del proceso de filtrado realizado hasta ahora a partir del grado promedio de la red y el sistema de *threshold*. Es decir que, el algoritmo verificaba si el nodo se encontraba dentro del componente principal de la red, y en caso de no encontrarse ahí, eliminaba todas las transacciones correspondientes a dicho nodo. Los resultados obtenidos con este nuevo algoritmo fueron fantásticos, resolviendo todos nuestros problemas y no borrando nodos deseados o viceversa. Algo a tener en cuenta, donde se tuvo especial cuidado, fue en la posibilidad de estar borrando posibles componentes chicos que luego se incorporaran a la red principal. Al realizar este tipo de filtrado, dicho proceso no podría observarse de forma paulatina, sino que aparecerían de golpe, debido a la eliminación de dichos nodos. Sin embargo, esto es algo que se tuvo en cuenta y nos pareció que las ventajas de la utilización de este método superaban las desventajas y por otro lado, también se observaron detenidamente los datos experimentales utilizados y no se encontraron casos donde esto sucediera. Es decir, que todos los nodos que se incorporaban a la red principal lo hacían de forma directa. En la figura 2.4 se puede observar los resultados obtenidos para un *chunk* aleatorio utilizado. El único sobreviviente fue el componente principal de la red y se logró obtener de forma consistente resultados como los observados en la figura. De esta manera fue posible observar con mayor claridad el proceso de desarrollo de la red como un ente interconectado sin tener nodos que no aportaran a la estadística del sistema que a nosotros nos interesaba.

2.2.6. Data sets finales

Como se describirá con mayor detalle en la sección de resultados, a partir de los *data sets* de exploración y aplicando la estadística descriptiva previamente señalada en esta tesis se obtuvieron resultados que posibilitaron la observación de tres etapas de desarrollo en la evolución de la red. Por lo tanto, se decidió generar otro *data set* llamado *Data set 3*, sobre el cual posteriormente fue aplicado el algoritmo de filtrado generando lo que fue llamado *Data set 4*. En este nuevo *data set* buscábamos tener

Figura 2.4: A la izquierda, los datos de un *chunk* previos al proceso de filtrado. A la derecha el resultado obtenido posterior al proceso del filtrado.

una mayor definición dentro de cada uno de estos períodos por lo que se generaron 241 *chunks* de 500 bloques cada uno cada 50 000 bloques guardados en la red. Estos *chunks* fueron divididos de la siguiente manera:

- Período 1: Desde el bloque 11 000 000 hasta el 14 600 000 con un total de 73 *chunks*.
- Período 2: Desde el bloque 14 800 000 hasta el 17 400 000 con un total de 53 *chunks*.
- Período 3: Desde el bloque 17 600 000 hasta el 23 000 000 con un total de 109 *chunks*.

Se decidió, por último, generar los que fueron llamados **super sets**, de 20 000 bloques consecutivos cada uno, uno para cada período observado. Nos aseguramos de que se encontraran centrados en lo que a nuestro criterio era la zona que mejor describía el comportamiento general de la red en dicho período. Estas posiciones fueron 14 000 000 para el primer período, 17 200 000 para el segundo y 20 000 000.

Lo que se buscó con la generación de estos últimos *super sets* fue asegurarnos de que la estadística obtenida para los diferentes *chunks* fuera consistente con lo observado en los *data sets* anteriores durante el mismo período de la red, esperando obviamente fluctuaciones en el comportamiento debido a la diferencia de tamaño entre ambos sets de datos. Es decir, nos queríamos asegurar de que los resultados fueran invariantes de escala, en particular a la hora de intentar ajustar la distribución del grado de los nodos de la red con los modelos propuestos en la introducción.

La exploración de los datos, posterior a la obtención y filtrado de los mismos, se organizó en diferentes *jupyter notebooks* a lo largo de todo el proceso de investigación. Estos se encuentran organizados en las carpetas *Exploration notebooks*, *Data sets 4 y 5* y *Data sets preliminares*.

2.2.7. Price data

Por otro lado, se buscó realizar comparaciones cruzadas utilizando el precio de la criptomoneda ALGO y otros parámetros relevantes del sistema. Para ello, fue necesario obtener la información de precios histórica, que se encontraba organizada en siete columnas.

- `Date`: Fecha del dato.
- `Open`: Precio de apertura del día.
- `High`: Precio máximo alcanzado ese día.
- `Low`: Precio mínimo alcanzado ese día.
- `Close`: Precio de cierre del día.
- `Adj`: Precio post cierre de mercado. Concuerda con el precio de `Close`.
- `Volume`: Volumen de la criptomoneda transaccionado en el día.

En particular, durante nuestro análisis, utilizamos la fecha para cada uno de los días y para obtener el precio tomamos el promedio entre los valores de las columnas `Open` y `Close`.

2.2.8. Suavizado

Debido a la cantidad de puntos utilizados y la alta volatilidad en los parámetros analizados, en algunos casos, se decidió realizar un suavizado sobre los diferentes sets de datos para poder, posteriormente, analizar las tendencias. Debido a un alto ruido en las variables, a veces se tornaba difícil poder realizar una inspección visual a fin de verificar la existencia de patrones de variación.

En un principio, se tuvieron en cuenta múltiples métodos de *smoothing* o suavizado, pero nos decidimos por utilizar un filtro de Savitzky-Golay debido a que, utilizando este método, fue sencillo realizar una aproximación polinómica sobre nuestros datos. En particular, se utilizó la biblioteca `Scipy` donde el módulo `Signal` tenía dentro la función `savgol_filter` la cual toma los datos, un ancho de ventana a promediar y el orden del polinomio deseado. Se decidió utilizar polinomios de grado siete ya que fue certero a la hora de ajustar los datos obtenidos, se probó también comparando con grados más altos y bajos, pero los resultados obtenidos con ellos eran peores que con el valor previamente determinado. Por otro lado, debido al nivel de volatilidad en las mediciones, se eligió un valor de ancho de ventana relativamente alto de alrededor de 5 a 20 puntos promediados en cada punto. En algunos casos, esto representó aproximadamente entre un 10 % y un 25 % de los puntos obtenidos.

El proceso de filtrado se implementó en el *script* `savgol_smoothing.py` donde se tomaron las listas obtenidas a partir de las funciones descritas anteriormente que habían generado las diferentes variables a estudiar y se las pasó por el filtro.

2.2.9. Ajustes de datos

Durante el proceso de estudio de la evolución del grado de los nodos se realizaron ajustes sobre los datos probando diferentes modelos para intentar describir de manera correcta dicha evolución. Para ello se utilizaron dos bibliotecas de `Python`:

- `Scipy`: un método de ajuste llamado `curve_fit` el cual ajustaba de forma numérica modelos no lineales como los utilizados en este trabajo.
- `Scikit-Learn`: se utilizó el método para el cálculo del R^2 que se incorporó para cuantificar la bondad del ajuste realizado.

Capítulo 3

Resultados

En este capítulo, detallaremos los resultados obtenidos para cada conjunto de datos tras llevar a cabo la exploración correspondiente. Nuestro objetivo principal consistió en crear una descripción detallada de la evolución del sistema a lo largo del período temporal analizado en esta tesis. Aspiramos a extraer conclusiones que permitieran identificar distintas etapas en su desarrollo. Adicionalmente, examinaremos las conclusiones derivadas de los datos relativos a las variables estudiadas, y explicaremos cómo estas interpretaciones guiaron la investigación hasta su conclusión. En la fase final, llevamos a cabo el ajuste de los datos obtenidos para la distribución del grado de los nodos mediante los modelos previamente descritos en la introducción.

3.1. Transacciones destacadas

Uno de los primeros resultados interesantes apareció al observar las transacciones que sucedían dentro de la red al realizar las primeras pruebas con los sets de datos preliminares. Se observaron por primera vez los nodos *spam* y en particular un tipo de transacción que se repetía a lo largo de toda la evolución de la red.

Estas transacciones de tipo `axfer`, es decir de transferencia de activos, se presentaban de una forma particular, donde siempre enviaban una cantidad de cero activos y, simplemente, pagaban la comisión de la red por transacción de 0,001 Algo. Sin embargo, al observar el contenido del mensaje dentro de la transacción se comprendió que la verdadera información que se estaba transmitiendo en realidad se encontraba allí. En la figura 3.1 se observa una transacción realizada por uno de los nodos observados de la red. En la transacción se indica que fueron enviados 0 Planets, un *token* creado por la empresa Planet Watch. En el mensaje se observan tres parámetros, los cuales se utilizan para transmitir información y que esta quede guardada dentro de la *blockchain* de manera inmutable.

Luego de indagar con miembros de la comunidad de Algorand y de observar su página web, aprendimos que Planet Watch es una empresa centrada en la medición de la calidad del aire en el mundo. Además, cada una de las transacciones realizadas en realidad eran provenientes de un dispositivo que distribuyen por el mundo y

que actualiza la información actual del aire local a través de una transacción en la *blockchain* de Algorand. Este resultado nos pareció muy interesante y fue uno de los primeros acercamientos a soluciones del mundo real para las cuales la *blockchain* de Algorand ya estaba siendo utilizada.

Sin embargo, éste es tan solo un ejemplo de este tipo de transacciones, las cuales fueron constantes a lo largo del tiempo e involucraron múltiples empresas distintas. A continuación se describirán algunas de las observadas.

- Yieldly: Se observaron transacciones correspondientes a recompensas entregadas por el protocolo de Yieldly, una aplicación perteneciente al grupo de las llamadas `finanzas descentralizadas`, un concepto que no fue discutido en esta tesis, pero que tiene gran relevancia hoy en día al listar los diferentes posibles usos de la tecnología blockchain. Actualmente, Yieldly es llamado YNFT.

- Ajedrez: Se observaron transacciones pertenecientes a una página de ajedrez, la cual parecía estar entregando como *rewards* de registro una fracción de la criptomoneda Algo y además utilizaba las transacciones para guardar el *rating* de los jugadores que se encontraban registrados en la página de forma inmutable en la *blockchain*.

- Phishing: Se observaron también transacciones de carácter *spam*, es decir, que mantenían el comportamiento descrito anteriormente pero en el mensaje no había información relevante que se deseara guardar sino que se encontraban mensajes de tipo *phishing*, buscando llevar al usuario a un URL particular, u ofreciendo algún tipo de descuento en un producto. Un ejemplo que se observó empíricamente fue: *"Take a look at the newest ALGO distribution initiative - airdropalgorand.com"*. Este tipo de transacciones en particular nos pareció sorprendente, ya que desde el comienzo de la red ya se encontraban circulando transacciones cuyo mero propósito era lo que hoy en una casilla de email llegaría a la parte de *spam*. Algo incluso más llamativo fue que momentos después de que el usuario recibiera dicha transacción, recibió otra con el siguiente mensaje adjunto: *"The airdrop site you were just spammed with is a KNOWN phishing site attempting to scam you. Do NOT connect or give your phrase to that site!"*. Es decir, de alguna manera estas transacciones *spam* eran identificadas por Algorand y se había incorporado un sistema de alerta para evitar que los usuarios cayeran en este tipo de estafas. Estas transacciones pueden haber sido el puntapié inicial para la necesidad de implementar la transacción de tipo opt-in descrita en la sección de transacciones en la introducción de esta tesis.

