



UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi Di Laurea

Bitcoin: Riconoscimento di
comportamenti periodici nella Blockchain

Laureando

Claudia Romeo

Matricola 461963

Relatore

Prof. Maurizio Pizzonia

Correlatore

Dott. Valentino Di Donato

Anno Accademico 2016/2017

alla mia famiglia

Ringraziamenti

Indice

| | |
|--|-------------|
| Indice | iv |
| Introduzione | vii |
| Elenco delle figure | viii |
| 1 Bitcoin & Blockchain | 1 |
| 1.1 Cosa sono i Bitcoin? | 1 |
| 1.1.1 Bitcoin: Panoramica | 2 |
| 1.2 Cos'è la Blockchain? | 5 |
| 1.2.1 Mining | 6 |
| 1.2.2 Consenso Decentralizzato | 8 |
| 1.2.3 Blockchain Forks | 9 |
| 1.2.4 Merkle Tree | 15 |
| 1.3 Le Transazioni | 17 |
| 1.3.1 Transaction Mining | 20 |
| 1.3.2 Forma Comune delle Transazioni | 21 |
| 1.3.3 Struttura di una Transazione | 23 |
| 1.3.4 Input e Output delle Transazioni | 24 |
| 1.4 Grafo delle transazioni | 25 |
| 2 I comportamenti automatici | 28 |
| 2.1 Anonimia delle transazioni | 28 |

| | | |
|----------|---|-----------|
| 2.2 | Bitcoin Mixing | 31 |
| 2.2.1 | I Mixer: cosa sono? | 32 |
| 2.2.2 | Mixing nel dettaglio | 33 |
| 2.2.3 | Tor | 35 |
| 3 | Analisi sul grafo delle transazioni | 37 |
| 3.1 | Esempio di grafo reale | 37 |
| 3.2 | Lunghe catene di transazioni | 38 |
| 3.3 | Analisi sul tx-graph | 39 |
| 3.4 | Formalizzazione delle definizioni | 40 |
| 3.4.1 | Frequenza delle transazioni | 40 |
| 3.4.2 | Varianza | 43 |
| 4 | Approccio utilizzato | 45 |
| 4.1 | Networkx per la creazione di grafi | 45 |
| 4.2 | Dataset | 46 |
| 4.2.1 | Struttura dati di un blocco | 46 |
| 4.3 | Cammino più lungo a varianza minima | 49 |
| 4.3.1 | Creazione grafo delle transazioni | 49 |
| 4.3.2 | Assegnamento timestamp alle transazioni | 52 |
| 4.3.3 | Dettagli dell'algoritmo per il calcolo del cammino più lungo a varianza minima | 57 |
| 5 | Sperimentazione & Risultati | 61 |
| 5.1 | Grafi di test | 61 |
| 5.2 | MatPlotLib per i grafici | 62 |
| 5.3 | Statistiche & Grafici | 63 |
| 5.3.1 | Al variare del Dataset | 64 |
| | Conclusioni e sviluppi futuri | 68 |

Introduzione

Negli ultimi anni, con il progresso tecnologico e con l'aumentare del coinvolgimento di Internet nella nostra quotidianità, si è arrivati a digitalizzare anche il denaro. Grazie a questa digitalizzazione, sono state create delle monete virtuali, o valute elettroniche, che possono essere utilizzate solo su Internet. La moneta elettronica che ha lanciato questa "tendenza" è una delle più famose, il Bitcoin.

Elenco delle figure

| | | |
|------|---|----|
| 1.1 | <i>Bitcoin overview</i> | 3 |
| 1.2 | <i>Payment request QR code</i> | 4 |
| 1.3 | <i>Visualizzazione di un evento fork–prima della fork</i> | 10 |
| 1.4 | <i>Visualizzazione di un evento fork: due blocchi minati simultaneamente</i> | 11 |
| 1.5 | <i>Visualizzazione di un evento fork: due blocchi propagati, la rete biforca</i> | 12 |
| 1.6 | <i>Visualizzazione di un evento fork: un nuovo blocco estende un ramo della fork</i> | 13 |
| 1.7 | <i>Visualizzazione di un evento fork: la rete si riunisce in un'unica catena più lunga</i> | 14 |
| 1.8 | <i>Merkle tree</i> | 15 |
| 1.9 | <i>Merkle tree con elemento duplicato</i> | 17 |
| 1.10 | <i>Una transazione come una riga del registro di contabilità</i> | 18 |
| 1.11 | <i>Una catena di transazioni, dove l'output di una transazione viene utilizzato come input della transazione successiva</i> | 19 |
| 1.12 | <i>La transazione di Alice inclusa nel blocco #277316</i> | 21 |
| 1.13 | <i>La transazione di Alice come parte di una catena di transazioni che va da Joe a Gopesh</i> | 22 |
| 1.14 | <i>Una comune transazione</i> | 22 |
| 1.15 | <i>Aggregazione</i> | 22 |
| 1.16 | <i>Distribuzione</i> | 23 |
| 1.17 | <i>Grafo delle transazioni</i> | 25 |

| | |
|---|----|
| 1.18 <i>Un esempio di tx-graph con 3 tx(1,2 e 3)</i> | 27 |
| 2.1 <i>Una transazione del sito Blockchain.info [Blo]</i> | 29 |
| 2.2 <i>Dettagli degli input/output</i> | 30 |
| 2.3 <i>Il mixing di bitcoin [Mix]</i> | 32 |
| 2.4 <i>Un servizio di mixing, che nasconde la relazione tra Alice e Bob</i> | 34 |
| 3.1 <i>Esempio di tx-graph reale</i> | 37 |
| 3.2 <i>Zoom di una chain</i> | 41 |
| 4.1 <i>Algoritmo assegnazione LLC [DDP17]</i> | 56 |
| 5.1 <i>Un cammino</i> | 61 |
| 5.2 <i>Un Y grafo</i> | 62 |
| 5.3 <i>Un X grafo</i> | 62 |
| 5.4 <i>G1(417113, 417256)</i> | 63 |
| 5.5 <i>20 blocchi</i> | 64 |
| 5.6 <i>50 blocchi</i> | 65 |
| 5.7 <i>100 blocchi</i> | 65 |
| 5.8 <i>150 blocchi</i> | 66 |
| 5.9 <i>200 blocchi</i> | 66 |
| 5.10 <i>250 blocchi</i> | 67 |
| 5.11 <i>1000 blocchi</i> | 67 |

Capitolo 1

Bitcoin & Blockchain

1.1 Cosa sono i Bitcoin?



Bitcoin è una valuta elettronica creata nel 2008 da Satoshi Nakamoto, uno pseudonimo dietro al quale non si sa ancora con la precisione chi si nasconde. Con il termine Bitcoin viene denotata sia la rete che consente il possesso e il trasferimento di denaro, sia la moneta. Per convenzione, Bitcoin si riferisce alla tecnologia della rete, mentre *bitcoin* alla valuta stessa.[Wik11a]

Come ogni valuta, i bitcoin possono essere trasferiti tramite gli utenti, grazie ad un protocollo che viene rispettato all'interno della rete Internet, il quale può essere eseguito su differenti dispositivi, in modo tale da permettere la fruibilità del servizio anche attraverso gli smartphones.

I bitcoins possono essere comprati, venduti e scambiati con altre valute, tramite degli organismi specializzati nel cambio di monete virtuali, chiamati *exchange*. In un certo senso, Bitcoin è la forma perfetta di denaro per Internet, dal momento che è estremamente veloce, sicuro e senza limiti.

A differenza delle altre valute, i bitcoin sono esclusivamente virtuali, dietro di essi non esistono monete fisiche. Tali bitcoin vengono coinvolti in transazioni

da mittente a ricevente, i quali possiedono delle chiavi crittografiche pubbliche e private che servono per trasmettere e sbloccare la spesa dei bitcoin ricevuti. Infatti, senza la chiave privata, chi riceve i bitcoin non può spenderli in nessun modo. Tali chiavi vengono conservate all'interno di un *wallet*, letteralmente un "portafoglio". Ogni wallet è caratterizzato da un indirizzo Bitcoin il quale è univoco e ha la funzione di fare riferimento ad uno dei partecipanti alla transazione. In questo modo, quando viene effettuato uno scambio di bitcoin, vengono visualizzati solamente gli indirizzi dei wallet. Questa caratteristica permette quindi di rendere anonime le transazioni, dato che agli indirizzi non è connesso in nessun modo il nome o il cognome dell'individuo o dell'associazione che interviene nello scambio.

La rete Bitcoin, oltre ad essere completamente virtuale, è priva di un'unità centralizzata, infatti essa è costituita da un sistema distribuito peer-to-peer.

I bitcoin vengono creati tramite un processo, detto *mining*, che permette a chiunque di mettersi in competizione per trovare una soluzione ad un problema matematico. Ogni persona che partecipa alla rete bitcoin, potrebbe operare come un *miner*, ovvero colui che cerca di risolvere il problema matematico per generare bitcoin, usando le capacità del proprio computer messo a disposizione della computazione.[Ant14]

Quando viene effettuato uno scambio di bitcoin tra due o più wallet, viene creata una *transazione*. Ogni transazione viene conservata in una struttura dati chiamata **blocco**, il quale a sua volta va a costituire la **Blockchain**.

1.1.1 Bitcoin: Panoramica

Nella *figura 1.1*, si può notare che il sistema bitcoin comprende: utenti con wallet che contengono chiavi, transazioni che sono propagate attraverso la rete, e miner che forniscono (attraverso una computazione competitiva) il consenso alla blockchain, la quale è il libro mastro di tutte le transazioni.

In questo paragrafo si andrà a tracciare il percorso di una singola transazione quando viene propagata attraverso la rete, e le interazioni tra ogni componente del sistema, ad alto livello.

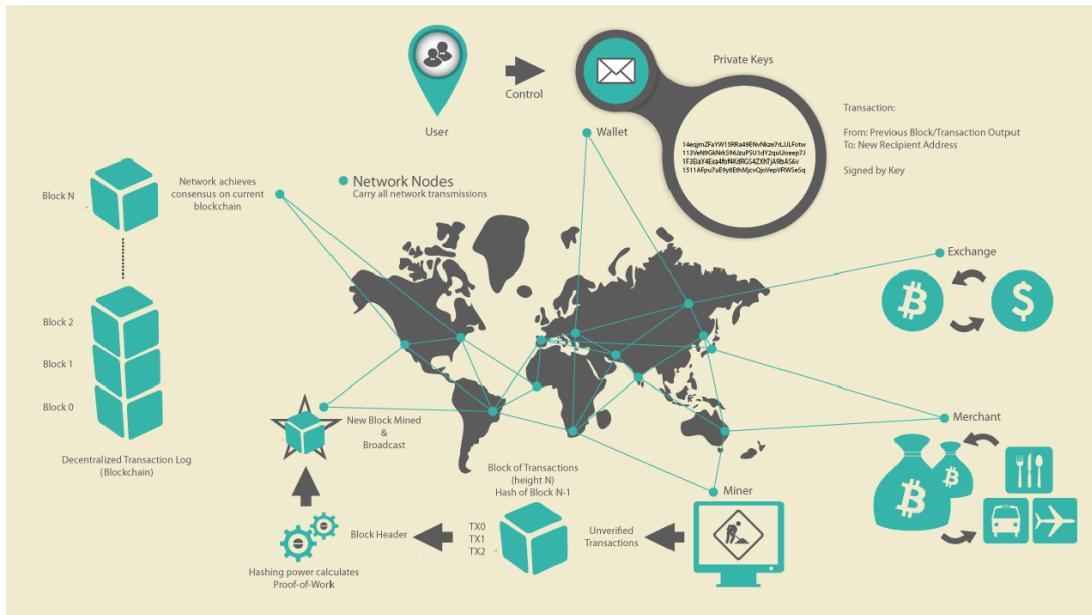


Figura 1.1: *Bitcoin overview*

Si prendano in considerazione per esempio, due utenti Bitcoin: Alice e Bob. Alice vorrebbe effettuare la sua transazione, comprando una tazza di caffè al bar di Bob (Bob's Cafe). Il bar di Bob, di recente ha iniziato ad accettare pagamenti in bitcoin, aggiungendo alla cassa un punto di pagamento Bitcoin. I prezzi del bar sono elencati in dollari, ma alla cassa, i clienti hanno l'opzione di pagare sia in dollari che in bitcoin.

Alice fa il suo ordine di una tazza di caffè e Bob inserisce la transazione nel registratore di cassa. Il punto di scambio di bitcoin, converte il prezzo totale dai dollari ai bitcoin secondo il tasso di mercato corrente, e mostra il prezzo nelle due valute, insieme ad un QR code contenente la *payment request* (richiesta di pagamento) per tale transazione (*figura 1.2*).

Se tale QR code viene scansionato, esso racchiude il seguente URL:



Figura 1.2: *Payment request QR code*

```
1     bitcoin:1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA?
2     amount=0.015&
3     label=Bob%27s%20Cafe&
4     message=Purchase%20at%20Bob%27s%20Cafe
```

Attraverso di esso si possono ottenere informazioni come: l'indirizzo bitcoin, l'ammontare del pagamento, un'etichetta per l'indirizzo del destinatario e una descrizione del pagamento.

A differenza di un QR code che contiene semplicemente l'indirizzo bitcoin del destinatario, un *payment request* è un QR-encoded URL che contiene un indirizzo di destinazione, il pagamento, e una descrizione generica come "Bob's Cafe". Questo permette all'applicazione del wallet di raccogliere le informazioni usate per effettuare il pagamento, al fine di mostrare successivamente quando richieste dal proprietario del wallet. Se Bob scansiona il QR code con un'applicazione per un wallet bitcoin, oltre a ottenere il denaro, vedrà i dettagli inseriti da Alice.

Quindi, Bob dice: "Sono un dollaro e 50, oppure 15 millibitcoin"

Alice usa il suo smartphone per scansionare il codice e visualizzarlo sul display. Esso le mostra il pagamento di 0.015 BTC al Bob's Cafe e seleziona *Invia* per autorizzare il pagamento. In pochi secondi (all'incirca lo stesso tempo che ci mette una comune carta di credito), Bob potrà vedere la transazione sul registratore di cassa, conclusa.[Ant14]

1.2 Cos'è la Blockchain?

La Blockchain, letteralmente *catena di blocchi*, è una base di dati distribuita, che permette la memorizzazione delle transazioni raggruppate in blocchi connessi tra loro, ognuno con il suo successivo. Ogni blocco è una struttura dati che contiene un numero variabile di transazioni, inserite dal *miner* che ha minato il blocco, fino ad un tetto massimo di 1MB per singolo blocco.

L'idea di base della Blockchain, deriva dal concetto di **libro mastro**, il registro della contabilità in cui sono riuniti tutti i conti che compongono un dato sistema contabile. In questo caso il sistema contabile sarebbe la rete Bitcoin, mentre la Blockchain sarebbe il libro mastro che contiene tutti i conti, ovvero le transazioni.

L'antico libro mastro, veniva utilizzato come fonte ufficiale per la memorizzazione degli scambi e dei passaggi di proprietà. Infatti, quando veniva fatta una compravendita tra mittente e ricevente, veniva controllato sul libro mastro se il ricevente non avesse speso precedentemente il denaro, e se il mittente non avesse già venduto la merce usata nello scambio.

Infine, se fosse andato tutto a buon fine, veniva registrata la transazione sul libro mastro, in modo da essere consultabile e pubblica per le successive transazioni.

Un principio importante di questo meccanismo è la fiducia, la quale tutti ripongono nel libro mastro: ognuno si fida del gestore della memorizzazione delle transazioni, al punto che, chi compra e chi vende, può effettuare scambi anche senza fidarsi reciprocamente. Quindi, il libro mastro è una garanzia, sia per il mittente che per il ricevente dello scambio. Inoltre, le banche possono perciò controllare gli scambi che vengono fatti e il denaro posseduto da ogni partecipante alle transazioni.