Sin embargo, debido a que la información provista por estas transacciones eran repetitiva y, dejando de lado el contenido de su mensaje, no aportaban al análisis que buscábamos realizar, este tipo de transacciones fueron filtradas de forma manual en los primeros dos *data sets* y luego de forma programática en los siguientes.

```
Sender: ZW3ISEHZUHPO7OZGMKLKIIMKVICOUDRCERI454I3DB2BH52HGLSO67W754
Amount: 0 Planets
Receiver: QZJDHVWGOUDCNECE7NMJKPTCRWKDUGTE4WZOTF7OUNX3WVOLII2EU2KALU
Asset ID: 27165954 (PLANET)
Sender ASA Balance: 27,540,160.499835 Planets
Receiver ASA Balance: 0 Planets
Note: {
        "sensor Id": "PW:KA_006397",
        "streams": 51,
        "data": "CGBiX/5YeGBHEVH0Dd7eGvdczhiJPiOB1RGQmjC8QeZzC0EqPU
        1oJh3R404WjbcuEgM6RrZgVULCya+cA9P7vvP8xud8TQRYX+Cm3uda/uOQzu
        uONWT4lk5wJyntFUinGWpbKE8v6C4NqmmUkobEnkVoRdRJaOBAFI7v"
}
```

Figura 3.1: Transacción destacada realizada en la red de Algorand. En el mensaje adjunto a la transacción se encontaron tres parámetros, *sensorId*, *streams* y *data*, a través de los cuales se guardaba información en la *blockchain*.

3.1.1. Porcentaje de transacciones filtradas

Otro de los resultados observados luego de haber implementado el algoritmo de filtrado fue un cambio drástico en la cantidad de transacciones remanentes en cada uno de los *chunks*. En particular, se observa en la figura 3.2 que hay tres fases claras dentro de la historia de la red. En una primera fase el total de las transacciones remanentes luego del proceso de filtrado rara vez superaba el 5 % del total de transacciones. Esto fue realmente sorprendente ya que implicaba que el 95 % de las transacciones que sucedían en la red se encontraban en alguna de las dos categorías de nodos no deseados descritas anteriormente. Utilizando la herramienta de visualización `Gephi`, se analizaron de forma manual varios de estos *chunks* y se confirmó la presencia de múltiples nodos *spam*, de los cuales provenían la mayor cantidad de las transacciones de la red.

En una breve segunda etapa, de forma explosiva, se generó un cambio considerable hacia un nuevo rango de valores de alrededor de los 25 puntos porcentuales de transacciones remanentes. Este fue el comienzo de la tercera etapa, donde se observó un mayor nivel de actividad, interconexión y crecimiento del componente principal de la red, el cual se desarrolló de manera exponencial durante la segunda etapa. Incluso dentro de esta nueva etapa, el porcentaje de transacciones remanentes luego del filtrado obtenido implicaba que cerca del 75 % de las transacciones realizadas dentro de la red estaban centradas alrededor de la existencia de los nodos *spam* o eran nodos solitarios, los cuales no interactuaban con otros nodos dentro de la red. Este resultado nos confirmó la necesidad de eliminar a los nodos *spam* y a los nodos solitarios, los cuales dada la cantidad porcentual de transacciones que ocupaban del total hubieran hecho que cualquier intento de realizar estadística sobre la red se

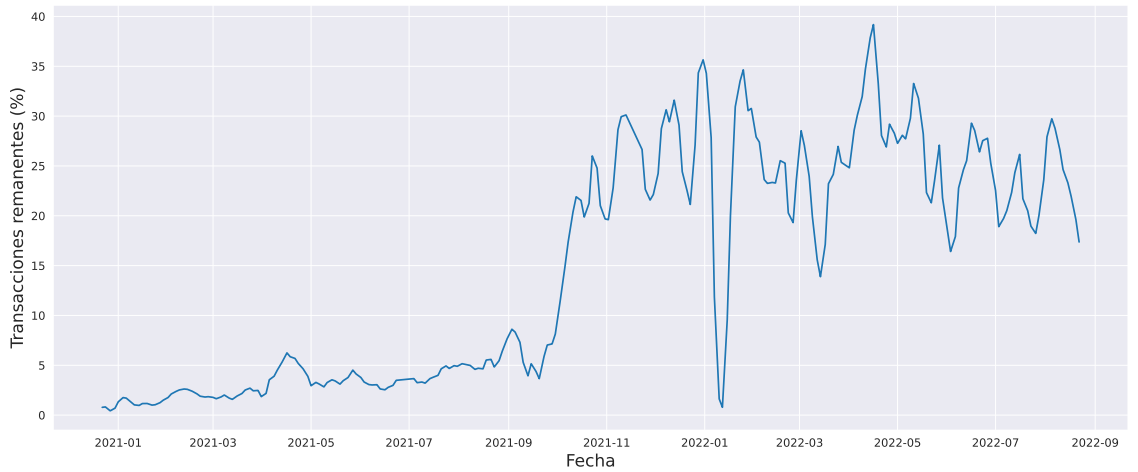


Figura 3.2: Porcentaje remanente de transacciones luego de realizar el proceso de filtrado sobre un *Data set*.

viera sesgada fuertemente por la existencia de dichos participantes.

Una particularidad observada en el gráfico que, inicialmente, nos llamó la atención, fue la existencia de un mínimo durante la tercera etapa observada. Esto podría haber indicado una falla en nuestro proceso de filtrado, mostrando la necesidad de un ajuste del mismo. Sin embargo, luego de investigar su origen, se determinó que, en realidad, una sucesión de 6 puntos se encontraban debajo de los diez puntos porcentuales de transacciones remanentes pero en el gráfico solo uno era observado debido al proceso de suavizado utilizado. La existencia de dichos mínimos fue atribuido al crecimiento exponencial en el número de nodos en la red, los cuales al no ser parte del componente principal fueron filtrados por el algoritmo, dejando así tan solo una pequeña parte de la cantidad de participantes originales.

Por ejemplo, el pico de nodos totales observados fue de 31 115, pero luego de pasar por el proceso de filtrado quedaron tan solo 1147. Una conclusión interesante que pudimos obtener a partir de lo observado fue que la expansión exponencial de la red se produjo con una velocidad muy alta, pero la mayoría de los nodos que se fueron incorporando pertenecían a alguna de las dos categorías de nodos no deseados que se describieron anteriormente. Este cambio tan abrupto observado nos dio los primeros indicios de la existencia de algún cambio en el comportamiento cualitativo del sistema que buscaríamos explicar en el futuro utilizando diferentes variables. En la siguiente sección se describirá detalladamente qué variables fueron utilizadas para generar una descripción de la red a través de un sistema de tres etapas.

3.2. Períodos temporales

Durante el análisis del *Data set 2*, se obtuvieron resultados que definieron la dirección de esta tesis. Dos situaciones encontradas sugirieron la existencia de tres períodos claramente definidos dentro del tiempo de vida de la red. A partir estos,

fue posible compartimentalizar los análisis realizados posteriormente a estos períodos por separado y de esa manera se procedió a mostrar diferencias entre las variables de estos períodos que avalaran nuestra decisión.

Una de las variables a estudiar fue el cambio en la distribución de los tipos de transacciones que se realizaban en la red a lo largo del tiempo. En particular, nos pareció interesante poder determinar si existía una dominancia de algún tipo de transacción sobre las otras. Por ejemplo, si la mayoría de las transacciones fueran de tipo `pay`, eso nos estaría indicando que la red era utilizada principalmente como medio transaccional de criptomonedas o dinero pero no tanto por sus propiedades como base de datos o aprovechando la posibilidad de generar aplicaciones dentro de la red.

Los resultados obtenidos pueden observarse en la figura 3.3. Lo primero que nos llamó la atención y que fue clave en determinar la existencia de un primer período del sistema fue la clara dominancia inicial de las transacciones de tipo `pay`. Es decir, que durante todo este período inicial de la red, éste era el mecanismo utilizado para transferir valor. Notablemente, solo transacciones de tipo `pay` y `axfer` se registraron durante este período. Es decir que, los demás tipos de transacciones aún no habían sido incorporados al protocolo. En particular, no se observaron transacciones de tipo `appl`, las cuáles tomarían protagonismo en los períodos siguientes.

La dominancia de transacciones de tipo `pay` cambió de manera abrupta, dando lugar a una nueva etapa de la red donde se incorporaron dos nuevos tipos de transacciones relevantes, `axfer`, la cual ya había aparecido en el período anterior pero en menor medida, y `appl`, la cual parecía haber sido incorporada al comienzo de esta segunda etapa. Esta nueva etapa se caracterizó por ser de transición donde el sistema pasó de tener la mayoría de sus transacciones centradas en pagos a mostrar una participación mucho mayor de transacciones realizadas por aplicaciones, apoyándose en las características de base de datos de Algorand descritas anteriormente.

Posteriormente, el análisis de los porcentajes de estos nuevos tipos de transacciones y las de tipo `pay` determinó el inicio de un tercer período donde la predominancia de las transacciones de tipo *appl* superó a las demás. Sin embargo, la distribución de porcentajes entre estos tres tipos se mantuvo distribuido de forma más equitativa que en los primeros dos períodos los cuales mostraron una amplia dominancia de las transacciones de tipo *pay*. En esta última etapa no se observó un cambio tan abrupto como en la anterior para los porcentajes de las transacciones, indicando una mayor estabilidad en las variables del sistema.

Las transacciones de tipo *acfg* fueron observadas mayoritariamente a partir de fines del segundo período, llegando a representar en algunos momentos hasta alrededor del 10 % de las transacciones totales. Sin embargo, no tuvo especial centralidad en ninguno de los períodos temporales estudiados. En lo que respecta a las transacciones de tipo *keyreg* y *afrz*, el número de transacciones observadas correspondientes a estos tipos fueron extremadamente bajas y no representaban un número relevante de transacciones en el sistema, por lo que no fueron estudiadas en profundidad durante los análisis siguientes.