La Blockchain, come già sottolineato, è una struttura dati composta da diverse

unità di base, dette blocchi. Infatti, blockchain significa letteralmente *catena di blocchi*.⁶

Quindi, tali blocchi vengono "incatenati", ovvero collegati tra loro tramite un protocollo ben definito nella struttura del sistema bitcoin.

Ogni blocco all'interno della blockchain, è identificato da un codice hash, generato applicando l'algoritmo di crittografia SHA256 all'header del blocco. Un singolo blocco è collegato al suo predecessore, conosciuto come "blocco genitore", attraverso il campo *previous block hash* all'interno del proprio header. In altre parole, ogni blocco contiene l'hash del proprio blocco genitore all'interno dell'header. Infine, la sequenza dei vari hash genera una catena che collega tutti i blocchi all'indietro, fino al blocco numero zero.[Ant14]

1.2.1 Mining

Le monete bitcoin sono "coniate" durante la creazione di ciascun blocco ad un tasso fisso. Ogni blocco, generato all'incirca ogni 10 minuti, contiene nuovi bitcoin, creati da zero. Il *mining* inoltre serve per proteggere il sistema bitcoin contro transazioni fraudolente o transazioni che cercano di spendere gli stessi bitcoin più di una volta, problema conosciuto con il nome di *double-spend* (doppia-spesa).

I miner validano le transazioni e le registrano sul *ledger* globale, ovvero la Blockchain, creando un nuovo blocco e inserendo le transazioni al suo interno, per poi aggiungere il blocco alla chain (=catena). Perciò, i miner raccolgono un certo numero di transazioni e cercano di inserirle all'interno di un nuovo blocco creato appositamente. Tutte le transazioni che poi risultano alla fine all'interno di tale blocco, sono considerate *confirmed* (=confermate), ovvero una sorta di etichetta che permette al ricevente di tali bitcoin di spenderli successivamente.

Lo scopo dei miner è quello di guadagnare bitcoin, e possono ottenerli in due modi:

- quando un nuovo blocco viene aggiunto alla Blockchain, si ottiene un premio in bitcoin
- per ogni transazione vengono pagate le *fees* (=tasse) al miner

Al fine di ottenere il premio, i miner devono trovare la soluzione ad un problema matematico molto difficile basato su un algoritmo di crittografia. La soluzione a tale problema, chiamata **proof of work**, è inclusa all'interno del nuovo blocco, e serve come prova che il miner ha impiegato un notevole sforzo di elaborazione. La competizione tra i miner per trovare la proof of work per guadagnare bitcoin è alla base della sicurezza del sistema Bitcoin.

Il processo della generazione di nuove monete è chiamato **mining** perchè la ricompensa è progettata in modo da simulare rendimenti decrescenti, esattamente come l'estrazione di metallo prezioso. La fornitura di valuta bitcoin è creata attraverso il mining, analogamente alla procedura con cui una banca centrale crea moneta stampando banconote.

L'ammontare del premio del mining a Gennaio 2009 era di 50 bitcoin per blocco e a Novembre 2012 già si era dimezzato fino ad ottenere 25 bitcoin. Attualmente il guadagno del premio del mining risulta 12.5 bitcoin per blocco. Seguendo questo processo, il premio per il mining di un blocco, decrescerà esponenzialmente fino all'anno 2140, quando tutti i bitcoin (20.9999998 milioni) saranno emessi. All'incirca dopo il 2140 non saranno più prodotti nuovi bitcoin e il miner andrà a guadagnare solamente tramite le tasse applicate per ogni transazione.

I miners guadagnano bitcoin per ogni transazione tramite le tasse. Esse vengono calcolate come eccesso di bitcoin tra le transazioni di input e quelle di output. Infatti, il miner che riesce a creare il nuovo blocco "tiene il resto" di ogni transazione inclusa in quel blocco.

Dopo il 2140, tutti i bitcoin potranno essere guadagnati tramite tasse.

La parola "mining", ovvero minare, potrebbe avere un significato ingannevole poichè sembrerebbe indicare l'estrazione di metallo prezioso, e quindi focalizza

l'attenzione sul premio che viene ottenuto dal minatore.

Sebbene il mining sia incentivato dal suo guadagno, il suo obiettivo principale non è il premio in denaro o la creazione di nuove monete.

Mining è il processo principale per la decentralizzazione del sistema, nel quale le transazioni sono validate e chiare. Rappresenta una sicurezza per il sistema e permette lo sviluppo di una rete basata sul consenso, senza l'intervento di un'autorità centrale.[Ant14]

Quando un blocco viene minato, ovvero viene aggiunto alla blockchain, viene etichettato con un timestamp, che rappresenta l'istante in cui viene ricevuto dal nodo bitcoin che ha la copia della blockchain.

1.2.2 Consenso Decentralizzato

La Blockchain non essendo creata da un'autorità centrale, è assemblata indipendentemente da ogni nodo all'interno della rete. Perciò, ogni nodo della rete, agendo sulle informazioni che vengono trasmesse attraverso delle connessioni di rete non sicure, può arrivare alla stessa conclusione e assemblare una copia della stessa blockchain, come ogni altro nodo.

La principale invenzione di Satoshi Nakamoto è il meccanismo decentralizzato del *consenso emergente*. "Emergente" perchè il consenso generale non è raggiunto esplicitamente – non esiste un'elezione o un momento prefissato quando il consenso viene espresso – è un risultato dell'interazione asincrona di migliaia di nodi indipendenti tra loro, i quali basati sulle stesse regole.

Tutte le proprietà dei bitcoin, inclusa la moneta, le transazioni, i pagamenti, il modello di sicurezza, i quali non dipendono da un'autorità centrale, derivano da questa invenzione.

Il consenso decentralizzato dei Bitcoin viene fuori dall'interazione di quattro processi che si verificano indipendentemente sui nodi attraverso la rete. Tali processi sono:

- la verifica indipendente di ogni transazione, basata su una lista globale di criteri
- l’aggregazione indipendente di tali transazioni all’interno di un nuovo blocco appena minato
- la verifica indipendente del nuovo blocco da parte di ogni nodo, e l’unione di tale blocco all’interno della chain
- la selezione indipendente, da parte di ogni nodo, della chain con la computazione che maggiormente raccoglie più transazioni e che è stata dimostrata correttamente tramite una *proof of work*

[Ant14]

1.2.3 Blockchain Forks

Dal momento che la Blockchain è una struttura dati decentralizzata, copie differenti non sono sempre consistenti. Infatti, i blocchi possono arrivare a nodi diversi in tempi diversi, creando differenti rami all’interno della stessa blockchain.

Sebbene un blocco abbia solo un genitore, esso può avere temporaneamente un diverso numero di figli. Ogni figlio si riferisce allo stesso blocco genitore e contiene lo stesso hash del genitore nel campo *previous block hash*. Questa temporanea molteplicità di figli può causare una *fork* (letteralmente *biforcazione*), ovvero una situazione in cui blocchi differenti vengono creati quasi simultaneamente da diversi miners.

Per risolvere il problema della biforcazione, ogni nodo Bitcoin seleziona sempre e cerca di estendere la catena di blocchi che soddisfa il sistema *proof-of-work*.

Le fork avvengono come risultato di inconsistenze temporanee tra versioni della blockchain, che vengono risolte con eventuali riconvergenze, ovvero i nuovi blocchi vengono aggiunti alla catena principale della biforcazione.

Nelle figure seguenti, si può capire un esempio di una fork della blockchain. Sebbene nelle figure si nota la rete globale, in realtà la topologia della rete bitcoin non è organizzata geograficamente, piuttosto forma una rete di nodi interconnessi, che potrebbero essere geograficamente molto lontani. Nella vera rete bitcoin, la *distanza* tra i nodi è misurata in **hop** da nodo a nodo, non sulla loro distanza fisica. Un singolo hop è una porzione di percorso tra la sorgente e la destinazione.

Per scopi illustrativi, blocchi differenti sono mostrati con colori diversi, che si diffondono attraverso la rete e le connessioni attraversate vengono colorate con colori diversi. Nel primo diagramma (*figura 1.3*), la rete appare con una singola prospettiva unificata della blockchain, con il blocco blu come estremità della catena principale.

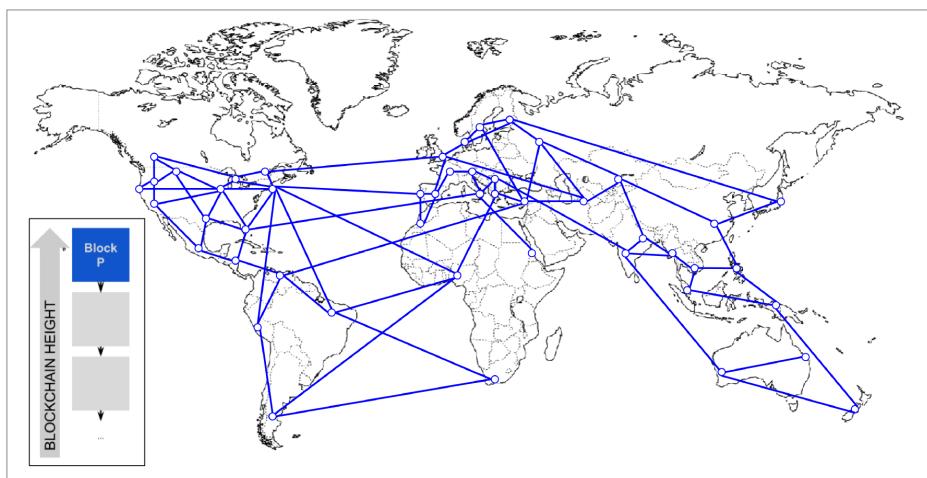


Figura 1.3: *Visualizzazione di un evento fork-prima della fork*

Un evento *fork* avviene quando ci sono due diversi blocchi che sono candidati ad essere aggiunti contemporaneamente alla Blockchain. Ciò può accadere quando due diversi miner risolvono la proof of work con uno scarto di tempo molto piccolo tra entrambi. Di conseguenza, tali miner immediatamente rivelano la loro soluzione in modo broadcast, e il nuovo blocco creato viene trasmesso tempestivamente ai loro vicini, che propagano i blocchi nuovi attraverso la rete.

Ogni nodo che riceve un blocco valido, lo incorpora all'interno della propria versione della blockchain, allungandola di un blocco. Se tale nodo si rende conto di aver ricevuto un ulteriore blocco che soddisfa le condizioni, lo aggiunge creando una catena secondaria alternativa al blocco aggiunto appena prima.

In *figura 1.4* si possono notare due miner che hanno minato due diversi blocchi quasi simultaneamente.

Entrambi i blocchi sono figli del blocco blu, ed hanno lo scopo di estendere la chain aggiungendosi sopra al blocco blu. Nella figura, un blocco è rappresentato in rosso e l'altro in verde.

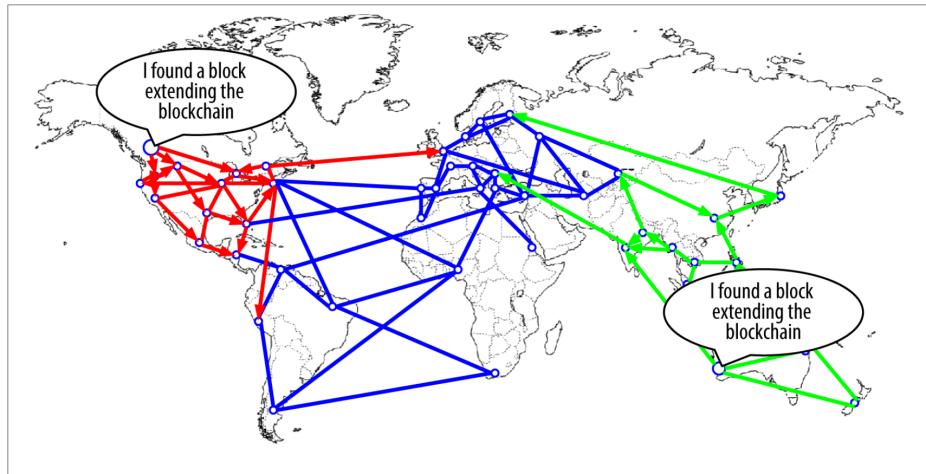


Figura 1.4: Visualizzazione di un evento fork: due blocchi minati simultaneamente

Per esempio, si assuma che un miner in Canada trova una soluzione proof of work per il blocco "rosso" che estende la blockchain come figlio del blocco "blu". Quasi simultaneamente, un altro miner in Australia trova un'altra soluzione per il blocco "verde" al fine di estendere la blockchain. Entrambi i blocchi sono validi, entrambi contengono una soluzione valida alla proof of work, e tutti e due sono figli del blocco "blu". Inoltre, entrambi contengono quasi le stesse transazioni, con solo alcune piccole differenze.

Appena i due blocchi vengono propagati, alcuni nodi ricevono il blocco "rosso" e altri il blocco "verde". Come si può vedere in *figura 1.5*, la rete si divide in due diverse prospettive della blockchain, un lato con all'estremità il blocco rosso e l'altro lato con il blocco verde.

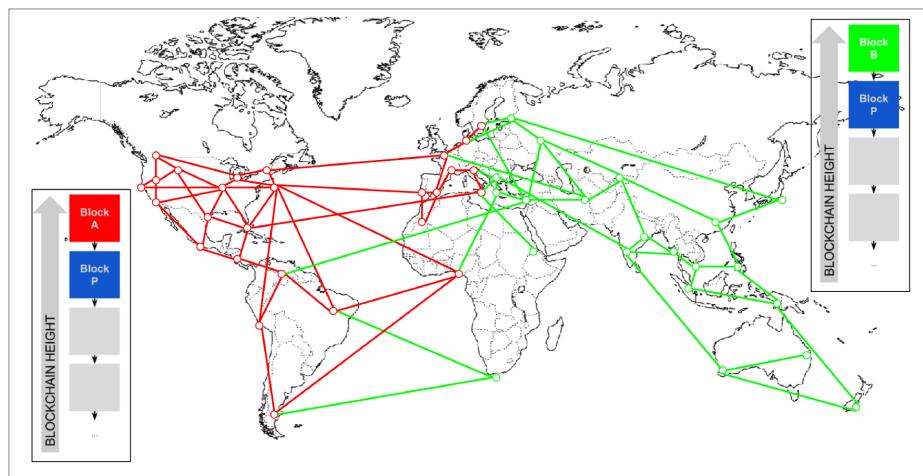


Figura 1.5: *Visualizzazione di un evento fork: due blocchi propagati, la rete biforca*

Da quel momento, i nodi della rete che sono più vicini al nodo del Canada, aggiungeranno anch'essi il blocco "rosso" per primo, e creeranno una nuova blockchain con il blocco "rosso" come ultimo blocco, ignorando il blocco "verde", che è arrivato un po' più tardi. Nello stesso momento, i nodi più vicini al nodo dell'Australia prenderà il blocco "verde" come vincitore e lo userà per estendere la sua versione della blockchain, aggiungendolo al blocco "blu", ignorando il blocco "rosso" che è arrivato un po' più tardi degli altri. Di conseguenza, ogni miner che vede aggiungere il blocco "rosso" in testa alla chain, immediatamente cercherà di creare altri blocchi che si aggiungeranno al blocco "rosso", ed andrà a risolvere la proof of work per tali blocchi candidati. Invece, i miners che accettano il blocco "verde" cominceranno ad estendere la porzione di chain che si andrà ad attaccare a tale blocco.

Le fork vengono quasi sempre risolte da un singolo blocco. Infatti, come parte del potere computazionale viene dedicato per aggiungere il blocco "rosso", un'altra parte della rete impiega le sue risorse per aggiungere il blocco "verde". Anche se il potere computazionale è quasi diviso in due parti, probabilmente un gruppo di miner troverà e propagherà la soluzione prima che lo faccia un altro gruppo di ulteriori miner. Per esempio, si supponga che i miner trovino un blocco "rosa" che estenda il blocco "verde", immediatamente lo propagherebbero all'intera rete (*figura 1.6*).