Por otro lado, dos indicadores utilizados para reafirmar la existencia de períodos

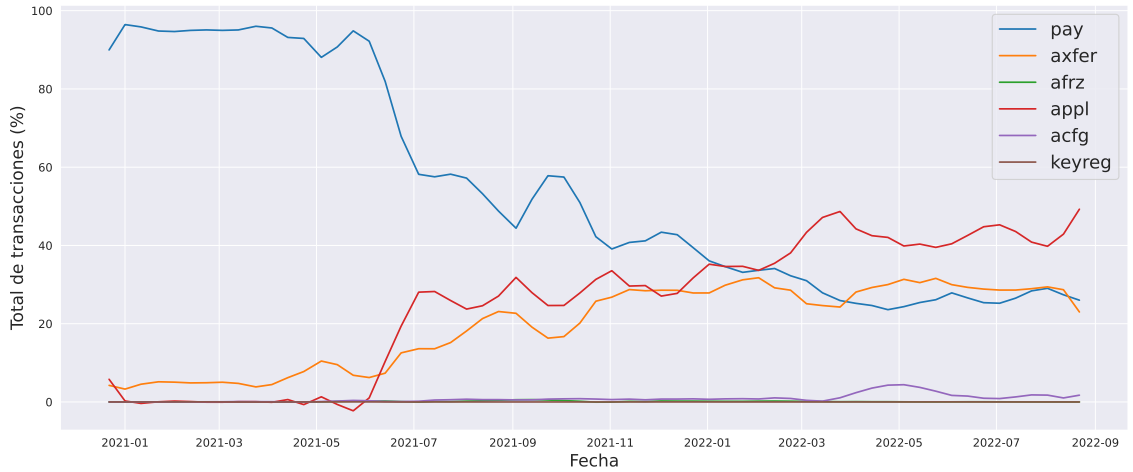


Figura 3.3: Evolución del porcentaje de transacciones por tipo de transacción utilizados en la red a lo largo del tiempo.

dentro de la red fueron la dinámica de la cantidad de transacciones y el número de nodos existentes en cada uno de los *chunks* a lo largo del tiempo. En particular, estas dos variables tuvieron un comportamiento similar durante el período de evolución de la red como puede observarse en la figura 3.4, manteniendo un coeficiente de correlación de 0,85.

Alrededor de la fecha donde el porcentaje de transacciones de tipo `pay` cayó de manera abrupta, se observó el comienzo de un comportamiento exponencial en la cantidad de transacciones y en el número de participantes dentro de cada *chunk*. Este comportamiento tuvo un punto de culminación que también coincidió temporalmente con el punto de cruce de los porcentajes mencionados anteriormente. A partir de ese punto, se observó que la conducta cambió, y el número de transacciones y de nodos por *chunk* mantuvieron un comportamiento contractivo durante todo el último período de la red.

Este tipo de comportamiento exponencial se observó en múltiples indicadores estudiados durante este trabajo, por lo que se consideró que la red en su totalidad sufrió una expansión de carácter exponencial en sus variables y luego se contrajo fuertemente hacia un nuevo punto de equilibrio menor que en su pico pero mayor que en la primera etapa de la red. Teniendo todo esto en cuenta, la evolución temporal de la red se dividió en tres períodos los cuáles fueron descritos anteriormente en la sección de metodología. Un nuevo objetivo propuesto fue el de lograr determinar variables relevantes que nos sirvieran para diferenciar estos tres períodos.

3.3. Nivel de actividad de la red

A partir del hallazgo de estos tres períodos, se buscó determinar diferencias fundamentales que distinguieran el comportamiento general de la red en cada una de estas etapas.

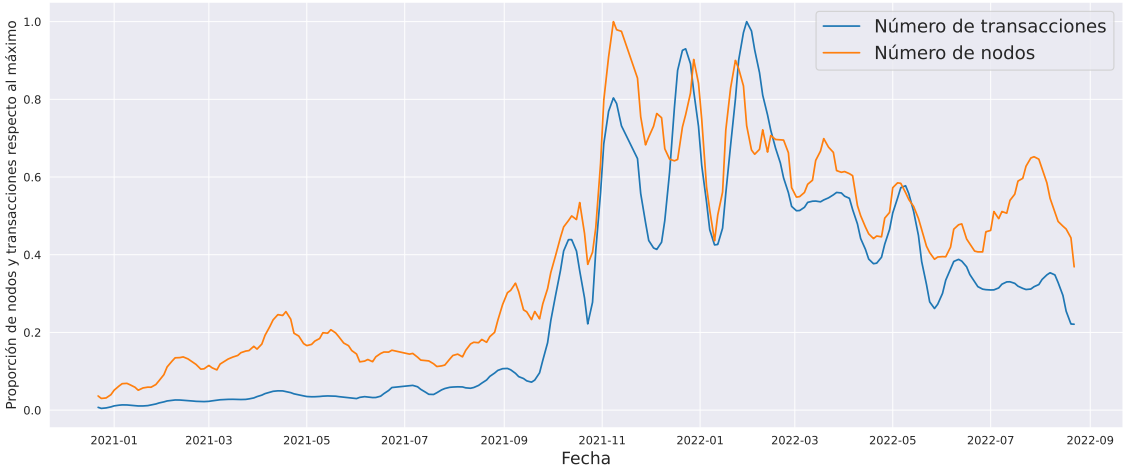


Figura 3.4: Comparación de la evolución entre el número de transacciones en la red y el número de cuentas o participantes observados en la misma.

A partir de lo observado en las transacciones, nos pareció interesante ver si el comportamiento de los nodos se mantenía a lo largo del tiempo. Es decir, si un nodo que se encontraba activo dentro de la red en los primeros *chunks* obtenidos, se mantenía de la misma manera en los últimos. Con esta motivación se realizó un análisis de actividad de los nodos. Para eso se desarrollo el *script* `checkActivity.py` que realizaba un registro de la actividad de cada uno de los nodos en cada *chunk* obtenido de la red. Si un nodo realizaba al menos una transacción durante ese *chunk*, ese nodo era considerado activo. En caso contrario era considerado inactivo. Luego sumaba todos los *chunks* donde dicho nodo se había encontrado activo y eso era lo que llamamos su actividad total. Debido al bajo muestreo no esperábamos que los nodos tuvieran actividad en todos los *chunks* obtenidos, sin embargo, los resultados obtenidos, que se pueden observar en la figura 3.5, muestran que la actividad total de más del 85 % de los nodos era igual a uno. Es decir, que solo aparecían en uno de los *chunks* pertenecientes a dicho set de datos y no volvían a aparecer en ningún otro. Esto nos indicó que en la red muy pocas cuentas se encontraban realmente activas a lo largo del tiempo y que la mayoría eran partícipes de alguna transacción pero no eran incorporadas a la red de forma recurrente.

Posteriormente se decidió realizar el mismo análisis pero tomando las transacciones de los períodos descritos anteriormente de forma separada. De esta manera, fue posible determinar si a partir de alguno de estos períodos se produjo algún cambio notable en la actividad de la red. Los resultados obtenidos se pueden ver en la figura 3.6. Se observa claramente un estiramiento de la distribución donde en un principio gran parte del total de cuentas se encontraban en el *bin* correspondiente a una sola aparición pero, a medida que la red se desarrolló, comenzaron a aparecer múltiples nodos con una actividad mucho mayor a lo largo del tiempo.

Tomando como referencia el *bin* de actividad total = 5, comparamos el porcentaje de cuentas pertenecientes a dicho *bin*. Inicialmente, en el período 1, dicho *bin* correspondía tan solo al 0,13 % de las cuentas totales de la red, en el período 2 a un

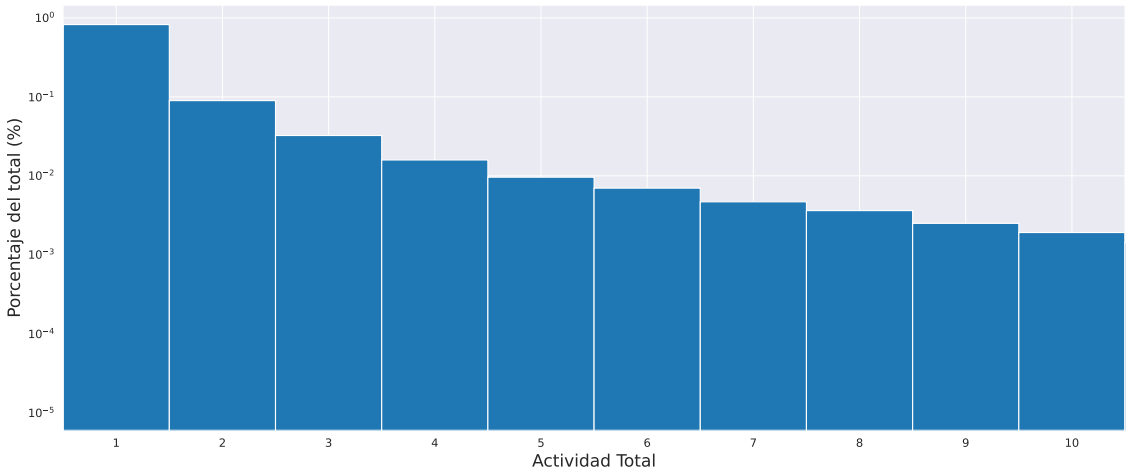


Figura 3.5: Porcentaje de cuentas acorde a su nivel de actividad en escala logarítmica. Los resultados obtenidos mostraron que alrededor de un 80 % de los nodos aparecían tan solo una vez durante toda la evolución de la red.

0,40 % mientras que en el tercer período a un 1,42 % del total de las cuentas. Entre los períodos 1 y 3 esto corresponde a un aumento significativo del número de cuentas correspondientes a dicho nivel de actividad total. Además las cuentas con actividad total mayores a 5 también dejaron de ser despreciables. Sin embargo, la mayoría de los nodos de la red, un porcentaje mayor al 80 %, mantuvieron una actividad total de uno, indicando que incluso en el período más desarrollado de la red, la mayoría de los nodos seguían apareciendo de forma extraordinaria y no de forma recurrente. Lo que pudimos concluir a partir de nuestro análisis fue no solo que el nivel de actividad total de un nodo en la red aumentó a lo largo del tiempo, sino también que incluso en el tercer período, durante el cual el número de transacciones totales y el número de nodos en la red disminuyeron, la actividad total de los participantes de la red siguió creciendo. Esto indicaría que por más que se hayan contraído el número de nodos y la cantidad de transacciones en la red durante este último período, los nodos que se mantuvieron activos tuvieron un nivel de actividad registrado mucho más elevado que en ambos períodos anteriores.

3.4. Correlación con el precio del Algo

Para diversificar los enfoques al intentar explicar la existencia de estas tres etapas dentro de la red, una propuesta fue salir del *micromundo* generado a partir de las transacciones obtenidas de forma directa y ver de qué manera se relacionaban con una variable del mundo real como podía ser el precio del activo Algo, la criptomoneda de Algorand. Al realizar este análisis esperábamos que nos pudiera dar un mayor conocimiento sobre la interconexión de la dinámica entre estas variables y si el precio del activo podía ser en parte responsable del comportamiento observado a partir de las transacciones.