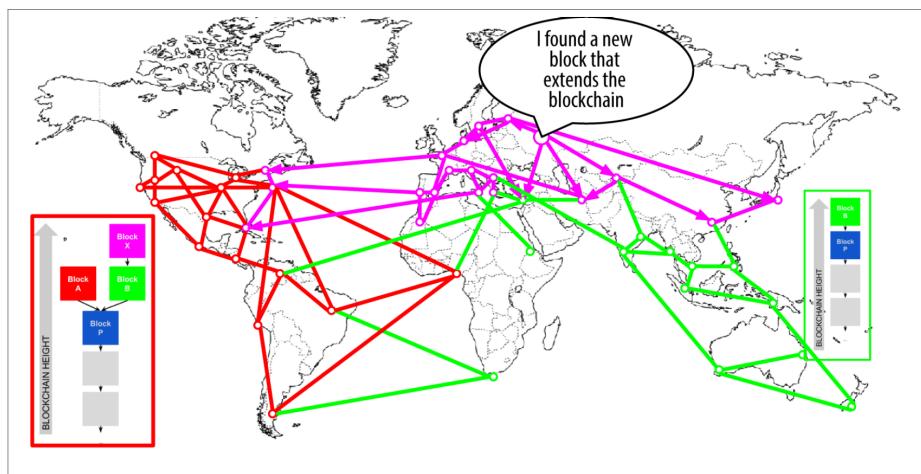


Figura 1.6: *Visualizzazione di un evento fork: un nuovo blocco estende un ramo della fork*

Tutti i nodi che hanno scelto il blocco "verde" come vincitore nel round precedente, estenderanno la catena di un ulteriore blocco. I nodi che hanno scelto il blocco "rosso" come vincitore, quindi, vedranno due chain: quella blu-verde-rosa e quella blu-rossa. La chain blu-verde-rosa è la più lunga, ovvero è più difficile trovare una proof of work per essa. Inoltre, in *figura 1.7* si possono vedere i nodi che hanno scelto come catena principale la chain blu-verde-rosa e come chain secondaria quella blu-rossa.

Questo processo è detto *chain reconvvergence* (letteralmente riconvergenza della catena), perché tali nodi vengono forzati a cambiare il loro punto di vista della

blockchain al fine di considerare la catena più lunga tra le due.

Ogni miner che lavora per estendere la chain blu-rossa non potrà più continuare poichè il blocco che stanno tentando di aggiungere al blocco rosso rimarrà "orfano", dato che il blocco rosso che dovrebbe fare da genitore non appartiene alla chain più lunga della blockchain.

Le transazioni che sono all'interno del blocco "rosso", vengono inserite di nuovo in coda per essere processate in un nuovo blocco, perchè, come si è già detto, il blocco a cui appartenevano non fa più parte della blockchain.

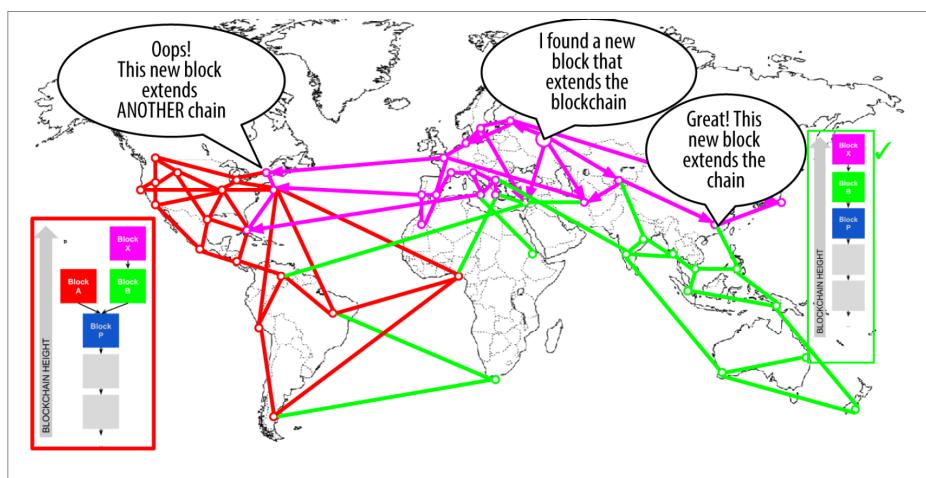


Figura 1.7: *Visualizzazione di un evento fork: la rete si riunisce in un'unica catena più lunga*

L'intera rete, quindi, riconverge la blockchain in una catena singola che fa capo ai blocchi blu-verde-rosa, con il blocco "rosa" come ultimo blocco all'estremità della catena. Tutti i miner immediatamente cominceranno a candidare nuovi blocchi che si andranno ad attaccare al blocco "rosa", al fine di estendere la catena blu-verde-rosa.

Teoricamente potrebbe essere possibile per una fork estendere due blocchi diversi, ma solamente se tali blocchi vengono creati quasi simultaneamente da miner che sono su "lati" diversi di una fork precedente. Tuttavia, la probabilità

che ciò accada è molto bassa. Anche se ci fosse una fork ogni settimana, una fork a due blocchi è molto rara.

L’intervallo tra un blocco e il suo successivo è all’incirca di 10 minuti, proprio perché è stato progettato per essere un compromesso tra la velocità dei tempi di ricezione delle conferme e la probabilità di una fork. Un intervallo più ristretto da un lato potrebbe rendere le transazioni più veloci, dall’altro potrebbe causare fork più frequenti. Al contrario, un intervallo più ampio farebbe diminuire il numero delle fork ma renderebbe più lenti i pagamenti.[Ant14]

1.2.4 Merkle Tree

Ogni blocco della blockchain contiene un riepilogo di tutte le transazioni nel blocco, sotto forma di un **merkle tree**.

Un *merkle tree*, altrimenti detto *binary hash tree*, è una struttura dati usata per verificare e riepilogare efficientemente l’integrità dei dati. I merkle tree sono alberi binari composti da hash crittografici, dove ogni transazione è convertita in un hash, usando una funzione crittografica.

Il merkle tree è un albero perfettamente bilanciato, dove le foglie corrispondono agli hash delle singole transazioni. I nodi intermedi invece, sono calcolati applicando la funzione di hash alla concatenazione dei valori dei due figli (*figura 1.8*).

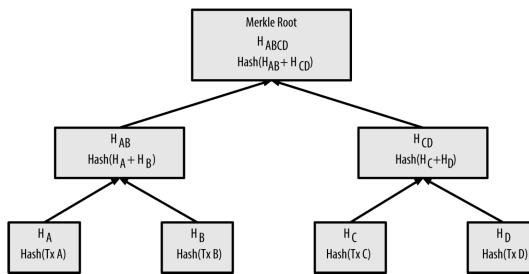


Figura 1.8: *Merkle tree*

Nella figura si può notare una porzione di un albero merkle, che prende in considerazione quattro transazioni: Tx A, Tx B, Tx C e Tx D. L'albero in questione presenta delle caratteristiche diverse se il nodo è una foglia oppure è un nodo intermedio. Esso è costruito secondo l'approccio bottom-up, prima vengono calcolati gli hash delle foglie, poi man mano si arriva al calcolo della radice.

Le foglie dell'albero, vengono calcolate come hash delle quattro transazioni, quindi si avranno le quattro foglie rispettivamente:

- $H_A = \text{Hash}(\text{Tx A})$
- $H_B = \text{Hash}(\text{Tx B})$
- $H_C = \text{Hash}(\text{Tx C})$
- $H_D = \text{Hash}(\text{Tx D})$

Successivamente si calcola il valore dei genitori delle foglie. Infatti, ogni nodo tra di essi, viene calcolato facendo l'hash della concatenazione dei valori contenuti nelle foglie. Quindi per ogni nodo intermedio si avrà:

- $H_{AB} = \text{Hash}(H_A + H_B)$ ¹
- $H_{CD} = \text{Hash}(H_C + H_D)$

Infine, la radice si ottiene facendo lo stesso procedimento dei nodi intermedi, ossia facendo la concatenazione dei valori contenuti nei nodi figli. In questo caso si otterrà:

$$H_{ABCD} = \text{Hash}(H_{AB} + H_{CD})$$

Nell'esempio appena visto, l'albero era un albero binario, dove il numero di transazioni risultava essere un numero pari. Se ci fosse un numero dispari di transazioni da considerare nel merkle tree, l'ultima transazione verrebbe duplicata ottenendo così un numero pari di transazioni. Si otterrà quindi un albero

¹In questo caso il simbolo + rappresenta la concatenazione

bilanciato. In figura 1.9 viene mostrata la duplicazione della transazione Tx C, e di conseguenza della foglia a cui appartiene. [Ant14]

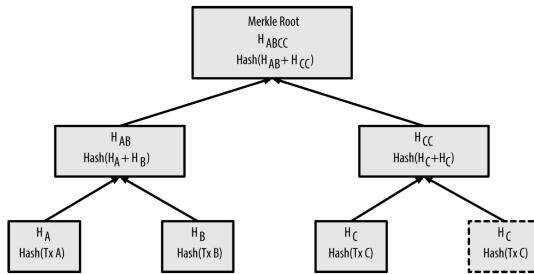


Figura 1.9: *Merkle tree con elemento duplicato*

1.3 Le Transazioni

Come è stato detto in precedenza, una transazione è uno scambio di monete bitcoin tra due o più individui. Per esempio, se Alice vuole dare 1BTC a Bob, ha bisogno di effettuare una transazione inserendo come input l'importo da trasferire e l'indirizzo del wallet di Bob.

In parole semplici, la transazione dice alla rete che il proprietario di un certo numero di bitcoin, ha autorizzato il trasferimento di alcuni di tali bitcoin ad un altro individuo. Il nuovo proprietario dei bitcoin allora può spenderli creando una nuova transazione che autorizza il trasferimento degli stessi ad un altro destinatario. Tutto ciò genera una catena di proprietà di bitcoin.

Le transazioni sono come una riga in un registro di contabilità dove vengono salvati i trasferimenti. In poche parole, ogni transazione contiene uno o più input, cioè l'addebito, ed uno o più output, cioè il credito aggiunto ad un altro account bitcoin. Gli input e gli output, rispettivamente l'addebito e il credito, non necessariamente devono essere la stessa cifra. Infatti, gli output aggiungono all'account destinatario una cifra leggermente minore di quella inviata negli input, la cui differenza rappresenta le *transaction fee* (ovvero le tasse di transazione), che

vengono elargite al miner che ha minato il blocco nel quale andrà ad aggiungersi la transazione in corso. Nella *figura 1.10* viene mostrata una transazione come sarebbe all'interno di un registro di contabilità.

| Transaction as Double-Entry Bookkeeping | | | |
|---|---|----------------|----------|
| Inputs | Value | Outputs | Value |
| Input 1 | 0.10 BTC | Output 1 | 0.10 BTC |
| Input 2 | 0.20 BTC | Output 2 | 0.20 BTC |
| Input 3 | 0.10 BTC | Output 3 | 0.20 BTC |
| Input 4 | 0.15 BTC | | |
| Total Inputs: | 0.55 BTC | Total Outputs: | 0.50 BTC |
| - | | | |
| <i>Inputs</i> | <i>0.55 BTC</i> | | |
| <i>Outputs</i> | <i>0.50 BTC</i> | | |
| <i>Difference</i> | <i>0.05 BTC (implied transaction fee)</i> | | |

Figura 1.10: *Una transazione come una riga del registro di contabilità*

La transazione contiene anche la firma digitale del destinatario dell'importo dei bitcoin, in modo da fungere come prova di proprietà dal momento che nessun altro all'infuori del proprietario può validare la transazione e prendersi gli stessi bitcoin. In termini tecnici, "spendere" significa firmare una transazione che trasferisce il valore da una transazione effettuata in precedenza ad un nuovo proprietario identificato tramite il suo indirizzo bitcoin.

Le transazioni muovono ammontare di bitcoin da *transazioni in input* a *transazioni in output*. Un input è dove la moneta arriva da un output di una transazione effettuata in precedenza. Un output assegna un nuovo proprietario al valore scambiato, associandogli la chiave. La chiave di destinazione è chiamata *encumbrance*(letteralmente "impedimento"). Esso impone l'esigenza di avere una firma che serve a riscattare i fondi in transazioni future. Gli output di una transazione possono essere utilizzati come input in una nuova transazione, creando così una

catena di proprietà che rappresenta il valore che si muove da indirizzo a indirizzo (*figura 1.11*).

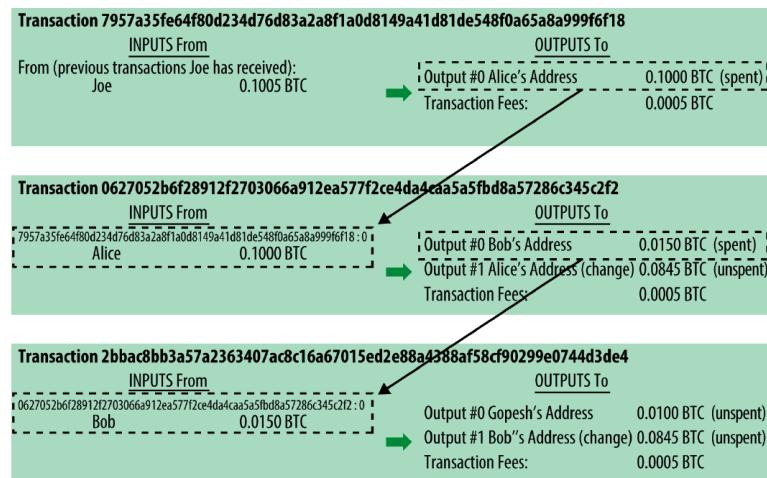


Figura 1.11: *Una catena di transazioni, dove l'output di una transazione viene utilizzato come input della transazione successiva*

Il pagamento di Alice al Bob's Cafe usa come input l'output di una precedente transazione. Infatti, prima di trasferire i suoi bitcoin a Bob, Alice li aveva ricevuti precedentemente da un suo amico Joe, che glieli aveva donati in cambio di una somma di denaro. Tale transazione ha un numero di bitcoin bloccati, che possono essere liberati solamente dalla chiave di Alice. La sua nuova transazione per pagare la tazza di caffè al bar di Bob fa riferimento come input alla transazione precedente e crea nuovi output grazie al pagamento.

Le transazioni formano una catena, dove gli input dell'ultima transazione corrispondono ad output della transazione precedente. La chiave di Alice ha la funzione di firma che sblocca gli output della transazione precedente, proprio per provare alla rete bitcoin che Alice possiede tali monete. Lei attribuisce il pagamento all'indirizzo di Bob, in tal modo bloccando l'output che poi successivamente sarà sbloccato dalla chiave di Bob. Tutto ciò rappresenta il trasferimento di bitcoin tra Alice e Bob. La catena di transazioni che parte da Joe e arriva a Bob è illustrata in *figura 1.11*.[Ant14]

1.3.1 Transaction Mining

Quando viene fatta l'operazione di mining dei blocchi, vengono inserite le transazioni all'interno del nuovo blocco. In questa sezione viene mostrato il mining dal punto di vista della transazione.

In media ogni 10 minuti, i miner generano un nuovo blocco che contiene tutte le transazioni che sono state generate dopo l'ultimo blocco inserito. Le nuove transazioni scorrono all'interno della rete dai wallet degli utenti e da altre applicazioni. Appena le transazioni incontrano la rete di nodi bitcoin, vengono aggiunte ad una struttura dati temporanea presente in ogni nodo dove vengono conservate tutte le transazioni che non sono ancora state verificate. Appena i miner generano un nuovo blocco, aggiungono le transazioni non verificate da tale struttura dati, al nuovo blocco e tentano quindi di trovare la proof of work.