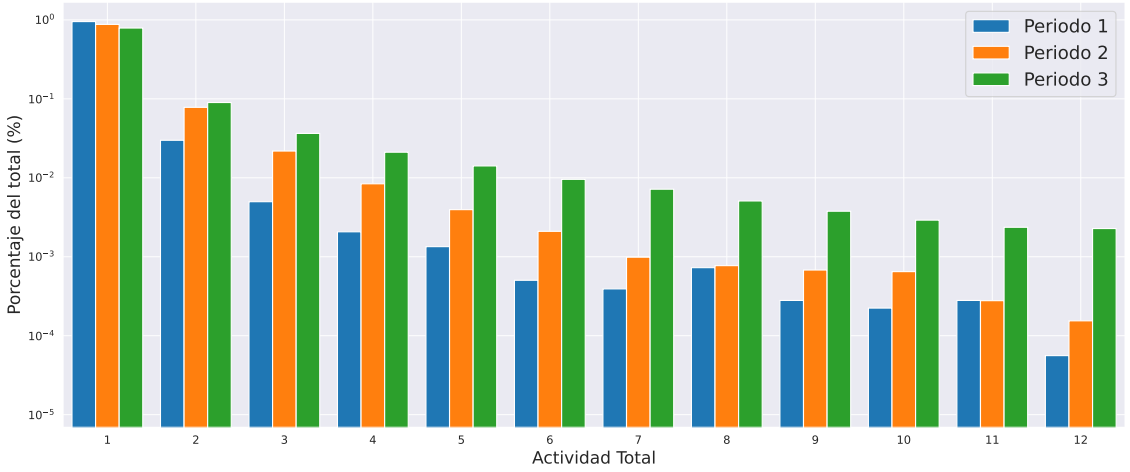


Figura 3.6: Comparación de la actividad total para cada uno de los períodos temporales hallados en la red en escala logarítmica.

Se realizaron dos análisis, uno correspondiente a la relación entre el precio de la criptomoneda Algo y el número de participantes observados en la red y el siguiente a la relación entre la cantidad de Algo transaccionados en cada uno de los *chunks* y el precio del mismo.

Para el primer análisis, dado que el número total de nodos podría considerarse como una expresión del estado de la red en términos de desarrollo, nos pareció que uno de los factores que podría estar acompañando dicho desarrollo sería el precio del activo subyacente de la red, su criptomoneda Algo. Para determinarlo, se realizó un estudio de la correlación entre ambas variables pero teniendo en cuenta la existencia de los períodos que habíamos determinado anteriormente. En particular, esta distinción se hizo para intentar determinar si existía un cambio en la relación entre las variables a lo largo del tiempo.

Los resultados obtenidos se pueden observar en la figura 3.7. Los coeficientes de correlación obtenidos fueron de 0,85 para el primer período, 0,67 para el segundo y 0,64 para el tercero. Durante toda la evolución de la red ambas variables se encontraron altamente correlacionadas. Esto muestra que el precio podría ser un buen indicador del estado actual de la red. Cómo se había mencionado anteriormente, dado que el número de nodos puede considerarse cómo una forma de describir el estado de actividad actual de la red, el precio podría entonces ser una variable posible a tener en cuenta a la hora de determinar la actividad de la red sin necesidad de tener que mirar las transacciones de forma directa. La conclusión que obtuvimos fue que el precio del activo es una representación del interés instantáneo en la red y dicho interés se corresponde bastante bien con la cantidad de nodos y, en consecuencia, con el nivel de actividad del sistema.

Para la segunda variable a analizar se realizó un estudio similar, los resultados se pueden observar en la figura 3.8. Lo que esperábamos al intentar relacionar estas dos variables era determinar si existía una relación entre el precio y la cantidad de moneda que se transaccionaba en la red. En caso de que la expresión del precio en

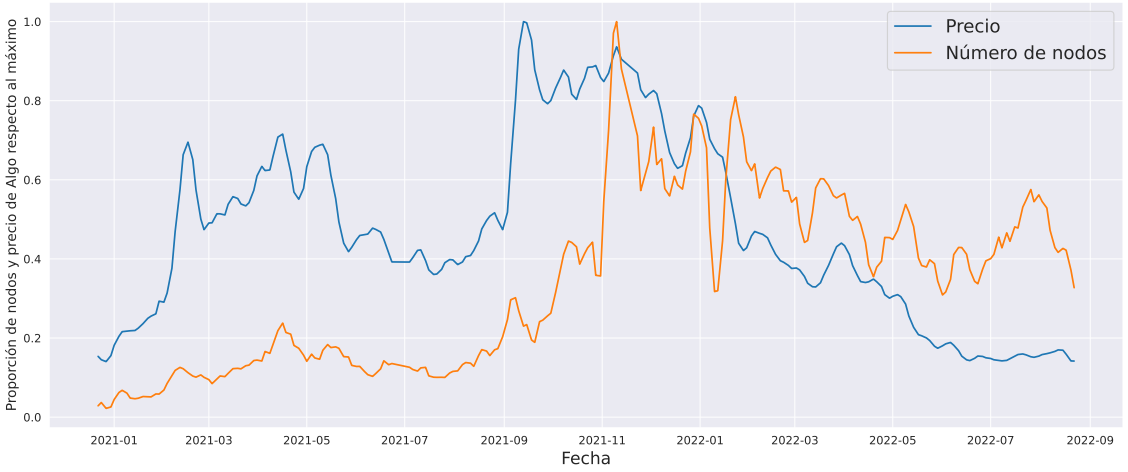


Figura 3.7: Comparación de la evolución de dos variables analizadas, el precio del valor del ALGO, la criptomoneda asociada a la red, y el número de nodos en la red.

dólares de la criptomoneda asociada a la red creciera, se esperaría que en consecuencia, el número de ALGO transaccionados por *chunk* fuera menor. Debido a que el inversionista promedio no suele tener un poder adquisitivo suficiente para realizar transacciones las cuales expresadas en dólares fueran de montos de elevados, se esperaría entonces que la subida del precio de la criptomoneda perjudicara su posibilidad de intercambiar altas cantidades de ALGO en la red, por lo menos mientras que el precio del ALGO se mantuviera elevado, llevando a una disminución en la cantidad de criptomonedas transaccionadas por *chunk*.

Sin embargo, en contraste con lo esperado, que sería una alta anticorrelatividad entre ambas variables, los resultados para la correlación obtenidos en cada período fueron de 0,33 para el primer período, 0,45 para el segundo y $-0{,}10$ para el tercero. Estos resultados no muestran una tendencia clara, por lo tanto, pudimos determinar que el precio del activo no parecía ser un factor importante a la hora de determinar la cantidad de ALGO transaccionado en la red. Sin embargo, el punto con la mayor cantidad de ALGO transaccionado en un mismo *chunk* se observó durante el tercer período, donde el precio era cercano al mínimo registrado desde el comienzo de la historia de la red. Esto podría indicar el comienzo de una nueva tendencia cercana a lo que esperabamos dados los niveles de cantidad de ALGO en los últimos *chunks* analizados, pero dado el resultado obtenido para la correlación no podemos afirmar eso sin obtener datos adicionales a los estudiados en este trabajo.

3.5. Distribución del grado de los nodos

Como última variable a analizar, se procedió a estudiar la distribución del grado de los nodos pertenecientes a la red en cada una de las etapas definidas. Se decidió hacer este análisis dado que consideramos que esta variable era una forma relevante de describir el estado actual del sistema, y por lo tanto poder definir, a partir de

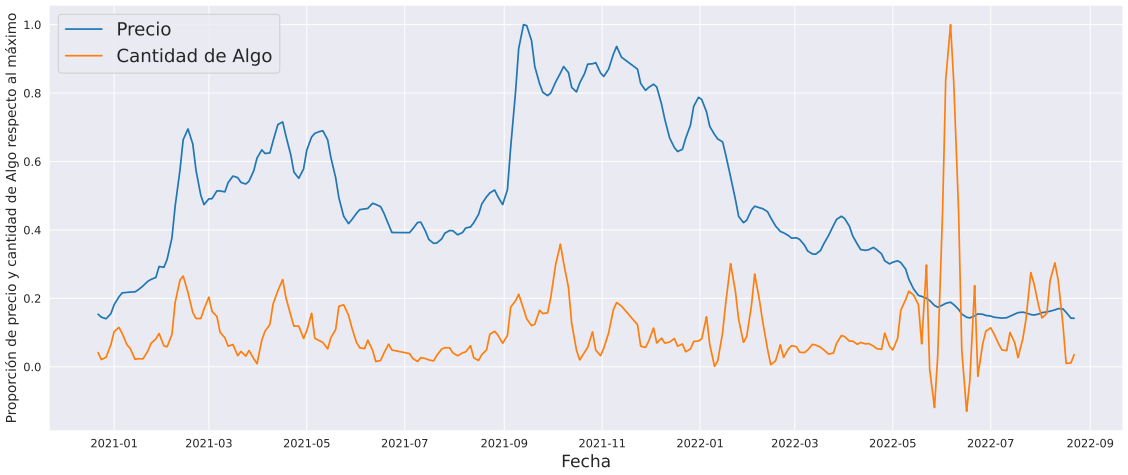


Figura 3.8: Comparación de la evolución de dos variables analizadas, el precio del valor del ALGO, la criptomoneda asociada a la red, y la cantidad de ALGO transaccionada en cada *chunk* analizado.

ésta, una diferencia entre cada una de las fases presentes durante su evolución.

Para realizar este análisis se utilizó el *Data set 4*, el cual había sido filtrado utilizando el algoritmo descrito en la sección de metodología. Se utilizaron inicialmente tres modelos y el modelo de Barabási-Albert que se tomó como referencia. Los modelos utilizados, fueron descritos en la introducción, donde incluimos la posibilidad de que la red se comportara como una *random network*, es decir, cuya distribución se viera explicada a partir de la distribución de Poisson, también se consideró una serie de potencias, donde en el caso de que el parámetro particular de esta distribución se encontrara entre los valores de 2 y 3 se consideraría que la red era de escala libre y por último, se incorporó también el modelo de *non-linear prerential attachment* en el caso en que el valor del parámetro $\alpha < 1$, para el cual teníamos una expresión analítica posible en ese caso que llamaremos exponencial estirada de ahora en más.

Se utilizó la función $curve\_fit$ para realizar un barrido sobre los parámetros de cada uno de los modelos propuestos, el parámetro γ en el caso de la serie de potencias y el parámetro α para la exponencial estirada, y así encontrar los valores óptimos para estos. Se ajustaron los histogramas del grado de los nodos obtenidos para cada uno de los *chunks* pertenecientes a cada período. Para determinar cuál era la expresión que mejor describía los datos en cada uno.