Prendendo in considerazione l'esempio precedente, la transazione di Alice quindi, viene prelevata dalla rete e inclusa all'interno della struttura dati temporanea. Essa viene aggiunta ad un blocco che accetta l'importo delle tasse pagate da Alice. All'incirca cinque minuti più tardi che la transazione è stata trasmessa dal wallet di Alice, un miner riesce a minare il blocco e pubblica il blocco contenente un certo numero di transazioni compresa quella di Alice. Il miner in questione pubblica il nuovo blocco sulla rete bitcoin, dove gli altri miner lo validano e possono continuare la corsa a trovare una nuova proof of work.

Pochi minuti più tardi, un nuovo blocco viene minato da un altro miner. Dal momento che un nuovo blocco viene aggiunto come figlio del precedente, il quale contiene la transazione di Alice, tale transazione aumenta la difficoltà di computazione della soluzione, rafforzando la fiducia su tali transazioni. Il blocco che contiene la transazione di Alice è considerato come una "conferma" aggiunta alle conferme della transazione in questione. Ogni blocco minato, quindi funge come conferma addizionale ad ogni transazione che contiene.

Dal momento che i blocchi vengono impilati uno sopra l'altro, diventa esponenzialmente difficile invertire la transazione, in tal modo aumenta la fiducia della

rete sulla Blockchain.

Nella *figura 1.12* si può osservare il blocco #277316 che contiene la transazione di Alice.

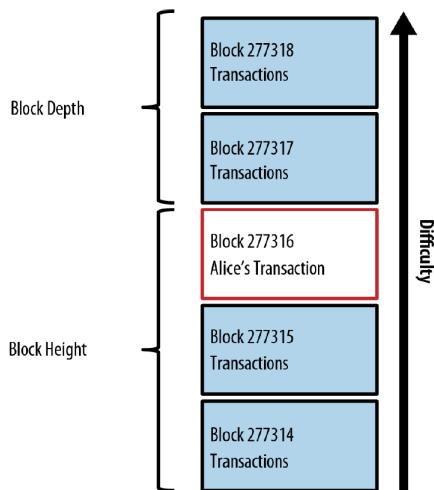


Figura 1.12: *La transazione di Alice inclusa nel blocco #277316*

Ora che la transazione di Alice è inserita all'interno della blockchain, ogni client bitcoin può verificare se la transazione è valida e spendibile. Bob allora può spendere i bitcoin che ha ricevuto da Alice, prendendo tale output e usandolo come input per una nuova transazione. Appena Bob spende tali bitcoin, allunga la catena di transazioni. Si supponga che Bob paghi il suo web designer Gopesh, per un nuovo sito web. Quindi, la catena di transazioni viene mostrata in *figura 1.13*.

1.3.2 Forma Comune delle Transazioni

La forma più comune di una transazione, è un semplice pagamento da un indirizzo ad un altro, in cui quasi sempre viene incluso il "resto" che ritorna al proprietario originale. Questo tipo di transazione ha un solo input e due output, di cui uno ha l'indirizzo pari all'indirizzo del mittente, tutto mostrato in *figura 1.14*.

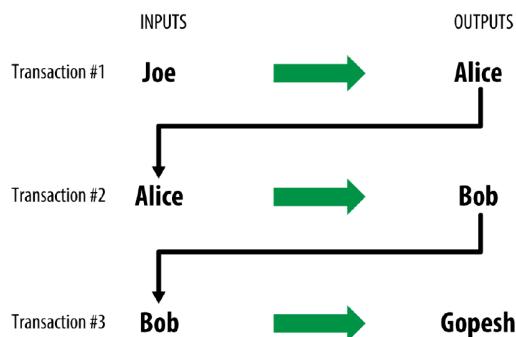


Figura 1.13: *La transazione di Alice come parte di una catena di transazioni che va da Joe a Gopesh*

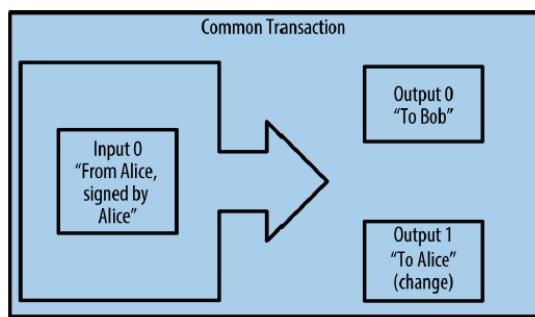


Figura 1.14: *Una comune transazione*

Un'altra forma comune è quella che aggrega molti input in un singolo output (*figura 1.15*). Questo rappresenta il reale scambio di diverse somme di denaro, in un singolo importo. Le transazioni come queste molto spesso sono generate da wallet per pulire piccoli importi che sono stati ricevuti come resto di altri pagamenti.

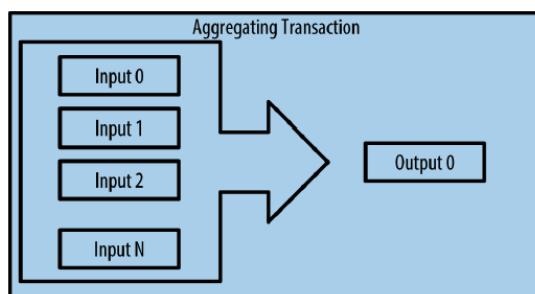


Figura 1.15: *Aggregazione*

Infine, un'altra forma può essere e quella che è il risultato di un'operazione in cui una singola transazione distribuisce il suo input su molti diversi output (*figura 1.16*). Questo tipo di transazione viene utilizzata di solito da entità commerciali per distribuire i fondi, come quando un'azienda distribuisce il denaro come compenso ai propri dipendenti. [Ant14]

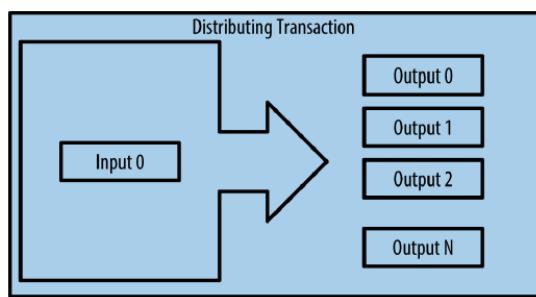


Figura 1.16: *Distribuzione*

1.3.3 Struttura di una Transazione

Una transazione è una struttura dati che racchiude un trasferimento di valori da una sorgente di fondi, chiamata *input*, ad una destinazione, chiamata *output*. Gli input e gli output non fanno riferimento ad account o ad identità. Al contrario, si potrebbe pensare ad esse come delle quantità di bitcoin – gruppi di bitcoin – che sono bloccati da una chiave crittografica. Solo il proprietario, possessore di tale chiave, può sbloccarli. Una transazione contiene dei campi di informazioni, come è mostrato nella *tavella 1.1*.

| Size | Field | Description |
|-------------------|----------------|---|
| 4 bytes | Version | Specifica le regole seguite dalla transazione |
| 1-9 byte (VarInt) | Input Counter | Numero degli input |
| Variable | Inputs | Transazioni in input |
| 1-9 byte (VarInt) | Output Counter | Numero degli output |
| Variable | Outputs | Transazioni in output |
| 4 bytes | Locktime | unix timestamp oppure il numero del blocco |

Tavella 1.1: *La struttura di una transazione*

Il campo **Locktime** (letteralmente "tempo di chiusura") definisce il primo istante in cui una transazione può essere aggiunta alla blockchain. Quando è uguale a zero, indica l'immediata esecuzione.

Invece, se il locktime è maggiore di zero e minore di 500 milioni, è interpretato come l'altezza del blocco, ovvero significa che la transazione non è inclusa nella blockchain prima di specificare altezza del blocco. Se è maggiore di 500 milioni, è interpretato come un timestamp di tipo Unix Epoch (il numero di secondi dal primo Gennaio 1970) e la transazione non è inclusa nella blockchain prima di quell'istante specifico.

1.3.4 Input e Output delle Transazioni

L'elemento fondamentale di una transazione bitcoin è l'**unspent transaction output** (letteralmente "output di transazione non speso"), nella forma contratta **UTXO**.

Gli UTXO sono gruppi indivisibili di monete bitcoin che possono essere sblocati solo dall'utente proprietario. Vengono registrati nella blockchain e riconosciuti come unità monetaria dall'intera rete. La rete bitcoin ha tracciato tutti gli UTXO disponibili, e corrisponderebbero a milioni di bitcoin. Ogni volta che un utente riceve dei bitcoin, tale ammontare è registrato all'interno della blockchain sotto forma di UTXO. Quindi, un utente bitcoin potrebbe seminare migliaia di UTXO attraverso migliaia di transazioni e centinaia di blocchi. In effetti, non c'è nessun tipo di bilancio tra indirizzi bitcoin e output non spesi; ci sono solamente UTXO, che fanno capo a specifici proprietari. Il concetto di bilancio dell'utente in termini di output non spesi è un costrutto creato dall'applicazione che governa il wallet. Il wallet infatti, calcola il bilancio dell'utente scansionando la blockchain e aggregando tutti gli UTXO che appartengono a tale utente.

Un UTXO può avere un valore arbitrario che viene definito in termini di multipli di {satoshi. Come i dollari possono avere dei sottomultipli chiamati

centesimi, i bitcoin possono essere divisi in satoshi. Ogni bitcoin corrispondono a 10^8 satoshi.[Ant14]

1.4 Grafo delle transazioni

Come sottolineato nelle sezioni precedenti, le transazioni sono collegate tra loro sotto forma di chain. Ogni output di ogni transazione diventa un input per la transazione successiva. In questo modo le transazioni formano catene che si possono intersecare tra di loro.

Se consideriamo ogni transazione come un nodo e il collegamento input/output come gli archi, si ottiene un grafo, dove ogni componente connessa è una chain di transazioni connesse tra loro. Per ciò ogni catena di transazioni viene rappresentata come un gruppo di nodi-archi.

Si prenda in considerazione l'esempio visto in precedenza, della catena generata dalle transazioni di Joe, Alice, Bob e Gopesh (*figura 1.13* di pagina 22). Alice riceve i bitcoin da Joe, poi li usa per pagare il caffè al bar di Bob, infine quest'ultimo utilizza una parte di tali bitcoin per pagare il suo web designer Gopesh.

In *figura 1.17* si può vedere come una catena di transazioni viene convertita in un grafo. In questo caso il grafo è un cammino, ovvero una sequenza di nodi collegati da archi, dove ogni nodo ha un solo predecessore ed un solo successore.

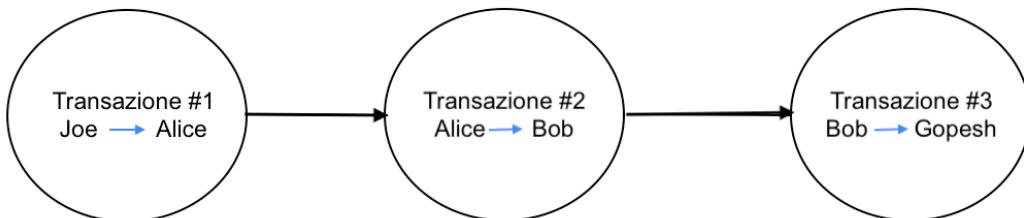


Figura 1.17: *Grafo delle transazioni*

Quindi, ogni transazione corrisponde ad un nodo, il quale è collegato agli altri nodi (alle transazioni) tramite flussi di denaro. Questi ultimi, sono gli output che vengono utilizzate come input da altre transazioni.

Per definizione, una transazione tx è caratterizzata da:

- un insieme di input i_t^1, \dots, i_t^h
- un insieme di output o_t^1, \dots, o_t^k

Ogni input e output sono associati ad un identificatore crittografico, chiamato *indirizzo* e un importo in bitcoin. La transazione tx trasferisce bitcoin dai suoi input ai suoi output.

Siano gli output di tx denotati con *txos*. Ad un certo istante T , ogni txo di una transazione t potrebbe essere stato speso (*stxo*) oppure no (*utxo*). L'unico modo per spendere un certo *utxo* o_t di t , è usarlo come input $i_{t'}$ della transazione t' (con $t \neq t'$). In questo modo, come è stato già sottolineato nei paragrafi precedenti, il flusso dei bitcoin da una tx ad un'altra, crea un entità chiamata *chain of ownership* (letteralmente "catena di proprietà"). [DDP17]

Il *Transaction graph* ("grafo delle transazioni") è un grafo diretto in cui i nodi sono le transazioni e viene indicato con *tx-graph*. I nodi t e t' sono collegati tra loro dall'arco (t, t') , se l'output o_t di t è usato come input $i_{t'}$ di t' . Più nel dettaglio, il *tx-graph* è aciclico, poiché esse vengono generate una volta soltanto, ed è un multigrafo, dal momento che più output di t possono corrispondere a più input di t' (relazione molti-a-molti). Un esempio di tx-graph è illustrato nella figura 1.18.

La transazione (1) ha un input i_1 e tre output (o_1, o_2 e o_3). Tali output sono spesi nelle transazioni 2 e 3. Per quanto riguarda gli input delle transazioni (2) e (3) (come $x1, y1, y2$), si può notare che provengano da altre tx, non considerate all'interno della figura d'esempio. [DDP17]

Considerando che ogni blocco della blockchain può contenere un numero variabile di transazioni – si valuti la possibilità di avere dei blocchi composti all'incirca

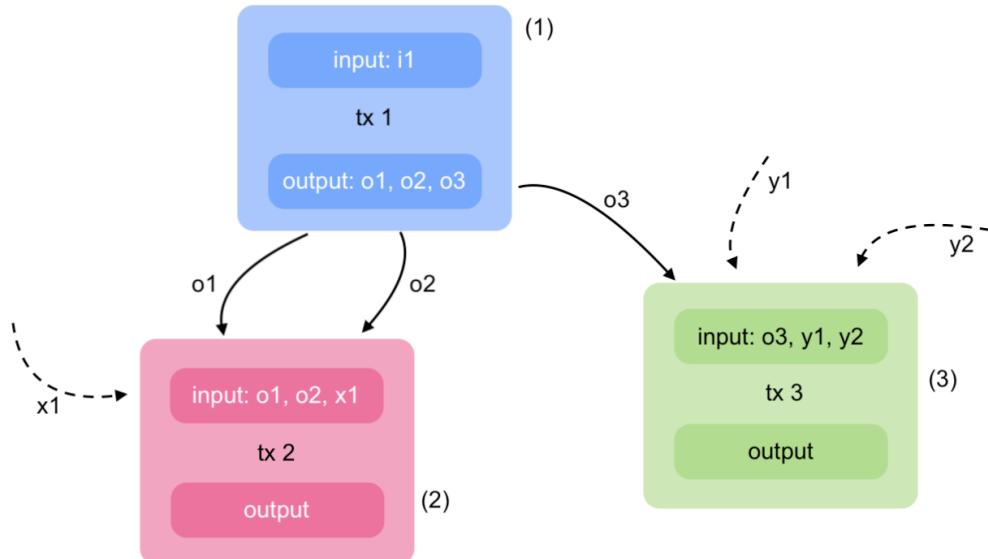


Figura 1.18: Un esempio di tx-graph con 3 tx(1,2 e 3)

da 1000 o 2000 txs – prendendo in esame una gran quantità di blocchi, si potrebbe creare un grafo di grandi dimensioni. Avendo a disposizione una tale mole di dati, si può ragionare su alcune tematiche trattate nei prossimi capitoli.

Capitolo 2

I comportamenti automatici

Per "comportamenti automatici" si intende quegli schemi metodologici che vengono generati automaticamente. In questo caso specifico, la parola "comportamento" va a sottolineare la generazione di catene di transazioni, ovvero la creazione di transazioni collegate tra loro mediante il riutilizzo degli output come input (come è stato visto nel capitolo precedente). Con il termine "automatico" si vuole andare a sottolineare un qualcosa che non è "manuale", ovvero non implica l'intervento dell'uomo.

2.1 Anonimia delle transazioni

Come è stato già sottolineato, le transazioni hanno diverse configurazioni a seconda di quanti input e output intervengono nello scambio. Infatti, è largamente probabile che una transazione abbia un solo input, e molteplici output. Oltre a questa struttura, però, la caratteristica più importante di questo sistema è però il completo anonimato delle transazioni. Sebbene all'interno di una transazione possiamo notare gli indirizzi hash tra cui viene effettuato lo scambio, essi non possono essere riconducibili ad un utente reale. Perciò, se si ha una transazione con un input e due output, si noterà uno scambio tra indirizzi bitcoin.

Per esempio, andando a visualizzare una transazione sul sito [Blo], si ottiene quello che si può notare in *figura 2.1*. Osservando la transazione (4), si osserva che



Figura 2.1: *Una transazione del sito Blockchain.info [Blo]*

il numero degli output è maggiore del numero degli input. Infatti viene effettuato uno scambio di bitcoin tra l'input (1) e gli output (2) (3). Inoltre, al lato della figura sono presenti due bottoni: uno blu, che indica il numero delle conferme che ha ottenuto la transazione, ed uno verde, che indica l'importo bitcoin che viene scambiato. Sopra al bottone verde, gli importi bitcoin ripartiti per i vari indirizzi.

Come ci si aspettava, la transazione identifica degli scambi esclusivamente tra indirizzi bitcoin, rappresentati tramite codici hash. Per andare più a fondo, cliccando su tali codici, si cerca di trovare qualche informazione in più riguardo gli agenti che intervengono nella transazione. Il risultato ottenuto è la *figura 2.2*.

Le tre figure soprastanti, rappresentano i dettagli che si possono estrapolare dal sito, riguardo gli indirizzi che intervengono nella transazione.

Per ogni indirizzo viene indicato il suo identificatore (nel caso di 2.2a l'indirizzo è

3CrtVSfkPJbGwssh9mgXTwuWWWDW3EMrc5k) e il suo Hash 160. L'indirizzo stesso è un codice hash, ma non va confuso con il campo *Hash 160*. Quest'ultimo, invece, rappresenta l'hash della chiave pubblica. Più nello specifico, quindi, viene indicato non solo il nome dell'indirizzo, ma anche la chiave pubblica di tale indirizzo, trasformata tramite la funzione hash di crittografia *sha256*.

Il sito *Blockchain.info* mostra, per ogni indirizzo, anche delle statistiche sulle transazioni in cui è stato coinvolto. Infatti, sotto il campo "transazioni" si può

Indirizzo Bitcoin Gli Indirizzi sono degli identificatori che usi per inviare bitcoin a qualcun altro.

| Sommario | | Le transazioni | |
|-----------|--|-----------------|-------|
| Indirizzo | 3CrVSfkPjbGwssh9mgXTwuWWWDW3EMrc5k | Nr. Transazioni | 2 |
| Hash 160 | 7a8756b3b2c7f81d4309c2a1ce5cfbc510944df2 | Totale Ricevuto | 1 BTC |
| Utensili | Tag correlati - Uscite non utilizzate | Bilancio finale | 0 BTC |

(a) Input

Indirizzo Bitcoin Gli Indirizzi sono degli identificatori che usi per inviare bitcoin a qualcun altro.

| Sommario | | Le transazioni | |
|-----------|--|-----------------|----------|
| Indirizzo | 1BxQiwQeUQaDZxicnwtPTUaKqLFzW6hMMT | Nr. Transazioni | 1 |
| Hash 160 | 782bf0526e66f917aec869e7724531fc9a405776 | Totale Ricevuto | 0.01 BTC |
| Utensili | Tag correlati - Uscite non utilizzate | Bilancio finale | 0.01 BTC |

(b) Output 1

Indirizzo Bitcoin Gli Indirizzi sono degli identificatori che usi per inviare bitcoin a qualcun altro.

| Sommario | | Le transazioni | |
|-----------|---|-----------------|------------|
| Indirizzo | 32FNKJAx2sM7CkVnpJbHXz24WNIX5qjS5y | Nr. Transazioni | 1 |
| Hash 160 | 061e555a3b27acdf5e131670f0bbec048935a31 | Totale Ricevuto | 0.9893 BTC |
| Utensili | Tag correlati - Uscite non utilizzate | Bilancio finale | 0.9893 BTC |

(c) Output 2

Figura 2.2: Dettagli degli input/output

osservare che *Nr. transazioni* indica in quante transazioni è implicato tale indirizzo, il campo *Totale Ricevuto* rappresenta il totale di denaro ricevuto e il campo *Bilancio Finale* dovrebbe corrispondere a quanto denaro è rimasto "non speso" ovvero corrisponde ad un UTXO.

Inoltre, ogni volta che si visualizzano tali dettagli, a lato vengono visualizzati anche i corrispondenti QR code, che servono per richiedere ed effettuare i pagamenti.

L'anonimia delle transazioni e degli indirizzi bitcoin, viene così sottolineata, in modo tale da mostrare che il sistema bitcoin e la blockchain sono stati pensati per rendere gli scambi di denaro sicuri, veloci, ma anche pubblici, in modo da dare possibilità a chiunque di effettuare una transazione senza utilizzare i propri dati personali. Tutto ciò, visto dal lato generale del sistema complessivo. Prendendo ovviamente in considerazione un esempio in cui, due amici si scambiano denaro

in una transazione (come nell'esempio di Alice che compra il caffè al bar di Bob), intuitivamente Alice saprà che tale indirizzo appartiene a Bob e solo a lui, perché potrà verificarlo fisicamente. Perciò, nel più comune caso d'uso di due persone che si conoscono, che si scambiano bitcoin, l'anonimia delle transazioni non viene resa nella misura in cui si vuole far sapere il proprio indirizzo alla persona con cui si sta scambiando denaro. Per esempio, dopo che Alice e Bob hanno fatto la loro transazione, si pensi all'intervento di una persona esterna, Charlie, che non conosce né Alice né Bob di persona. Charlie vuole conoscere chi sono i proprietari degli indirizzi della transazione fatta da Alice a Bob, ma non essendoci una corrispondenza diretta tra l'utente fisico e l'indirizzo bitcoin, Charlie non potrà mai scoprire di chi sono tali indirizzi. L'unico caso in cui Charlie li possa scoprire è quello in cui Bob o Alice glieli riveli spontaneamente.

Perciò grazie a questa caratteristica che distingue le transazioni bitcoin da tutte le forme di scambio di denaro fisico, si è portati ad immaginare un altro possibile scenario che verrà trattato nel paragrafo successivo.

2.2 Bitcoin Mixing

Grazie all'anonimia che contraddistingue gli utenti che intervengono nelle transazioni di bitcoin, è possibile pensare che questa caratteristica faciliti lo scambio di denaro all'interno della rete bitcoin. Il rovescio della medaglia però, viene rappresentato da tutti gli utenti che utilizzano la rete per operazioni illecite.

L'anonimia in questione, permette anche allo stesso utente di possedere due o più indirizzi bitcoin diversi. Gli utenti, possedendo ad esempio due indirizzi diversi, vorranno effettuare un certo spostamento di denaro da un indirizzo all'altro. Supponendo che un utente voglia trasmettere una grande quantità di denaro da un indirizzo ad un altro e rendere le transazioni ancora più anonime, egli si rivolge ad un servizio di **Bitcoin Mixing**.

Il mixing di bitcoin è realmente il riciclaggio di denaro, fatto però attraverso

i bitcoin. Il riciclaggio di denaro è quell'operazione che permette di ottenere dei soldi "puliti" a partire da soldi ottenuti in modo illegale, detti "sporchi". Questo processo viene fatto da chi svolge delle attività illegali. Tutto ciò può essere fatto solamente in modo anonimo, proprio per eludere le forze dell'ordine. L'anonymia necessaria viene sfruttata anche attraverso la rete Bitcoin, proprio a causa delle sue caratteristiche. Infatti, nel caso in cui si voglia effettuare un bitcoin mixing ci si deve rivolgere a dei software chiamati Mixer.

2.2.1 I Mixer: cosa sono?

Il software che offre tale servizio è chiamato **mixer** (letteralmente "miscelatore"). Un mixer svolge quindi l'attività di riciclaggio di bitcoin. Infatti, prendendo in input una certa quantità di bitcoin "sporchi", li "miscola", ovvero crea delle catene di transazioni fittizie con lo scopo di restituire tali soldi "puliti" al mittente (oppure ad un certo indirizzo indicato come output). Il processo viene mostrato nella *figura 2.3*.

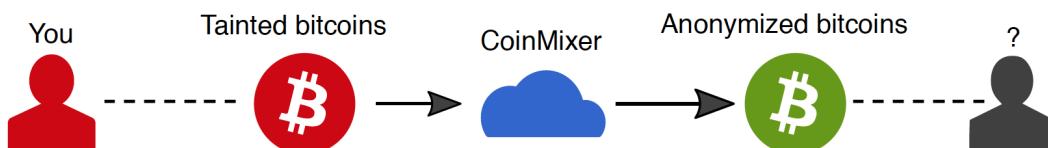


Figura 2.3: *Il mixing di bitcoin* [Mix]

Inizialmente, l'utente intenzionato a usufruire di questo servizio, mette a disposizione del mixer, il denaro che vuole pulire. All'interno di un'interfaccia utente, egli inserisce il proprio indirizzo, l'importo dei bitcoin da riciclare e l'indirizzo di destinazione. Inoltre viene inserito anche un parametro numerico che sta a indicare il numero di ore di ritardo che si può tollerare per ricevere i bitcoin puliti. Ad esempio, se si inserisce 2, il ritardo con cui si riceveranno i soldi sarà pari a 2 ore. Tale parametro rappresenta quindi il tempo necessario affinché il

mixer riesca a riciclare i soldi. Più il ritardo è maggiore, maggiormente i bitcoin saranno mixati e resi anonimi, più è vicino a zero, più le transazioni del mixer saranno più facilmente presenti all'interno dello stesso blocco della blockchain. Esistono dei siti di mixing in cui, nel caso di una grossa quantità di denaro, il ritardo da specificare rappresenta l'intervallo di tempo con cui si vuole ricevere ogni fetta dell'intero importo. In questo modo, ogni intervallo di tempo, si riceverà una parte dei soldi riciclati. Così facendo si eluderà maggiormente qualsiasi tentativo di controllo sulle transazioni e sull'automatismo di tali processi.

Il processo che compie un mixer per riciclare i bitcoin, è quello di creare tante transazioni collegate tra loro in una chain, automaticamente, ovvero seguendo un certo algoritmo. Esso prende in input la cifra da riciclare, e distribuisce il denaro tra gli utenti che fanno parte della rete di "riciclatori" del mixer. Successivamente crea ulteriori transazioni da tali utenti verso l'indirizzo indicato come output. È possibile che il servizio che svolge il riciclaggio abbia un pool di indirizzi dove temporaneamente salva gli importi e, quando richiesto, crei solamente le transazioni di output verso il destinatario. In questo modo non sarà mai possibile collegare il mittente con l'indirizzo di output.

Questo è proprio l'obiettivo dei siti di mixing, ovvero rendere il denaro più anonimo attraverso transazioni che sembrano "normali", mantenendo l'anonimato e sfruttando la struttura della rete e gli utenti connessi. Il termine "mixing" non assume una accezione negativa nel caso in cui si abbia lo scopo di rendere maggiormente anonime le proprie transazioni. Tuttavia, tale termine potrebbe significare che i bitcoin vengono riciclati esattamente come avviene il riciclaggio del denaro sporco nel mondo reale, creando transazioni irrintracciabili e caratterizzate da una forte anonimia.

2.2.2 Mixing nel dettaglio

L'idea base del mixing è quella di assicurare comunicazioni anonime tra due utenti o indirizzi.

La figura 2.3 mostra l'idea di base, la relazione tra Alice e Bob viene nascosta dal servizio di mixing.



Figura 2.4: *Un servizio di mixing, che nasconde la relazione tra Alice e Bob*

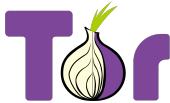
Un mixer prende un insieme di input, che sono stati criptati con la chiave pubblica c_M del mixer e contiene un messaggio criptato $c_A(z_0, m)$, un indirizzo A di destinazione, e una stringa random z_1 che serve per mantenere la dimensione del messaggio in arrivo uguale a quella del messaggio in uscita.

Ricevute le informazioni, il mixer decripta il messaggio, rimuove la stringa random (equazione (2.1)) ottenendo il messaggio di lunghezza giusta, e lo critta di nuovo in base all'indirizzo associato a cui deve mandarlo.

$$c_M(z_1, c_A(z_0, m), A) \rightarrow c_A(z_0, m), A \quad (2.1)$$

Al fine di ridurre il pericolo che un singolo mixer venga attaccato da un hacker, che potrebbe conoscere la relazione tra gli input e gli output, molti mixer vengono collegati insieme, creando un mixing a cascata. L'utente successivamente dovrà decriptare il messaggio con tutte le chiavi pubbliche dei vari mixer, e ciò assicura che ogni mixer ha ricevuto solamente il messaggio criptato e il successivo indirizzo di destinazione.[Mös13]

2.2.3 Tor



A causa di questo processo, tutto ciò non è praticabile all'interno di una rete internet standard. Infatti, per usufruire dei software di mixing occorre essere connessi ad internet tramite il browser Tor.

In informatica **Tor**(acronimo di The Onion Router) è un sistema di comunicazione anonima per Internet basato sul protocollo di rete di *onion routing*. Tramite l'utilizzo di Tor è molto più difficile tracciare l'attività Internet dell'utente; infatti l'uso di Tor è finalizzato a proteggere la privacy degli utenti, la loro libertà e la possibilità di condurre delle comunicazioni confidenziali senza che vengano monitorate. *Onion routing* è una tecnica per rendere anonime le comunicazioni all'interno della rete. In una rete onion (letteralmente "cipolla"), i messaggi sono incapsulati in "strati" di crittografia che vengono paragonati agli strati di una cipolla. [Wik11b]

Il mixing dei bitcoin e lo scambio illecito di denaro, può essere intercettato in una rete internet standard. Tor, quindi è una rete secondaria che permette lo scambio di transazioni di denaro (e altro), che su un normale browser non sarebbe possibile. Infatti, essendo propriamente una rete nascosta è difficile intercettarne i pacchetti e capire quali scambi vengano fatti. Questo perché, proprio il protocollo con cui è stata pensata non permette che si spoofi la comunicazione tra due utenti. La caratteristica principale si può dedurre dal nome della tecnologia, the onion router. Letteralmente "onion" significa cipolla, e sta ad indicare la stratificazione che accomuna tali pacchetti con il tubero puzzolente. I pacchetti della rete tor sono letteralmente stratificati, ossia ogni volta che viene aggiunto uno strato, si applica una funzione di crittografia. Una terza persona che riesce a sniffare un pacchetto di questo tipo, non riuscirà ad aprirlo, anche conoscendo le chiavi del ricevente e del destinatario.

Tali pacchetti sono adatti alla topologia della rete tor, poichè la rete è pensata per rendere anonime le comunicazioni. Tale anonimia è resa dal fatto che i pac-

chetti non seguono un algoritmo di instradamento, ma letteralmente "rimbalzano" da un router ad un altro, arrivando infine a destinazione. Proprio per tale caratteristica, la connessione tor appare lenta, poichè la trasmissione del pacchetto impiega più tempo ad arrivare a destinazione, più di un normale pacchetto.