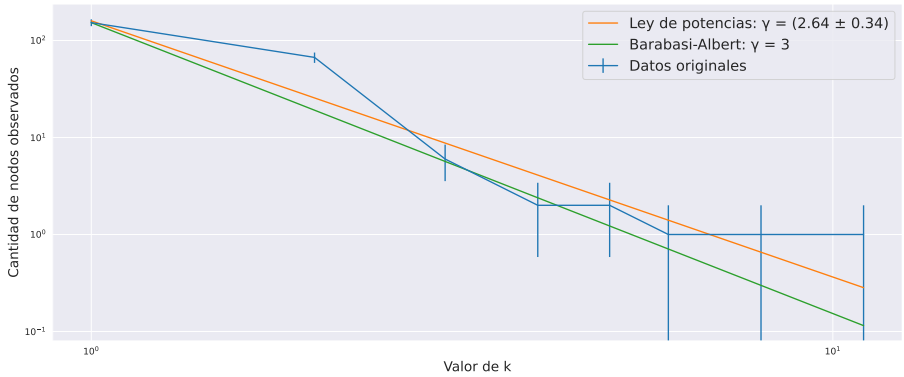
Como primer resultado, se determinó que la distribución de Poisson no era una buena descripción del problema a lo largo todos los períodos existentes. Esto se debió a que al realizar el ajuste descrito anteriormente, la función no parecía ser una buena explicación para los resultados obtenidos empíricamente. En consecuencia, decidimos descartar este modelo como posible explicación de los datos. Por el contrario, los otros dos modelos fueron bastante buenos a la hora de describir el comportamiento de la evolución del grado de los nodos de la red a lo largo del tiempo.

Centrándonos particularmente en estos dos modelos, se ajustaron todos los *chunks* generados para el *Data set 4*, manteniendo el modelo de Barabási-Albert como refe-

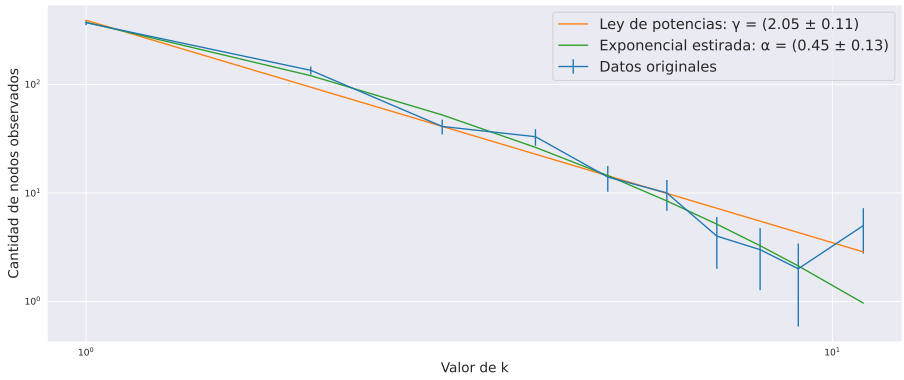
rencia. Una de las primeras cosas a destacar fue que el modelo de exponencial estirada no devolvía buenos resultados al intentar ajustar, e incluso fallaba en múltiples ocasiones durante todo el período 1 y parte del período 2. Inicialmente esto casi nos lleva a descartar el modelo. Sin embargo, a partir de aproximadamente la mitad del período 2, el comportamiento de la función ajustada empezó a ser una descripción muy buena de los datos empíricos, lo cual nos hizo mantener dicho modelo como una posible forma de explicar el comportamiento del sistema pero centrado en esta sección particular del tiempo de vida de la red. Por el contrario, la expresión para la serie de potencias fue una descripción muy buena durante el primer período de desarrollo de la red, pero a medida que el sistema evolucionó se alejó cada vez más de ser una buena descripción de los datos empíricos.

Estos dos procesos nos indicaron que inicialmente el sistema se encontraba muy cerca de una descripción utilizando el modelo de red libre de escala, una ley de potencias, pero que a medida que se desarrolló y más participantes se incorporaron dicha descripción dejó de ser la mejor explicación para los resultados obtenidos y el sistema pasó a estar descrito por una exponencial estirada, debido a un cambio en el apego preferencial del sistema, donde inicialmente la probabilidad de que un nodo se conectara a un nodo con valor de k grande era mayor que en el período final de la red. Esto podría explicarse teniendo en cuenta que la cantidad de nodos que existían inicialmente en la red era muy baja, y por lo tanto los nuevos participantes tenían muy pocas opciones a la hora de decidir con quién conectarse, la mayoría terminaban uniéndose a los *hubs* centrales y por lo tanto llevando el sistema a uno descrito por el modelo de Barabási-Albert. Sin embargo, luego con la incorporación de nuevos jugadores, los nuevos nodos que se incorporaron al sistema tenían mayores opciones a la hora de conectarse con nodos preexistentes llevando a una disminución en la centralidad de los *hubs* presentes en el sistema. Este cambio radical en el comportamiento del sistema nos permitió determinar entonces la existencia de tres períodos en la red, el primer período, con una serie de potencias, un segundo período de transición donde ambas descripciones eran comparables al intentar explicar los datos y un tercer período del sistema descrito por la exponencial estirada. Para mostrar el cambio del comportamiento en cada una de estas etapas, en la figura 3.9 se puede observar un ejemplo de los ajustes obtenidos para cada uno de los períodos en escala logarítmica. Una consideración adicional fue que en el caso del período 1 sólo fue posible ajustar el modelo de escala libre, mientras que para el período 2 y 3 ambos fueron ajustados.

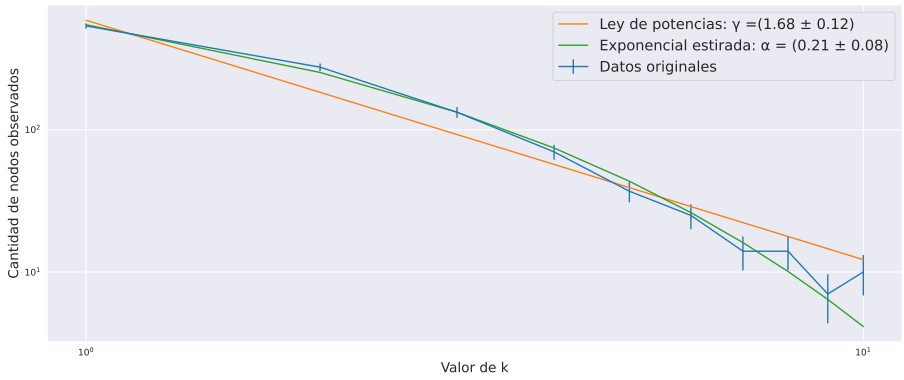
Durante el primer período, se observó que el parámetro γ de la ley de potencias evolucionó hacia el valor dado por el modelo de Barabási-Albert, donde $\gamma = 3$. Es decir que este modelo, por lo menos durante una parte de este primer período, era una buena descripción del sistema, y podría utilizarse la matemática que acompaña dicho modelo para poder sacar mayores conclusiones. Una particularidad de este primer período fue que al tener tan pocos puntos al graficar, dado que el grado de la mayoría de los nodos era k = 1, no fue sencillo confirmar que, dado nuestro muestreo, estábamos captando la distribución real de la red. Esto nos llevó a proponer posteriormente la generación de *super sets*, los cuáles se encuentran en el *Data set*



(a) *chunk* 14 300 000.



(b) *chunk* 16 450 000.



(c) *chunk* 20 450 000

Figura 3.9: Resultados obtenidos al ajustar los modelos propuestos para cada uno de los períodos temporale hallados en la red en escala logarítmica.

5, que se discutirá mas adelante.

Durante el segundo período, obtuvimos dos situaciones distintas. En los comienzos de esta segunda etapa, la ley de potencias era una muy buena aproximación a los datos, pero la exponencial no lograba ajustarse bien y por lo tanto fue considerada como una mala descripción en estos casos. Posteriormente, en una segunda etapa dentro de este segundo período la exponencial estirada comenzó a ajustarse de forma correcta a los datos y la ley de potencias continuó siendo una buena descripción del sistema, sin embargo comenzó a notarse que el ajuste no se acercaba del todo a los datos empíricos, particularmente para los grados más altos graficados en esta distribución, es decir para los $k \geq 8$. Una observación fue que para el valor observado de $k = 2$, esta descripción falló de forma sistemática al intentar acercarse a dicho punto, es decir que, para puntos intermedios en general fue una buena aproximación pero la cola de la distribución empezó a ser menor a la recta generada por la ley de potencias y para $k = 2$ se obtuvo un valor mayor que lo generado por dicho modelo. Por el contrario, en esta segunda etapa del segundo período, el modelo de exponencial estirada comenzó a tener muy buenos resultados para todos los valores de k.

Durante el período 3, nos llamó fuertemente la atención que por más que las variables del sistema analizadas anteriormente se encontraban en una etapa de contracción, contrario a lo que sucedió durante todo el segundo período de la red, los resultados obtenidos al ajustar ambos modelos no cambiaron de manera significativa. Por el contrario, la única diferencia con el final del período anterior fue una acentuación en la diferencia a la hora de explicar los datos de ambos modelos, donde el modelo de exponencial estirada comenzó a dar resultados muy superiores a la ley de potencias. Esta diferencia se mantuvo a lo largo de todo el período 3 de la red. Algo llamativo fue que uno esperaría que al retornar las variables a un valor más cercano a los originales en el período 1, el comportamiento de la red retornara también más cerca al punto de origen, es decir, esperaríamos que la ley de potencias volviera a ser una buena descripción del sistema al contraerse el mismo. Esto tendría sentido, ya que al volverse más chica la red, los nuevos nodos tenderían a conectarse más con los nodos de mayor grado debido a su centralidad en la red y dado que el número de nodos totales descendió se esperaría ver un crecimiento en el *preferential attachment* de la red justamente reflejando este cambio en el número de participantes. Sin embargo, por lo menos a la hora de cuantificar los resultados del ajuste, no se observó un retorno a la ley de potencias, ni tampoco la exponencial estirada dejó de poder ajustarse de manera consistente.

Esto podría deberse a que, por más que en este nuevo período la red se encontrara en una etapa de contracción, la depuración de nodos se realizara en participantes que no eran esenciales para la red y solo quedaran los que realmente se encontraban realizando alguna actividad de forma permanente. Esto llevaría a que por más que no se observara la misma cantidad de nodos totales en el sistema y sus variables se contrajeran, el atractivo de cada nodo sería mayor debido a que la centralidad de los nodos remanentes sería mucho mayor que la observada en el período anterior.