I messaggi, quando vengono trasmessi, è come se seguissero la strada più lunga attraverso la rete, passando anche più volte per gli stessi router. La stratificazione infatti viene realizzata ad ogni singolo passaggio di un pacchetto attraverso un router. Così facendo, il successivo router destinatario, toglierà man mano uno strato di crittografia.

Capitolo 3

Analisi sul grafo delle transazioni

3.1 Esempio di grafo reale

Per analizzare meglio il sistema Bitcoin, viene creato un grafo delle transazioni, utilizzando dei dati reali. Nella *figura 3.1* è visualizzato il grafo generato dall'elaborazione solo di 5 blocchi, in particolare dal blocco #417113 al #417117.

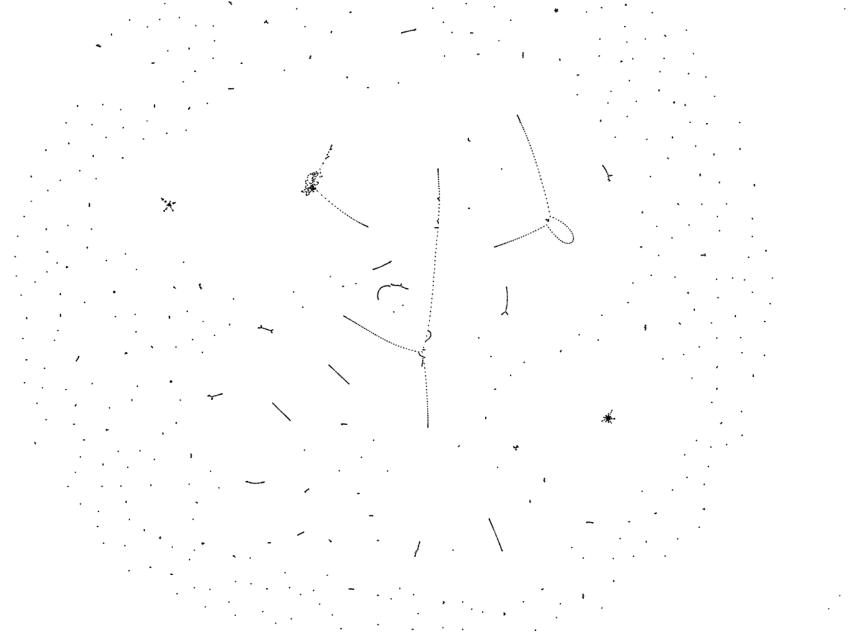


Figura 3.1: *Esempio di tx-graph reale*

Tale grafo viene visualizzato utilizzando il metodo *force-directed*, ovvero assegnando ad ogni arco delle forze attrattive in modo da disporre i nodi del grafo. A colpo d'occhio, al centro dell'immagine si notano alcune linee, che rappresentano delle sequenze di nodi collegati tra loro uno dopo l'altro. Guardando man mano verso l'esterno, si notano molti nodi scollegati dal grafo: molto probabilmente rappresentano transazioni i cui output non sono stati spesi in transazioni incluse nell'intervallo di blocchi considerato, oppure che non sono stati spesi affatto.

3.2 Lunghe catene di transazioni

La media del numero di transazioni Bitcoin giornaliere è all'incirca 280'000, valutato a Maggio 2017. Questo numero è conosciuto per essere largamente influenzato da alcuni fenomeni. Il mescolamento dei wallets e il mixing dei soldi sono solamente due esempi di attività che generano transazioni le quali non sono direttamente collegate con il reale acquisto di beni o servizi. Un altro esempio viene dall'attività di alcuni *exchange*(organizzazioni che permettono agli utenti di scambiare Bitcoin per la valuta legale e viceversa) che usano lunghe catene di transazioni per emettere pagamenti verso degli utenti che hanno deciso di ritirare i Bitcoin.

Questo tipo di organizzazioni aggregano diversi depositi di bitcoin dentro una singola grande transazione. Sebbene tali catene di transazioni vengono innescate da attività umane (per esempio utenti che decidono di scambiare bitcoin), esse sono comunque generate da un meccanismo automatico, il quale incrementa il numero giornaliero delle transazioni associate al commercio esplicito di beni o servizi tra utenti.

Sulla base di ciò, si pensa che le organizzazioni con interessi sui Bitcoin, generano transazioni con il mero obiettivo di attrarre investitori e incrementare il tasso di scambio. Essendo generate da software, queste transazioni "artificiali" spesso introducono dei pattern regolari all'interno della Blockchain. E tali pat-

tern vengono generati proprio perchè le catene di transazioni sono generate da un software programmato.

Queste considerazioni vengono trattate approfonditamente nel paper [DDP17]. L'argomento principale di tale documento è il parametro *LLC*, acronimo di *length of the longest chain*, ovvero "lunghezza della catena più lunga". Ogni transazione viene etichettata con un LLC, ovvero un numero che rappresenta la catena più lunga a cui appartiene la transazione. Per assegnare questo parametro, viene precedentemente effettuata un'analisi sul grafo delle transazioni. Quindi, per ogni transazione, ovvero per ogni nodo, viene calcolato l'LLC e assegnato alla singola transazione. Infine vengono fatte delle analisi sui dati e sulla distribuzione del parametro rispetto ai blocchi considerati nell'esecuzione dell'algoritmo.

Capire se una chain di transazioni è stata generata automaticamente o riflette una chain creata da acquisti reali di umani, è una sfida stimolante. Infatti, la *chain of ownership* ("catena di proprietà") introduce naturalmente lunghe chain di transazioni nel tx-graph. Quello che non è naturale è la velocità con cui tali chain vengono generate.

3.3 Analisi sul tx-graph

Per effettuare analisi sul grafo delle transazioni, si è pensato di considerare dei parametri fondamentali nello studio di un fenomeno periodico come l'emissione di transazioni automatiche: la frequenza e la varianza.

Dal momento che le transazioni automatiche sono, molto spesso, generate a velocità elevata, si pensa di dare un parametro che quantizzi tale velocità. Tale parametro potrebbe essere la frequenza, ovvero quante transazioni vengono emesse nell'unità di tempo.

Così facendo si potrebbe analizzare la frequenza delle singole catene di transazioni e dedurre quali sono quelle automatiche e non. Purtroppo, non si sa ancora

in base a quale regola tali chain possono essere classificate. Si pensa soprattutto che questa analisi possa dare un contributo alla scoperta di nuovi andamenti.

Un altro parametro definito per l'analisi del grafo, è la varianza, ovvero la misura di quanto ci si discosta dalla media delle frequenze. Sebbene, la varianza debba essere calcolata istantaneamente, e la frequenza globalmente, si può fare una analisi "online". Infatti, man mano che si visita il grafo topologicamente e si scoprono nuove chain, si può calcolare la frequenza della chain fino alla transazione corrente, poi si calcola la varianza, e si potrebbe scegliere il cammino più lungo a varianza minima. In questo modo si potrebbe trovare una catena di transazioni generata automaticamente.

3.4 Formalizzazione delle definizioni

In questa sezione si tratterà di come sono state definiti alcuni parametri per lo studio e l'analisi che ha portato poi alla formulazione di una tesi sul comportamento automatico del tx-graph. In particolare vengono formalizzate alcune definizioni per il calcolo della frequenza e della varianza, due parametri che intervengono di pari passo nell'analisi delle lunghe catene di transazioni.

3.4.1 Frequenza delle transazioni

Dal momento che tali catene di transazioni vengono generate automaticamente, esse vengono emesse ad una velocità elevata. Come si è già sottolineato nel capitolo precedente, un bitcoin mixer prende in input l'importo in bitcoin, l'indirizzo di destinazione e il tempo di ritardo con cui si vogliono ricevere tali bitcoin. In base al tempo inserito, il mixer creerà una certa quantità di transazioni collegate; più è maggiore il tempo, più transazioni verranno emesse, più è minore, meno si otterrà un anonimato su tali bitcoin.

Proprio per questo motivo, il mixer impiegherà una certa quantità di tempo per creare le transazioni che mischiano i bitcoin. Essendo un software program-

mato, esso genera le transazioni velocemente e ad intervalli regolari. Perciò, le catene generate da un mixer in modo automatico, verranno emesse in modo molto rapido ed ogni transazione di tali catene risulterà essere alla stessa distanza di tempo tra la sua precedente (se è presente) e la sua successiva nella catena.

Nella *figura 3.2* si osserva lo zoom su una catena del grafo di pagina 37.



Figura 3.2: *Zoom di una chain*

Dal momento che le chain vengono generate ad un'elevata velocità, si può pensare allo studio della densità temporale di tali transazioni. A partire da queste considerazioni, si può definire il concetto di **frequenza**.

La frequenza f_T tra le n transazioni t_0, \dots, t_n , perciò, viene definita nella formula 3.1.

$$f_T = \frac{n}{ts(t_n) - ts(t_0)} \quad (3.1)$$

Dove f_T è la frequenza totale tra un numero n di transazioni. $ts(t_0)$ è il timestamp della transazione iniziale t_0 , mentre $ts(t_n)$ è il timestamp dell'ennesima transazione t_n . Tramite questa formula si ottiene la frequenza totale delle transazioni emesse in un certo intervallo di tempo.

Il concetto di frequenza totale, deriva dal concetto di frequenza fisica. Si definisca la **frequenza istantanea** come:

$$f_{ist}(t_n) = \frac{1}{T} \quad \text{con} \quad T = ts(t_n) - ts(t_{n-1}) \quad (3.2)$$

La definizione di frequenza istantanea viene ripresa da quella della frequenza in fisica. In fisica la frequenza di un fenomeno, viene data dal numero degli eventi che vengono ripetuti in una data unità di tempo. L'unità di tempo T in fisica è chiamata periodo, ed è l'intervallo temporale tra due eventi.

Nel caso del tx-graph, gli eventi sono le transazioni, e il periodo è calcolato tramite la differenza tra il timestamp dell'ultima transazione e quello della transazione iniziale, come si può notare nell'equazione 3.2. La frequenza istantanea rappresenta il singolo valore di frequenza calcolato su due transazioni e su un singolo intervallo di tempo. Tale parametro non ha un ruolo significativo, se viene considerato a sè stante. Esso può essere inserito come parte della formula del calcolo della frequenza totale f_T .

Nell'equazione 3.1 la frequenza totale viene rappresentata come il rapporto tra la cardinalità delle transazioni diviso l'intervallo totale di tempo, ovvero il tempo che intercorre tra la transazione iniziale e quella finale. Allo stesso modo, se si riscrive la formula tenendo conto della media delle frequenze istantanee si otterrà:

$$f_T = \frac{f_{ist}(t_1) + \cdots + f_{ist}(t_n)}{n} = \quad (3.3)$$

$$= \left(\frac{1}{ts(t_1) - ts(t_0)} + \cdots + \frac{1}{ts(t_n) - ts(t_{n-1})} \right) \cdot \frac{1}{n} = \quad (3.4)$$

$$= \frac{n}{(ts(t_1) - ts(t_0)) + \cdots + (ts(t_n) - ts(t_{n-1}))} \quad (3.5)$$

dove il denominatore dell'equazione alla riga 3.5 è la somma di tutti i singoli intervalli di tempo tra le transazioni t_0 e t_n . Empiricamente, da tale somma si ottiene quindi l'intero intervallo di tempo tra la prima e l'ultima transazione, quindi il periodo totale sarà pari a $ts(t_n) - ts(t_0)$.

L'equazione al punto 3.5 si può scrivere:

$$f_T = \frac{n}{ts(t_n) - ts(t_0)} \quad (3.6)$$

Quindi, si ottiene la formula 3.1 di pagina 41. Sebbene le formule di frequenza totale e istantanea siano diverse tra loro, concatenando le varie frequenze istantanee si può arrivare a ottenere la formula f_T .

Tutto ciò, al fine di sottolineare come la frequenza influenzi lo studio e l'analisi del grafo delle transazioni. Per ogni singola catena di transazioni viene calcolata la frequenza media tramite la formula 3.6. Dalla frequenza, viene estrapolato un ulteriore parametro, la varianza, di cui si tratterà nella successiva sezione.

3.4.2 Varianza

Per analizzare meglio il grafo delle transazioni viene introdotto un ulteriore parametro la **varianza**, che dipende direttamente dalla frequenza. La nozione di varianza viene ripresa dalla nozione di varianza statistica. infatti per definizione la varianza fornisce una misura della variabilità dei valori assunti da una certa variabile, ovvero di quanto i valori di tale variabile si discostino dalla loro media aritmetica. [Wik11c]

Dal punto di vista statistico, la varianza viene calcolata come la media quadratica dei discostamenti delle variabili prese in considerazione con la media aritmetica delle variabili stesse. Più generalmente la varianza va a misurare la variazione di un certo parametro rispetto al suo andamento medio.

Per adattare il concetto di varianza allo studio del tx-graph, si è deciso di utilizzare la formula 3.7 per il calcolo della varianza.

$$v = \frac{\sum_{i=1}^n (f_{ist}(t_i) - \mu)^2}{n} \quad (3.7)$$

In questa formula, al denominatore è presente una sommatoria che sta a significare la somma delle varie quantità $f_{ist}(t_i) - \mu$ cioè i singoli discostamenti delle frequenze istantanee dalla media μ .

La media infatti può essere calcolata con la formula standard della media aritmetica, sulle frequenze istantanee, ottenendo così:

$$\mu = \frac{\sum_{i=1}^n f_{ist}(t_i)}{n} \quad (3.8)$$

Gli studi che sono stati effettuati sul grafo delle transazioni, hanno tutti alla base queste nozioni di matematica. Infatti, la varianza viene calcolata in base alle frequenze istantanee, e quest'ultime vengono calcolate attraverso i timestamp delle singole transazioni. Le singole f_{ist} rappresentano la frequenza con cui viene emessa una transazione, ovvero l'intervallo di tempo con cui una singola transazione viene effettuata. In questo modo si può capire quanto velocemente è stata emessa, e se eventualmente superi una certa soglia, si potrebbe intuire che essa è stata generata automaticamente. Empiricamente si può dedurre che un mixer può generare transazioni a intervalli regolari e velocemente. L'intervallo regolare è possibilmente deducibile con il calcolo della varianza, poiché più la varianza è bassa, più è possibile che le frequenze seguino un andamento regolare, uguale all'andamento del valor medio, ovvero del valore che più esprime la regolarità di una variabile.

Capitolo 4

Approccio utilizzato

In questo capitolo verranno trattate le metodologie con cui si è deciso di approfondire con l'analisi del tx-graph. In particolare si è preferito adottare un approccio più incentrato sullo studio degli andamenti e della frequenza delle varie catene di transazioni.

4.1 Networkx per la creazione di grafi

Per questo progetto, si è deciso di utilizzare il linguaggio di programmazione Python, in aggiunta con l'utilizzo di varie librerie, in particolare *Networkx*, la quale permette la creazione, la gestione e l'analisi dei grafi.

Alcune caratteristiche di tale libreria sono:

- Strutture dati per grafi diretti, indiretti
- Molti algoritmi standard per la gestione dei grafi
- Software open source

Per la gestione di un grafo come il tx-graph, ovvero un grafo generato da tutte le transazioni dei blocchi che vengono considerati nel dataset, si è deciso di utilizzare un *MultiDiGraph*, un grafo diretto con auto archi ed archi paralleli.

4.2 Dataset

Il dataset utilizzato è un insieme di blocchi reali. Essi sono stati scaricati precedentemente e salvati su file in formato json. Ogni singolo blocco è caratterizzato da un numero, il numero che caratterizza l'ordine nella blockchain, e il file che rappresenta i dati di tale blocco ha il nome che corrisponde a tale numero con l'estensione `.json`.

4.2.1 Struttura dati di un blocco

Per ogni blocco, esiste un file `<numblocco>.json`, con determinate caratteristiche.

Listing 4.1: Prime righe del file `417113.json`

```

1  {"txs": [
2      "valueOut": 25.29311068,
3      "isCoinBase": true,
4      "vout": [{"...}],
5      "blockhash": "000000000000000000fa1d0daee52356ca68a5964efc389e237a5cf589fe82d6",
6      "vin": [{"...}],
7      "txid": "c4be4eda4bb0fce550f89b7ed00d1202a8dd41bb62c2a1254b61b96b0c3e627b",
8      "blocktime": 1466381104,
9      "version": 1,
10     "confirmations": 59748,
11     "time": 1466381104,
12     "blockheight": 417113,
13     "locktime": 863856133,
14     "size": 185
15 },
16 {
17     "valueOut": 2.39,
18     "vout": [{"...}],
19     "blockhash": "000000000000000000fa1d0daee52356ca68a5964efc389e237a5cf589fe82d6",
20     "valueIn": 2.4,
21     "fees": 0.01,
22     "vin": [{"..."}, {"...}],
23     "txid": "d064980449192201f5d23cf8f6a4fec1ddff58ec9d6692c8996c1007b98250a",
24     "blocktime": 1466381104,
25     "version": 1,
26     "confirmations": 59748,
27     "time": 1466381104,

```

```
28      "blockheight": 417113,  
29      "locktime": 0,  
30      "size": 339  
31 },
```

Nel codice 4.1, vengono indicate solamente le prime righe del file 417113.json. Dal momento che un file di questo tipo rappresenta un singolo blocco, al suo interno si potrà trovare una lista di transazioni. Il file, essendo di tipo json, viene rappresentato come un dizionario in cui i dati sono rappresentati nel formato chiave-valore.

Nella riga 1, al corrispondere dell'attributo "txs" si ha una lista di transazioni. Dal momento che queste sono le prime righe del file, la prima transazione di un blocco è essenzialmente quella generata dal miner quando il blocco viene minato. Di conseguenza come prima transazione si troverà sempre un oggetto che avrà il campo "isCoinBase" uguale a True, che sta a significare che tale transazione è la prima del blocco e non ha transazioni in entrata.

Dal secondo oggetto della lista in poi, si ha una configurazione standard per tutte le altre transazioni che fanno parte del blocco. Come primo campo si ha "valueOut", il quale indica il valore totale di bitcoin che vengono emessi dalla transazione. Il campo "vout" rappresenta invece la lista di transazioni verso le quali gli output vengono spesi. Esiste anche il campo "vin" che raccoglie la lista delle transazioni di cui gli output vengono spesi dalla transazione in questione.

Il campo "txid" rappresenta l'id della transazione, ovvero il principale identificativo dell'oggetto preso in considerazione.

Nella singola transazione, si possono notare anche parametri che indicano attributi legati all'appartenenza al blocco:

- "blockhash" rappresenta l'identificativo hash del blocco di appartenenza della transazione
- "blocktime" è il timestamp del blocco

- "blockheight" è il numero che identifica il blocco all'interno della blockchain
(in questo caso "blockheight" = 417113)

Inoltre, si può notare anche il campo "fees" che rappresenta appunto le fee (cioè le tasse), che tale transazione versa al miner che è riuscito a minare la blockchain.

Nell'esempio del codice sopra citato, vengono riscritte le prime righe del file json, che comprendono le prime due transazioni del blocco. In realtà, come si è già sottolineato, la il primo oggetto della lista è la transazione che fornisce la ricompensa al miner, non è una transazione vera e propria. Perciò il codice visualizza solamente la prima transazione vera e propria, che è la seconda nella lista di oggetti.

Per ogni transazione sono conservati anche i parametri del blocco di appartenenza, come il numero del blocco, il suo codice hash e il suo timestamp. Naturalmente tali parametri sono uguali per tutte le transazioni dello stesso blocco. Tutto ciò facilita molto l'analisi e la creazione del tx-graph poichè ogni transazione ha già il parametro del blocco di appartenenza, senza bisogno di creare ulteriori strutture dati.

Questo esempio riguarda solamente una transazione di un blocco, ma l'analisi in questione viene fatta prendendo in considerazione al più 1000 blocchi, che vengono tutti letti come file json.

Si considerino due possibili situazioni: inizialmente è stata fatta un'analisi su un primo dataset che comprende i blocchi dal 417113 al 417256, proprio come nel paper [DDP17]. Successivamente, come nel suddetto paper, vengono considerati i blocchi dal 413000 al 419143. I due dataset hanno lo scopo di rappresentare intervalli di tempo diversi. Infatti il primo, sta a rappresentare l'attività di emissione di transazioni che comprende 24 ore. Il secondo invece, rappresenta quanto una macchina può computare al livello di memoria. Per motivi di efficienza e mancanza di risorse computazionali elevate, il secondo dataset è stato ridotto a solamente 1000 blocchi, ovvero dal 413000 al 414000. In questo modo, anche con il secondo dataset è stato possibile generare dei risultati adeguati che poi

sono stati infine graficati, per osservare meglio gli andamenti e le periodicità più importanti.

Perciò vengono definiti due dataset $D1 = (417113, 417256)$ e $D2 = (413000, 414000)$, che successivamente verranno utilizzati per creare i grafi $G1 = G(417113, 417256)$ e $G2 = G(413000, 414000)$.

4.3 Cammino più lungo a varianza minima

Come è stato ampiamente illustrato nel capitolo precedente, l’analisi del grafo delle transazioni verte specialmente sul problema di trovare un cammino, ovvero una catena di transazioni, tale che sia il più lungo e contemporaneamente sia a varianza minima.

Si è deciso di considerare questi parametri, proprio perchè è possibile che il risultato restituito sia proprio corrispondente ad una catena di transazioni generate automaticamente da un mixer. Purtroppo questo processo di verifica, non permette di essere certi di quello che si sta valutando, ma potrebbe porre le fondamenta per un successivo studio approfondito di tali comportamenti.

Per calcolare il cammino più lungo a varianza minima è stato definito un algoritmo. Esso effettua un’analisi sul grafo delle transazioni che viene creato nel prossimo paragrafo.

4.3.1 Creazione grafo delle transazioni

Prima di parlare dell’algoritmo, verrà illustrato come sono stati letti i dati e come viene creato il grafo.

Utilizzando le funzioni di networkx, diventa semplice costruire un grafo in python. Infatti basta usufruire delle funzioni di tale libreria.

Il metodo che crea il grafo prende in input i numeri dei blocchi come range, e legge, in ordine, tutti i file json. Leggendo i singoli json, crea una lista di oggetti, dove ogni oggetto è un blocco con la rispettiva lista di transazioni. Dopo

la lettura, viene iterato sulla lista di blocchi, viene presa la lista di transazioni, e vengono creati i nodi del grafo, dove ogni nodo corrisponde ad una transazione.

Per creare gli archi, si cicla su tutte le transazioni e si legge il campo "vin" e "vout". Rispettivamente, il primo indica gli archi entranti in quella transazione, e il secondo gli archi uscenti. In base a questi campi, si cicla sulle due liste "vin" e "vout" e si verifica che le transazioni presenti in tali liste siano corrisposte da una loro rappresentazione all'interno del grafo sotto forma di nodi.

Ovviamente, se si considera un numero limitato di blocchi, è molto probabile, che singole transazioni vengano rappresentate esclusivamente da un singolo nodo che costituisce una componente a sé stante. Infatti è importante gestire le eccezioni in questo caso specifico.

Nel codice seguente, si può osservare la modalità di creazione del grafo.

```

1 def create_graph(start_block , end_block):
2     blocks = read_blocks(start_block , end_block)
3     G = nx.MultiDiGraph()
4     ts = blocks[0][ 'ts '], dic ={}
5     for i in blocks:
6         for j in i[ 'txs ']:
7             count = count+1
8             temptxid = j[ 'txid ']
9             G.add_node(count , txid=temptxid , blockhash=i[ 'blockhash '],
10                         numblock=i[ 'numblock '], blocktime=i[ 'blocktime '], ts=0)
11             dic[temptxid] = count
12             currId = dic[j[ 'txid ']]
13             for k in j[ 'vout ']:
14                 try:
15                     outNodeId = dic[k[ 'spentTxId ']]
16                     if not G.has_edge(currId , outNodeId):
17                         G.add_edge(currId , outNodeId)
18                 except KeyError: break
19             for h in j[ 'vin ']:
20                 try:
21                     inNodeId = dic[h[ 'txid ']]
22                     if not G.has_edge(inNodeId , currId):
23                         G.add_edge(inNodeId , currId)
24                 except KeyError: break

```

Nella riga 2, viene richiamato il metodo che legge i file .json che corrispondono ai blocchi, e restituisce un file di tipo lista. Infatti la variabile "blocks" corrisponde ad una lista di oggetti, dove per ogni oggetto si ha un blocco, con le sue caratteristiche e la sua lista di transazioni. Come si può notare, infatti, nella riga 5, si fa un'iterazione su tale lista, in modo da creare il grafo. Per ogni blocco, quindi, si itera sulla lista di transazioni e, come già detto in precedenza, si crea un nodo del grafo per ogni transazione.

Come già sottolineato nei paragrafi precedenti, si è preferito utilizzare la libreria Networkx, per la creazione di grafi. Nella riga 3 viene inizializzata la variabile che corrisponde al grafo. Tale libreria, per aggiungere i nodi al grafo, utilizza il metodo "add_node", riga 9, che prende come input l'id con cui viene identificato il nodo dalla libreria stessa e diversi parametri che si vogliono aggiungere al nodo. Infatti "count" in questo caso è l'id del nodo all'interno del grafo G, ed è un contatore che rappresenta il numero di tutte le transazioni all'interno di tutti i blocchi letti. Perciò, in ordine di lettura, ogni transazione avrà un id corrispondente ad un contatore che conta nel complessivo tutti i nodi del grafo.

Così facendo i nodi risultano numerati da 1 ad un numero pari alla cardinalità di G. Per creare gli archi invece, per ogni transazione appartenente alla lista di transazioni del blocco, esiste una lista di archi entranti e uscenti. Per trovare una corrispondenza con tali transazioni collegate, occorre utilizzare un dizionario "dic" che ha lo scopo di stabilire una corrispondenza diretta tra l'id del nodo del grafo e l'id della transazione. Purtroppo questo procedimento si è reso necessario poichè non è presente un metodo che fa questa cosa, nella libreria Networkx.

Infine, vengono creati gli archi che collegano i nodi del grafo, con il metodo "add_edge", che prende in input gli id dei nodi da collegare. Infatti, è proprio per questo motivo che si utilizza il dizionario, perché nell'aggiunta degli archi, il metodo prende come input gli id dei nodi, invece degli id delle transazioni.

4.3.2 Assegnamento timestamp alle transazioni

Per implementare l'algoritmo principale del progetto, ovvero quello che calcola il cammino più lungo a varianza minima, occorre conoscere i singoli timestamp delle transazioni. Purtroppo, data la sua struttura, la blockchain non prevede l'assegnamento di timestamp alle transazioni, solamente ai blocchi. Perciò si è deciso di implementare un algoritmo anche per l'assegnamento dei timestamp alle transazioni.

Tale algoritmo prevede di ricavare i vari timestamp delle singole transazioni, dai timestamp del blocco di appartenenza. Sebbene si potrebbe fare una banale divisione tra il tempo che è presente tra i blocchi e il numero delle transazioni del blocco, l'algoritmo non è stato pensato in modo così semplice.

Per ogni blocco, si definisce un intervallo di tempo con il blocco successivo. Infatti, il timestamp del blocco che viene salvato dalla blockchain, sarebbe l'istante finale con cui vengono salvate le transazioni nel blocco. Perciò per calcolare l'intervallo di tempo che si è impiegato per definire tale blocco, occorre fare una sottrazione tra il timestamp del blocco e quello del blocco precedente. Nel caso particolare del primo blocco invece, si prende come considerazione un intervallo di tempo prestabilito uguale a 10 minuti. Ogni dieci minuti viene creato un blocco, perciò il primo blocco come intervallo di tempo, sceglierà il suo timestamp, a cui vengono sottratti 600 secondi.¹

4.3.2.1 Timestamp blocchi

Per assegnare i timestamp alle transazioni, l'algoritmo da implementare necessita di un campo principale: il timestamp del blocco. Tramite l'osservazione dei vari timestamp ci si è resi conto che l'ordine numerico dei blocchi può non corri-

¹il timestamp del blocco è nel formato *unixstamp* e viene rappresentato in secondi a partire dalla mezzanotte del 1º Gennaio 1970. Per esempio, per il blocco 417113 il timestamp è 1466381104 che corrisponde al 20 Giugno 2016 alle ore 00:05:04.

spondere all'ordine temporale con cui i blocchi vengono emessi. Si prenda come esempio il range di blocchi (417113, · · · , 417116), con i seguenti timestamp:

- 417113 ts=1466381104 (20/06/16 00:05:04)
- 417114 ts=1466381748 (20/06/16 00:15:48)
- 417115 ts=1466381660 (20/06/16 00:14:20)
- 417116 ts=1466382461 (20/06/16 00:27:41)

Si nota immediatamente come il timestamp del blocco 417114 è maggiore del blocco 417115, ovvero è stato ricevuto dopo il blocco 417115. Quindi in questo caso, l'ordine numerico dei blocchi non corrisponde all'ordine cronologico.

Le cause per cui si sia verificata tale situazione possono essere molteplici. Probabilmente il blocco è stato sincronizzato in un secondo momento, magari a causa di qualche guasto tecnico che non ha permesso la perfetta ricezione e distribuzione di informazioni attraverso la rete. Potrebbe anche essersi verificata una fork, dove il blocco 417115 è stato ricevuto prima della scelta della catena principale a cui apparteneva anche il blocco 417114.

Questa situazione genera un uteriore interrogativo: è possibile che i timestamp assegnati non siano corrispondenti alla reale ricezione dei blocchi? Assumendo di poter rispondere affermativamente, ci si rende conto che si potrebbero quindi mettere in discussione tutti i valori di timestamp di quasi tutti i blocchi. Perciò le informazioni ottenute dalla blockchain, possono anche non essere certe.

Sulla base di questa assunzione, si è pensato quindi di assegnare arbitrariamente i timestamp ai blocchi. Così facendo si potrebbe ovviare al problema dell'ordine numerico dei blocchi. Infatti, si è deciso di rispettare l'ordine numerico per correggere le informazioni sul tempo di ricezione dei blocchi.

Per fare ciò, viene preso l'intervallo dei blocchi, calcolato il tempo totale che intercorre tra un blocco e l'ultimo della lista, e si calcolano i singoli intervalli da assegnare a ciascun blocco. Tali intervalli singoli vengono quindi calcolati

dividendo semplicemente il tempo totale per il numero dei blocchi. Come risultato i blocchi hanno tutti lo stesso intervallo di tempo.

```

1 def set_blocks_timestamp(blocks):
2     numblocks = len(blocks)
3     start_interval = blocks[0][ 'blocktime' ]
4     end_interval = blocks[numblocks - 1][ 'blocktime' ]
5     full_interval = end_interval - start_interval
6     single_interval = full_interval/float(numblocks - 1)
7     count = 0
8     for i in range(numblocks - 1):
9         blocks[i][ 'blocktime' ] = start_interval + (count*single_interval)
10        count = count+1

```

il metodo void *set_blocks_timestamp* prende come parametro la lista di blocchi e assegna i timestamp corretti ad ogni blocco. La variabile *single_interval* rappresenta il singolo intervallo che viene assegnato da un blocco ad un altro.

Così facendo, per ogni blocco si otterrà un nuovo timestamp calcolato arbitrariamente. Si potrebbe pensare che in questo modo l'analisi non potrebbe coincidere con i dati reali. Si è pensato infatti anche a questa eventualità. Sebbene i blocchi siano "spaziati" temporalmente in modo equivalente, ogni blocco contiene un numero qualsiasi di transazioni, che può andare da poche centinaia fino ad arrivare ad alcune migliaia (almeno per quello che si è riusciti a visionare). In questo modo, l'analisi sarebbe comunque corretta, se si analizza in base al numero di transazioni.