Para poder cuantificar dicho comportamiento, se tomaron los diferentes valores

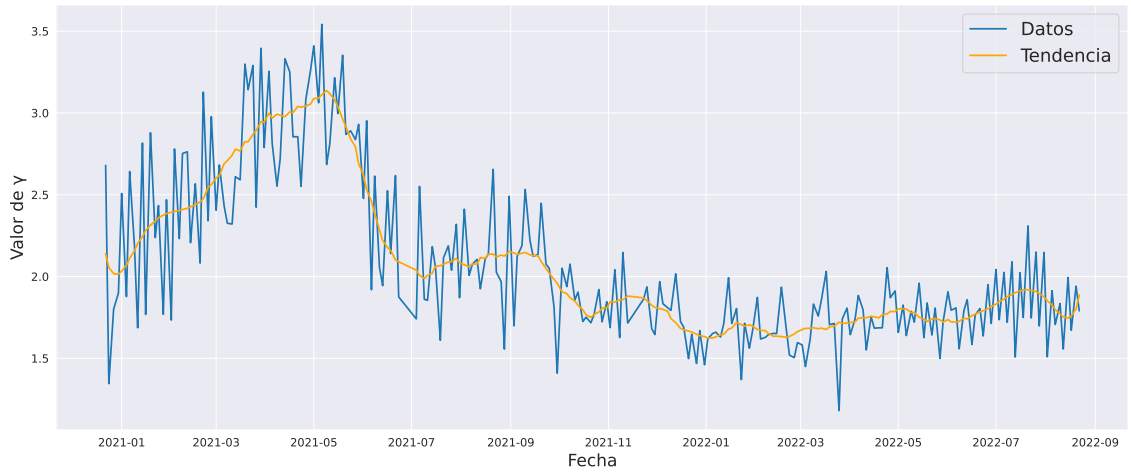
obtenidos para el parámetro γ perteneciente al modelo de escala libre y al α del modelo de exponencial estirada y se estudió su evolución a lo largo del tiempo, lo que puede verse en las figuras 3.10a y 3.10b. En el caso del parámetro α correspondiente a la exponencial estirada, dado que no logró ser ajustada de forma consistente hasta mediados del período 2, se realizó un corte para comenzar a partir del *chunk* correspondiente al bloque 16 000 000, a partir del cual el comportamiento del ajuste fue más acorde al que mantuvo a lo largo de los *chunks* siguientes. La falta de convergencia del ajuste podría deberse a múltiples factores, pero en particular creemos que se debió al bajo número de puntos obtenidos para realizar los ajustes. Dado que en el primer período particularmente la mayor cantidad de nodos en la distribución se encontraba con grado 1 o 2, llevando a que en algunos casos los puntos obtenidos para realizar el ajuste fueran entre 2 y 4, otorgándole una gran dificultad al algoritmo para lograr hacer converger una función exponencial.

En el caso del parámetro γ, se observaron tres fases claras que determinaron su proceso de evolución. En una primer fase, el valor del parámetro se mantuvo encapsulado en el rango de 1,5 a 3,5 saliendo de estos valores en tan solo dos ocasiones. La mayor concentración de puntos se mantuvo en el rango de 2 a 3, el cual era el rango de valores dentro de los cuáles una red podía ser considerada libre de escala.

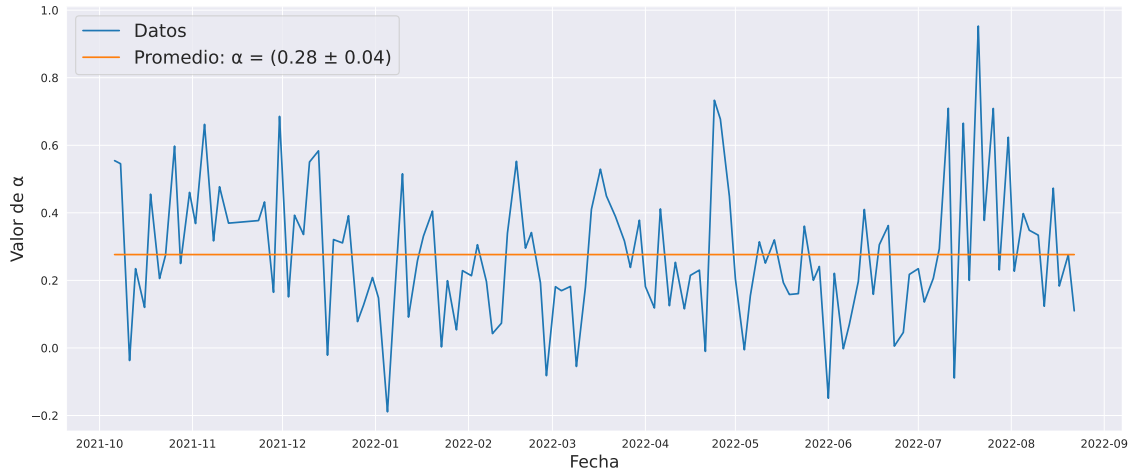
Algo llamativo fue que durante este primer período el valor del parámetro evolucionó en ese rango siguiendo una lineal con pendiente positiva. Este comportamiento implicaba un descenso de la concentración en nodos centrales del sistema. Esto se deduce de la forma de la serie de potencias, donde a mayor valor de γ, menor es la representación dentro de la distribución de nodos de grados más altos y mayor es la concentración en nodos de menor grado, los cual indicaría que la mayoría de los nodos tenían un grado bajo, por lo que la centralización de la red sería menor.

Sin embargo, con el comienzo del período 2, de forma precipitada, al igual que en todos los resultados observados a lo largo de esta tesis, se observó un cambio cualitativo en el comportamiento de dicha evolución. El valor del parámetro γ tomó una pendiente negativa y evolucionó desde valores cercanos a $\gamma = 3$ hasta asentarse alrededor del rango de valores finales entre 1,5 a 2 aproximadamente. Este cambio en el comportamiento del sistema podría implicar que, debido al aumento exponencial de participantes de la red, por más que el número de nodos con un valor de k bajo aumentó, la cantidad de participantes que aumentaron su centralidad en el sistema y por lo tanto, su valor de k, fue incluso mayor, llevando a un estiramiento de la función debido a un mayor porcentaje de nodos con grados $k \geq 3$.

El comienzo del tercer período se designó coincidiendo temporalmente con el que se había determinado anteriormente a lo largo de esta tesis. A partir de este momento, el parámetro se mantuvo dentro de un rango entre 1,5 a 2. En particular no se observó un cambio significativo en el comportamiento del sistema en comparación con la segunda mitad del período 2, en donde el modelo de exponencial estirada comenzó a estar vigente. Sin embargo, algo que sí se notó, fue un leve aumento del valor del parámetro cerca del final del tercer período el cual se puede observar en el gráfico. Esto podría deberse al comienzo de una nueva etapa en la red, pero no es posible confirmarlo debido a la falta de datos más actuales de la red. Uno de los



(a) γ



(b) α

Figura 3.10: Evolución de los parámetros γ (ley de potencias) y α (exponencial estirada) en el tiempo.

análisis posibles para un trabajo futuro podría ser incorporar datos actualizados del sistema y estudiar la evolución de esta variable.

En el momento en que el decrecimiento de la variable cesó y comenzó a mantenerse dentro de un rango se designó el comienzo de un tercer período para la variable de la misma manera que habíamos observado en múltiples otras analizadas hasta ahora. En ese nuevo rango ya no se podía afirmar que una red libre de escala pudiera ser una buena interpretación de la distribución del grado de los nodos en la red y por lo tanto del estado actual del sistema.

Consideramos que los resultados obtenidos para esta variable fueron acordes a lo esperado dado lo analizado hasta ahora, sin embargo los mejores resultados para el modelo de serie de potencias se obtuvieron durante el período 1 y parte del período 2, dado que posteriormente el modelo de exponencial estirada se convirtió en la mejor expresión para explicar la evolución del sistema, por lo que por más que hubiéramos podido obtener un posible valor del parámetro γ, no necesariamente sería una buena explicación del sistema estudiado.

En el caso del parámetro α del modelo de la exponencial estirada, el cual comenzó a tener relevancia recién a fines del segundo período de la red debido a problemas con la convergencia del modelo, tuvo un comportamiento dentro del rango esperado en el caso de que buscáramos determinar que efectivamente la red se encontraba en una etapa de *sublinear preferential attachment* como habíamos supuesto desde un principio. Esto se debió a que el rango esperado para un sistema de este tipo debería ser con un valor de α de 0 a 1 y efectivamente se observó $\alpha_{prom} = (0{,}28 \pm 0{,}04)$ para el rango temporal en el que este modelo tuvo relevancia. Al igual que en lo discutido anteriormente para el caso del parámetro γ, no se logró identificar una tendencia de variación del valor del parámetro α sino que se mantuvo dentro de un rango de 0 a 0,8. Una de las características observadas de este segundo modelo fue que el parámetro α en varias ocasiones se ajustó con valores negativos. Esto sucedió durante todo el primer y segundo período dado que o el algoritmo no lograba ajustar la función y devolvía un error, o la curva generada no se acercaba a los datos obtenidos empíricamente. En particular, también durante el tercer período, pero de forma mucho menos regular, la convergencia de la función no fue muy buena y devolvió valores de α negativos, lo cual se observa claramente en el gráfico. Sin embargo, la cantidad de veces que esto ocurrió en comparación con las que sí logró ajustar de manera adecuada a los datos fueron muy pocas.

Para establecer de forma cuantitativa qué modelo tuvo mejores resultados a la hora de intentar explicar los datos empíricos obtenidos se eligió el R^2 para lograr determinar, no solo que efectivamente el ajuste sobre los datos era bueno, sino también poder realizar una comparación entre los resultados de los ajustes obtenidos para ambos modelos y poder posteriormente determinar cuál de los dos modelos había logrado explicar el fenómeno observado de mejor manera. El método utilizado fue un sistema de *threshold*, donde se tomó el valor de $R^2 = 0{,}95$ como el límite para determinar que efectivamente era un buen ajuste. A partir de esto se obtuvo el porcentaje de puntos que superaron este límite. En el caso del modelo de la exponencial estirada, su valor de R^2 a lo largo del tiempo superó este valor un

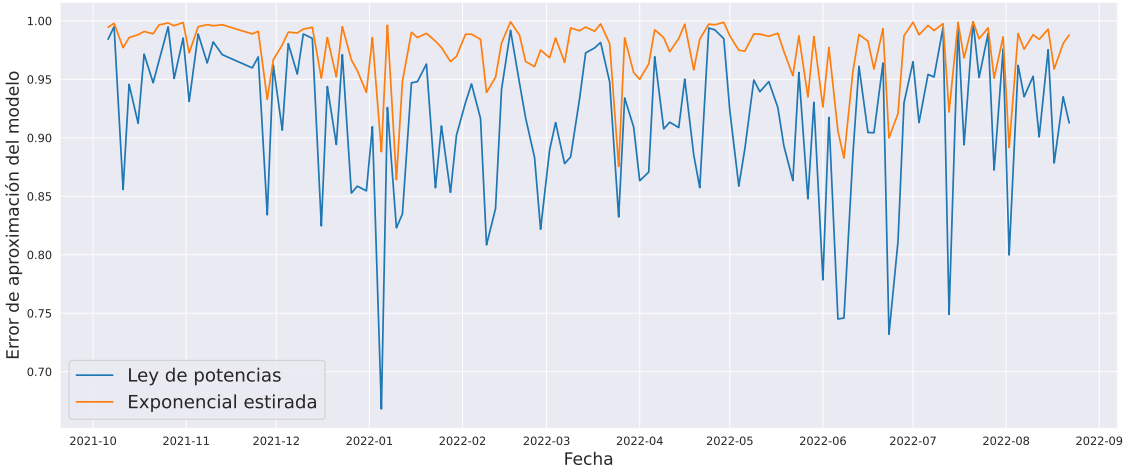


Figura 3.11: Resultados obtenidos para el valor de R^2 de ambos modelos a lo largo del tiempo, centrado en la parte final del segundo período y todo el tercero.

89,78 % de las veces, mientras que el de la ley de potencias, tan solo lo hizo un 26,8 % de las veces, ubicando su promedio general alrededor de 0,90 en este rango temporal de los períodos 2 y 3. En la figura 3.11 se observa el comportamiento general del R^2 a lo largo de todos los *chunks* pertenecientes al período en el cual el modelo de exponencial estirada fue relevante. En particular, se destacaron algunos *chunks* en donde el valor de R^2 descendió fuertemente para ambos modelos y al compararlos con los resultados obtenidos en los ajustes se observó que algunos coincidían con los *chunks* en los cuáles el valor del parámetro α era negativo, es decir, donde la exponencial claramente no era una buena explicación de los datos estudiados.

Dada la naturaleza de nuestros datos, los resultados obtenidos para este parámetro fueron variables, sin embargo, el modelo de exponencial estirada tuvo un valor de R^2 mayor al de serie de potencias de forma consistente, mostrando así que efectivamente, por lo menos durante el segundo y tercer período donde ambos modelos fueron utilizados, este modelo fue mejor al intentar ajustarse a los datos obtenidos y por lo tanto sería nuestro mejor candidato al intentar ajustar nuevos datos en el futuro.

3.5.1. Super sets

Como se comentó anteriormente, una preocupación que se tenía principalmente durante el primer período de la red, donde luego de realizar el proceso de filtrado los nodos remanentes eran alrededor de 100 en los primeros bloques analizados, temíamos que las distribuciones obtenidas no fueran representativas de lo que realmente estaba sucediendo dentro de la red. Nuestra solución consistió en estudiar una serie de nuevos datos, el *Data set 5*, los cuáles fueron llamados *super sets*. Estos consistieron en un solo *chunk* por período, seleccionando el período temporal que nos pareció más representativo de cada etapa y para cada uno de estos *super sets* se tomaron 20 000 bloques consecutivos. De esta manera nos aseguramos de que efec-

tivamente la distribución del grado de los nodos fuera representativa del sistema en ese momento. Los *chunks* de bloques seleccionados para cada *super set* fueron:

- Período 1: 14 000 000-14 019 999
- Período 2: 17 200 000-17 219 999
- Período 3: 20 000 000-20 019 999

Los resultados obtenidos para estos *super sets* en escala logarítmica se pueden observar en la figura 3.12, donde se muestran los ajustes obtenidos para cada uno de los períodos estudiados.

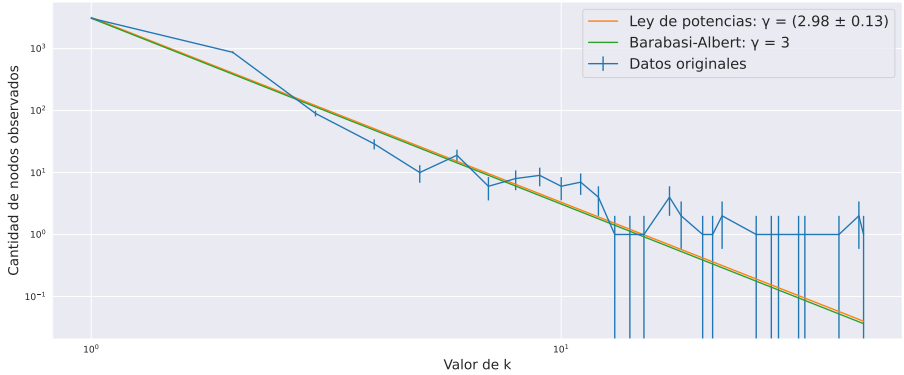
En el caso del período 1, para un *chunk* dentro del rango de bloques que abarcaba el primer *super set*, el valor del parámetro γ se encontraba en $\gamma = (2{,}84 \pm 0{,}52)$, y para el *super set* el valor obtenido fue de $\gamma = (2{,}98 \pm 0{,}13)$, es decir que para el *super set* obtenido el resultado del valor del parámetro γ se encontraba dentro del rango de error hallado anteriormente. Este resultado logró mermar nuestra preocupación de que, particularmente para el primer período, el número de nodos no fuera suficiente para obtener una distribución del grado de los nodos que fuera representativa de lo que realmente sucedía dentro de la red.

Los valores obtenidos para el parámetro γ durante el segundo y tercer períodos fueron $\gamma = (2{,}17 \pm 0{,}04)$ y $\gamma = (1{,}95 \pm 0{,}03)$. Los valores obtenidos anteriormente habían sido de $\gamma = (2{,}04 \pm 0{,}08)$ y $\gamma = (1{,}74 \pm 0{,}13)$ respectivamente. En ambos casos, el valor obtenido para el parámetro γ en los *super sets* se encontró dentro de 2σ del valor obtenido anteriormente.

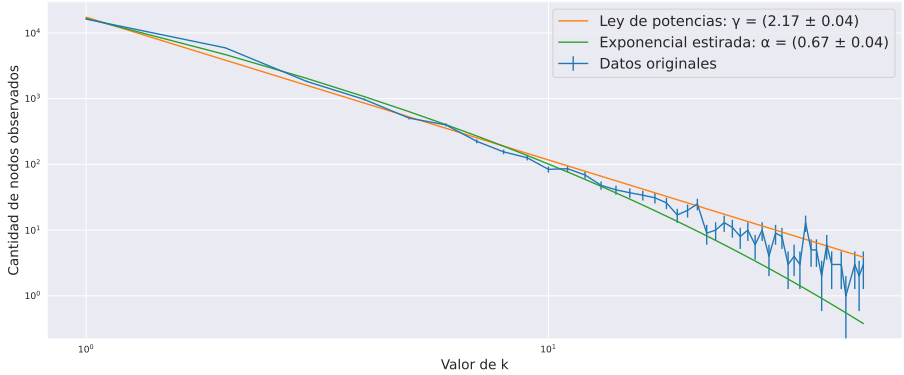
Los valores obtenidos para el parámetro α fueron de $\alpha = (0{,}67 \pm 0{,}04)$ y $\alpha = (0{,}72 \pm 0{,}04)$ respectivamente que se comparan con los obtenidos anteriormente de $\alpha = (0{,}66 \pm 0{,}08)$ para el segundo período y $\alpha = (0{,}24 \pm 0{,}13)$. El valor obtenido para el segundo *super set* se encontró dentro del rango de error del obtenido anteriormente. Sin embargo, en el caso del tercer período, el valor obtenido para el parámetro se encontró muy por fuera del error obtenido anteriormente. No solo el valor obtenido para α no se acercó al valor anterior, sino que también superó el valor promedio del parámetro, ubicándose como un valor casi límite observado para esta distribución. A partir de este resultado se intentó buscar una explicación para la existencia de semejante diferencia entre los valores para este parámetro en el tercer período, se probó analizando múltiples *chunks* alrededor del período de 20 000 bloques desde los cuáles fue generado dicho *super set* y se analizaron sus valores de α. En todos los casos, los valores hallados se encontraron cercanos al promedio, lejos del valor obtenido para el *super set*. Esto podría estar indicando que para este último período, el muestreo utilizado durante esta tesis podría no estar siendo suficiente para poder dar una explicación certera de lo que efectivamente estaba pasando dentro de la red. Esto podría ser un tema a explorar en un trabajo futuro.

En comparación con lo obtenido para el *Data set 4*, en todos los casos se registró un menor error para los parámetros debido a que el nuevo muestreo permitió obtener una cantidad de nodos mucho mayor al promedio obtenido para los *data*

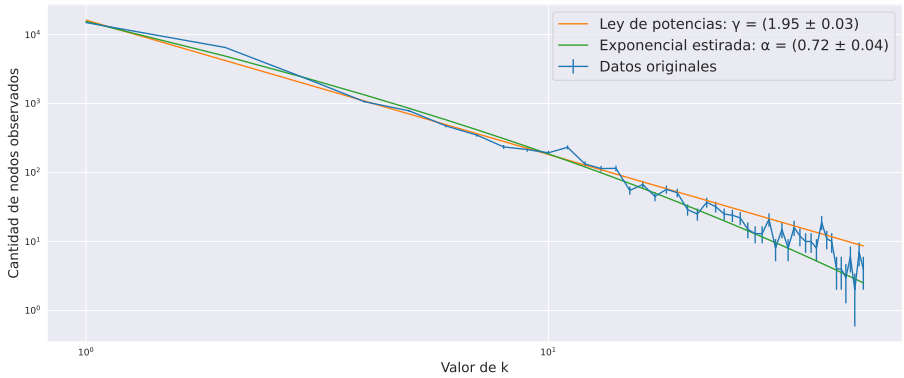
sets anteriormente estudiados y por lo tanto, el error estadístico cometido disminuyó considerablemente. Esto podría indicar que una estrategia posible para un futuro estudio podría ser la incorporación de un muestreo con menos cantidad de *chunks* pero un mayor número de bloques por *chunk*, a modo de minimizar el error estadístico cometido para determinar cada uno de estos parámetros.



(a) Primer Periodo



(b) Segundo Periodo



(c) Tercer período

Figura 3.12: Resultados obtenidos para los ajustes realizados de los *super sets* en escala logarítmica.

Capítulo 4

Conclusiones y trabajo futuro

A lo largo de este trabajo nos propusimos estudiar, exclusivamente a través de sus transacciones, la red de Algorand. Para ello fue necesario afrontar múltiples desafíos a lo largo de estos meses de trabajo. Primero debimos determinar la forma de obtener los datos, lo cual fue resuelto utilizando una API. También de organizarlos y curarlos para su posterior análisis, lo cual fue realizado a partir del desarrollo de múltiples *scripts* en `Python`, en particular destacando el proceso de filtrado, el cual fue un elemento central a la hora de permitir generar buenos resultados a lo largo de esta tesis gracias a la eliminación de los dos tipos de nodos no deseados, los nodos *spam* y los nodos solitarios.

Habiendo comenzado el análisis obtuvimos múltiples resultados interesantes abarcando numerosas aristas que involucraban al sistema de Algorand. Éstas incluyeron la observación del comportamiento de los tipos de transacciones existentes en la red y qué porcentaje del total de las transacciones existentes ocupaba cada tipo. El resultado obtenido para esta variable fue el puntapié inicial para lo que seguiría posteriormente en este trabajo. Dado el desarrollo temporal particular observado en esta variable se intentó determinar posteriormente a lo largo de toda esta tesis la existencia de tres períodos temporales durante la red con comportamientos muy diferenciados entre sí.

Habiendo realizado el análisis sobre numerosas variables logramos confirmar la existencia de los tres períodos. En un principio, la red tenía un nivel de actividad general bastante baja, con tan solo alrededor de 100 participantes y transacciones por *chunk* observado. Posteriormente en un segundo período, se observó un cambio radical en el comportamiento general de red con un crecimiento exponencial de todas las variables asociadas y por último en un tercer período se observó un decrecimiento de las variables en comparación con el máximo observado en el período anterior. Sin embargo, algo que caracterizó a este último período fue un proceso de estancamiento a lo largo de toda la evolución en esta etapa.

Para poder, efectivamente, dividir la evolución temporal de la red en múltiples períodos, fueron analizadas numerosas variables que nos posibilitaron concluir su existencia. Entre ellas, analizamos el porcentaje de transacciones remanentes luego del proceso de filtrado, el cual nos dio los primeros indicios de la existencia de

los períodos, ya que mostraba un cambio drástico en esta variable en simultáneo al observado en la variable discutida anteriormente. También, obtuvimos la correlación entre el número de transacciones y de cuentas observadas en cada *chunk*, el cual al igual que las variables anteriores mostraron un comportamiento similar a las previamente analizadas.

Por otro lado, se analizó también la actividad total de la red a lo largo de todos los *chunks*. En particular, se buscaba ver si la aparición de los nodos era recurrente a lo largo de la evolución temporal o simplemente se trataba de nodos que interactuaban con la red de forma aislada. Inicialmente, se observó una red altamente aislada, con la mayoría de los nodos no apareciendo más de una vez en un período temporal. Sin embargo, durante los dos períodos siguientes se observó una mayor interconexión temporal en la red, donde el porcentaje de cuentas que reaparecían en múltiples *chunks* comenzó a no ser despreciable. Esto comprobó una diferencia fundamental en el grado de interconexión de la red entre los tres períodos observados.

También, se estudió la correlación entre dos variables de la red y el precio de la moneda del sistema, el cual era una variable externa al sistema estudiado, por lo cual hasta ahora no había sido un factor utilizado para obtener conclusiones sobre su comportamiento. Los resultados hallados mostraron una alta correlación entre el precio y el número de nodos del sistema, es decir que el precio podría utilizarse como un indicador al intentar determinar la tendencia de crecimiento de la red. Por otro lado, al estudiar el precio y la cantidad de Algo transferidos por *chunk* no se obtuvieron los resultados esperados, y por el contrario, a lo largo de toda la evolución observada, se logró determinar que el precio no influía en la cantidad de Algo transaccionados en cada *chunk*.

Por último, se analizó la distribución del grado de los nodos en cada *chunk*, donde se obtuvieron resultados consistentes con lo observado en la evolución de la red. En particular, fue posible describir los datos obtenidos usando los modelos de redes propuestos al comienzo de esta tesis. Durante este estudio se logró determinar una gran diferencia entre la capacidad de cada modelo de describir el sistema en cada uno de los períodos propuestos, reafirmando nuestra creencia de la existencia de los mismos en la historia de la evolución del sistema. El único modelo que logró ajustar a los datos en el primer período fue el de la ley de potencias. Mientras que en los períodos 2 y 3, logramos incorporar el modelo de exponencial estirada como un posible contendiente para explicar los resultados obtenidos empíricamente y, efectivamente, nuestras conclusiones fueron que el segundo modelo logró explicar los datos mejor que el propuesto para el primer período.

Además, con el estudio posterior realizado utilizando los *super sets*, logramos anular la posibilidad de que el muestreo utilizado durante los períodos 1 y 2 no fuera suficiente para poder determinar si los datos obtenidos eran efectivamente una buena descripción del estado del sistema analizado. Por lo que, a partir de los resultados obtenidos para estos *super sets* y en comparación con los de los *data sets* estudiados anteriormente, se determinó que para los períodos 1 y 2, el sistema era invariante de escala. Por otro lado, en el caso del período 3 se observó una diferencia significativa en el resultado obtenido para el parámetro α del *super set* y de los

data sets anteriores indicando que durante este último período de la red no era posible afirmar que el sistema se encontraba en un estado invariante de escala. Por el contrario, los resultados obtenidos sugerían que había una pérdida de información debido al tamaño de los *chunks* utilizados durante este trabajo. Un posible trabajo futuro podría ser indagar sobre esta cuestión e intentar acertar el muestreo óptimo para este tipo de sistemas que logre describir al sistema de la mejor manera posible.

Debido a la evolución contiñua de la red al pasar el tiempo, con nuevas actualizaciones al protocolo y la aparición de múltiples nuevos participantes es difícil determinar si la vigencia de los resultados obtenidos en esta tesis se mantendrán para trabajos futuros. Una de las cosas que consideramos debería incorporarse es la comparación constante con datos más actualizados a los usados durante este proceso de tesis. De esta manera podría determinarse si se le ha dado lugar a un nuevo período en la red, o si el último estudiado sigue vigente.

Por otro lado, un enfoque posible para determinar el *preferential attachment* del sistema, el cual para mantener acotado el alcance de esta tesis fue dejado afuera, podría ser utilizando el mecanismo presentado por Barabási en su libro. Dicho análisis, brindaría una nueva forma de obtener el valor del parámetro α, el cual fue estudiado en la sección de distribución del grado de los nodos. Esto permitiría comparar los resultados obtenidos con los datos utilizados con este nuevo valor de α y de esta manera confirmar si la obtención del parámetro a partir del ajuste de dicha función fue una buena estrategia.

Yendo incluso un poco más, podría comenzar a estudiarse un nuevo fenómeno el cual no tuvo lugar en los análisis presentados a lo largo de este proceso de tesis, el cual es el *fitness*, también altamente discutido en el libro de Barabási. La posibilidad de existencia de *fitness* por parte de los nodos existentes en la red, indicarían la necesidad de proponer otros modelos que también tuvieran en cuenta este fenómeno.

En el caso del estudio de las transacciones, el auge en la utilización de aplicaciones y de configuración y transferencia de activos, estaría indicando un cambio en el enfoque de utilización de la red de Algorand. Podría realizarse un estudio en profundidad de este tema, e incluso estudiar el *preferential attachment* sobre activos altamente transaccionados dentro de la red para obtener un mayor conocimiento sobre las preferencias que la red tiene sobre un activo u otro en determinado momento de su historia.

Debido a la constante evolución del sistema, las variables a analizar son tantas como información podamos obtener. Por la transparencia existente en las transacciones realizadas sobre estas nuevas redes *blockchain*, la información se encuentra al alcance de cualquier curioso con suficiente predisposición. Esto es una de las consecuencias del auge de esta nueva tecnología y esperamos que dicha transparencia pueda ser utilizada para obtener un mayor conocimiento del funcionamiento y de las preferencias de los participantes del sistema en el futuro.

Bibliografía

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] Joan Antoni Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartí. The bitcoin P2P network. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 87–102, Berlin, Heidelberg, 2014. Springer.

[3] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 51–68, New York, NY, USA, 2017. Association for Computing Machinery.

[4] Morgen E. Peck. Blockchains: How they work and why they'll change the world. *IEEE Spectrum*, 54(10):26–35, 2017.

[5] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSACConference on Computer and Communications Security*, CCS '16, pages 3–16, New York, NY, USA, 2016. Association for Computing Machinery.

[6] Moritz Platt, Johannes Sedlmeir, Daniel Platt, Jiahua Xu, Paolo Tasca, Nikhil Vadgama, and Juan Ignacio Ibañez. The energy footprint of blockchain consensus mechanisms beyond proof-of-work. In *Proceedings of the IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 1135–1144. Institute of Electrical and Electronics Engineers, Inc, 2021.

[7] Cong T. Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.

[8] Vitalik Buterin. A next-generation smart contract and decentralized application platform. Technical report, Ethereum Foundation, January 2014.

[9] Hongyu Li, Liehuang Zhu, Meng Shen, Feng Gao, Xiaoling Tao, and Sheng Liu. Blockchain-based data preservation system for medical data. *Journal of Medical Systems*, 42(8):141, Jun 2018.

[10] Fabrizio Lamberti, Valentina Gatteschi, Claudio Demartini, Matteo Pelissier, Alfonso Gomez, and Victor Santamaria. Blockchains can work for car insurance: Using smart contracts and sensors to provide on-demand coverage. *IEEE Consumer Electronics Magazine*, 7(4):72–81, 2018.

[11] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.

[12] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *Proceedings of the Annual Symposium on Foundations of Computer Science*, pages 120–130. Institute of Electrical and Electronics Engineers, Inc, 1999.

[13] Haojun Liu, Xinbo Luo, Hongrui Liu, and Xubo Xia. Merkle tree: A fundamental component of blockchains. In *Proceedings of the International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pages 556–561. Institute of Electrical and Electronics Engineers, Inc, 2021.

[14] Mark E. J. Newman. *Random graphs as models of networks*, chapter 2, pages 35–68. John Wiley & Sons, Ltd, 2002.

[15] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016.

[16] Mohammad Dabbagh, Mehdi Sookhak, and Nader Sohrabi Safa. The evolution of blockchain: A bibliometric study. *IEEE Access*, 7:19212–19221, 2019.

[17] Cristian Lepore, Michela Ceria, Andrea Visconti, Udai Pratap Rao, Kaushal Shah, and Luca Zanolini. A survey on blockchain consensus with a performance comparison of pow, pos and pure pos. *Mathematics*, 8:1782, October 2020.

[18] Bilash Saha, Md Mehedi Hasan, Nafisa Anjum, Sharaban Tahora, Aiasha Siddika Arshi, and Hossain Shahriar. Protecting the decentralized future: An exploration of common blockchain attacks and their countermeasures. 07 2023.

[19] I Gusti Ayu Kusdiah Gemeliarana and Riri Fitri Sari. Evaluation of proof of work (pow) blockchains security network on selfish mining. In *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 126–130, 2018.

[20] Saurabh Singh, A. S. M. Sanwar Hosen, and Byungun Yoon. Blockchain security attacks, challenges, and solutions for the future distributed iot network. *IEEE Access*, 9:13938–13959, 2021.

[21] Francesco Collibus, Alberto Partida, Matija Piškorec, and Claudio Tessone. Heterogeneous preferential attachment in key ethereum-based cryptoassets. *Frontiers in Physics*, 9, 10 2021.

[22] Anwar Said, Muhammad Janjua, Saeed-Ul Hassan, Zeeshan Muzammal, Tania Saleem, Tipajin Thaipisutikul, Suppawong Tuarob, and Raheel Nawaz. Detailed analysis of ethereum network on transaction behavior, community structure and link prediction. *PeerJ Computer Science*, 7:e815, 12 2021.

[23] Fengyang Guo, Xun Xiao, Artur Hecker, and Schahram Dustdar. Modeling ledger dynamics in iota blockchain. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 2650–2655, 2022.

[24] Zhen Wang, Rui Zhang, Yipeng Sun, Hong Ding, and Qiuyun Lv. Can lightning network's autopilot function use ba model as the underlying network? *Frontiers in Physics*, 9, 2022.