4.3.2.2 Algoritmo per assegnare i timestamp

Dopo aver assegnato i corretti timestamp ai vari blocchi, è stato formulato un algoritmo per meglio etichettare le transazioni, anche in base al numero di transazioni contenute in un blocco, e in base anche alle chain di nodi che vengono comprese tra due o più blocchi.

L'idea di base è quella di considerare tutte le possibili catene di transazioni, ordinarle in base al numero di elementi e assegnare i timestamp. Ogni nodo della chain più lunga avrà come intervallo di tempo tra un nodo e l'altro, l'intervallo di

tempo in cui è compreso il blocco, diviso il numero di elementi della chain. Dopo tale assegnazione, viene successivamente considerata la seconda catena più lunga. Si itera su tale catena, e per ogni nodo, si va a esplorare il nodo precedente nella catena. Se il nodo non ha predecessori, allora gli verrà assegnato il timestamp del blocco precedente, ovvero l'istante iniziale del tempo di spaziatura del blocco di appartenenza. Se invece il nodo avesse dei predecessori, si assegnerebbe la media dell'intervallo di tempo tra il successore e il predecessore. Nel caso di considerazione di tali catene, normalmente il predecessore, se è presente, è unico.

In seguito verrà mostrato il dettaglio dell'algoritmo.

Algorithm 1 Algoritmo per l'assegnazione dei timestamp ai nodi del grafo

```

for b in blocks do S = sottografo corrispondente ad un blocco b
  esegue il metodo label_llc(S)
  interval = intervallo di tempo del blocco
  paths = chain in ordine decrescente in base alla lunghezza

  for p in paths do per ogni nodo

    if il primo elemento di p non ha predecessori then assegna il timestamp
      alla chain

      for j in p[1] do salva il nodo con il timestamp in un dizionario
      end for
      else se il nodo ha un predecessore presente nel dizionario
        calcolo il timestamp con istante iniziale pari a quella del predecessore
      end if
    end for
  end for

```

Nell'algoritmo 1 si nota come la lista path, corrisponda alla lista di tutte le chain del grafo. In particolare tali chain sono le componenti connesse del grafo e potrebbero essere composte da tante transazioni oppure solo da un nodo. Infatti per classificare le catene in base alla loro lunghezza si è deciso di utilizzare un metodo che calcola il valore **LLC**, ovvero la lunghezza della catena più lunga di

appartenenza. In parole povere, il valore llc rappresenta il cammino più lungo che passa attraverso quel nodo. Tale algoritmo per il calcolo dell'LLC, viene ripreso dal paper [DDP17], e viene riportato in questo paragrafo in *figura 4.1*.

Algorithm 1 Label nodes with their LLC

```

1: procedure LABELNODES(txGraph)
2:   Label each source node of txGraph with attribute backward initialized to 0
3:   Label each sink node of txGraph with attribute forward initialized to 0
4:   sortedNodes  $\leftarrow$  topologicalSort(txGraph)
5:   for n in sortedNodes do
6:     if predecessors(n).length  $\neq$  0 then
7:       bPredecessors  $\leftarrow$  List of backward attributes of all predecessors of n
8:       txGraph.getNode(n).backward  $\leftarrow$  max(bPredecessors) + 1
9:     for n in reverse(sortedNodes) do
10:      if successors(n).length  $\neq$  0 then
11:        fSuccessors  $\leftarrow$  List of forward attributes of all successors of n
12:        txGraph.getNode(n).forward  $\leftarrow$  max(fSuccessors) + 1
13:      for n in txGraph.nodes() do
14:        n.LLC  $\leftarrow$  sum(n.forward + n.backward)

```

Figura 4.1: *Algoritmo assegnazione LLC* [DDP17]

Il campo LLC, quindi serve per ordinare i vari cammini dei nodi, per assegnare i timestamp. Tale algoritmo si serve di due attributi: b, e f. "b" corrisponde all'attributo *backward* che rappresenta il numero dei predecessori, "f" corrisponde all'attributo *forward* che indica invece il numero dei successori. Visitando prima il grafo in "avanti" si etichettano i nodi secondo l'attributo b. Visitandolo poi a ritroso, si riescono a calcolare tutti gli attributi f. Infine, per ogni nodo si sommano gli attributi b e f e si ottiene il numero LLC. Ovviamente questo processo è reso possibile se il grafo viene visitato in ordine topologico. Tale ordine indica per primi i nodi che non hanno archi entranti, poi vengono visitati i nodi indicati dagli archi uscenti. In poche parole l'ordinamento topologico guida una visita dal nodo "sorgente" al nodo "destinazione" passando per tutti i nodi del grafo.

4.3.3 Dettagli dell'algoritmo per il calcolo del cammino più lungo a varianza minima

La parte principale del progetto, si basa sull'algoritmo per il calcolo del cammino più lungo a varianza minima. Tale algoritmo, è stato pensato ed implementato per cercare di capire quali sono le transazioni che vengono generate automaticamente e quali no.

Come è stato già sottolineato in precedenza, una catena automatica di transazioni, viene generata a velocità e frequenza elevati. Contemporaneamente, una chain creata tramite un software, seguirà un andamento sempre costante. Grazie a questa caratteristica, si può quindi pensare di utilizzare la nozione di varianza, che misura quanto un evento si discosta dal suo andamento medio. Infatti, cercando di capire quali sono le catene più lunghe a varianza minima, si potrebbero trovare alcuni riscontri per quanto riguarda gli andamenti automatici.

L'algoritmo in questione, visita il grafo in ordine topologico, ossia dai nodi sorgenti ai quelli target, e visita ogni nodo. Durante questa visita iterativa, la prima cosa che viene fatta è quella di calcolare il cammino a varianza minima che porta al nodo che si sta considerando. Ovvero la sequenza di nodi che arriva al nodo corrente, che non soddisfa le condizioni di varianza minima.

Algorithm 2 Algoritmo per il cammino a varianza minima

```

for n in topological_sort(G.nodes()) do
    path = get_path(G,n)
    if path.length() == 0 then
        freq(n) = 0
        v(n) = 0
    else
        f_ist(n) = calcola freq istantanea del nodo
        freq(n) = calcola freq media del path
        v(n) = calcola varianza del path
    end if
end for

```

Per ogni nodo del grafo in ordine topologico, viene calcolata la variabile *path*

che rappresenta il cammino a varianza minima. Se il path ha lunghezza uguale a 0, vuol dire che n è il nodo iniziale, e gli vengono assegnati i valori iniziali di frequenza e varianza. Altrimenti, viene calcolata la frequenza istantanea del nodo rispetto al predecessore, poi viene calcolata la frequenza media di tutto il path, e infine la varianza, che si serve dei singoli valori di frequenza istantanea. Ogni volta, la variabile path viene salvata in un oggetto temporaneo che rappresenta il path che fino a quel momento presenta i valori di lunghezza massimo e varianza minima. Alla fine dell'algoritmo, il path viene aggiornato se e solo se la sua lunghezza è maggiore di quello precedente e la varianza è minore rispetto a quella del path precedente.

In questo metodo, l'ordinamento topologico è garantito dai metodi di base della libreria Networkx di python.

Il metodo *get_path* che interviene nell'algoritmo 2, prende come input il grafo, il nodo e ritorna il cammino a varianza minima e viene mostrato nell'algoritmo 3.

Algorithm 3 get_path(G,n)

```

function GET_PATH(graph G,node n)
    return get_path_ric(G,n,[])
end function

function GET_PATH_RIC(G,n,path)
    if not G.has_node(n) then
        return []
    else
        path.append(n)
        pred = get_pred(G,n)
        temp = path con var min tra i predecessori
        return [temp] + get_path_ric(G,temp,path, lenp-1)
    end if
end function

```

L'algoritmo 3 si serve di una funzione ricorsiva per calcolare il path, poichè di norma nella visita dei grafi si utilizzano metodi ricorsivi come la DFS("Depth

First Search"). Alla fine, la funzione ritornerà il path corrispondente.

Per quanto riguarda invece il calcolo della frequenza istantanea e della varianza, gli algoritmi sono mostrati nelle figure 4 e 5. La frequenza media invece, viene semplicemente calcolata come media tra i diversi valori di frequenze istantanee, direttamente nell'algoritmo get_path.

Algorithm 4 Frequenza istantanea

```
function CALCULATE_F_IST(tsn, tsnprev)
  if tsn - tsnprev == 0 then:
    return 0
  end if
  return 1/(tsn-tsnprev)
end function
```

Algorithm 5 Varianza

```
function CALCULATE_V(G,path,n)
  for i in path do
    v += math.pow(freq(i),2)
  end for
  return v
end function
```

Negli algoritmi precedenti viene mostrato il calcolo dei due parametri chiave dell'algoritmo principale.

Successivamente, è stata effettuata una modifica a tali algoritmi, specialmente al 3. Infatti, dal momento che vengono analizzati tantissimi dati, si rischia di incappare in catene lunghissime che difficilmente si riescono a gestire. Dal momento che l'obiettivo principale è di calcolare la varianza di tale catena, allora si è pensato di effettuare tale calcolo sugli ultimi p elementi del path. Infatti, i primi elementi normalmente hanno valore di frequenza e varianza uguale all'inizializzazione, perciò andrebbero ad influenzare in modo negativo l'analisi finale. Perciò, prendendo man mano in considerazione gli ultimi p nodi, si otterranno dei valori di varianza e di frequenza molto più realistici.

Per comodità, quindi viene passato come parametro anche tale valore **p**, e viene arbitrariamente inizializzato a p=100.

Alla fine dell'esecuzione di tutti questi algoritmi, si ottiene un grafo dove per ogni nodo si hanno gli attributi di frequenza e di varianza, calcolati sui p nodi predecessori.

Capitolo 5

Sperimentazione & Risultati

5.1 Grafi di test

Dopo l'implementazione degli algoritmi descritta nel capitolo 4, è stato fatto un test con alcuni grafi arbitrariamente costruiti. Come prima cosa, vengono generati manualmente 4 grafi che rappresentano alcuni degli schemi di catene che potrebbero verificarsi all'interno del tx-graph.

Il primo schema rappresenta il classico cammino dritto, e si può notare in figura 5.1. All'esecuzione dell'algoritmo, viene stampato il path = [0,1,2,3,...,9]. Nella figura 5.2 si può notare un grafo a forma di Y, ovvero due cammini che con-

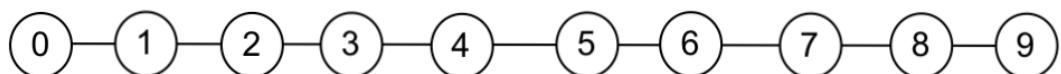


Figura 5.1: *Un cammino*

vergono sullo stesso nodo. Potrebbe essere un classico esempio di una situazione di "merge". Nella fase di test su questo tipo di grafo, tra il nodo 2 e il nodo 5, il nodo 2 ha varianza minore, e il path restituito dall'algoritmo corrisponde proprio a [0,1,2,6,7,8,9].

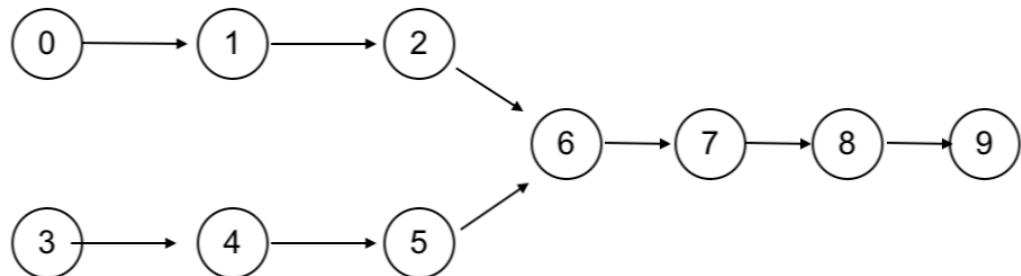


Figura 5.2: Un Y grafo

Nella figura 5.3 si ha un nuovo schema di grafo a forma di X. Se si esegue l'algoritmo anche su questo, a parità di varianza dei cammini, si otterrà come risultato il path [0,1,2,3,4,5], che è il più lungo.

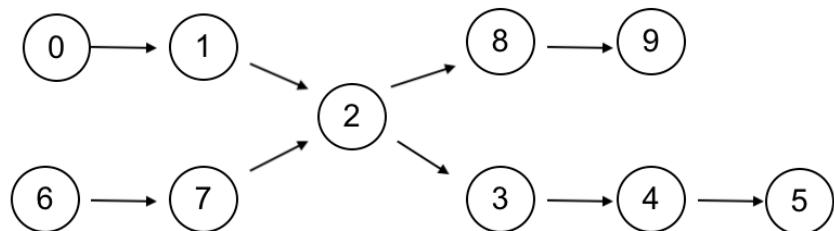


Figura 5.3: Un X grafo

Per quanto l'algoritmo del calcolo del cammino più lungo a varianza minima si corretto, a parità di cammini di eguale lunghezza ed egual varianza, stampa solamente il primo di essi, tralasciando gli altri cammini equivalenti.

5.2 MatPlotLib per i grafici

Dopo l'esecuzione dell'algoritmo, quindi, ogni nodo viene etichettato con attributi di frequenza(sia istantanea che media) e varianza. La frequenza istantanea non è molto rilevante per quanto riguarda lo studio sul tx-graph.

Perciò si è deciso di graficare la distribuzione di tali valori su dei grafici con istogrammi, fatti con una libreria di python chiamata **MathPlotLib**. I grafici ottenuti rappresentano, sull'asse x la distribuzione dei singoli valori di frequenza(o varianza) dei nodi, e sull'asse y quanti sono quei nodi che hanno tale valore di frequenza(o varianza).

5.3 Statistiche & Grafici

Inizialmente si è voluto graficare il primo dataset D1(417113, 417256), perché è quello che più semplicemente si presta a generare un grafico attendibile. Nella figura 5.4 si può notare l'istogramma del primo dataset.

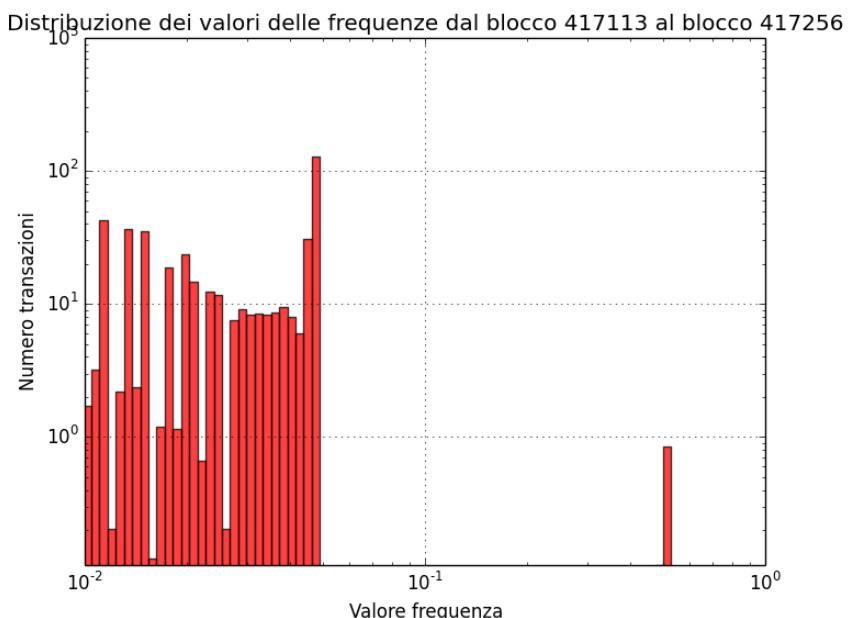


Figura 5.4: $G1(417113, 417256)$

5.3.1 Al variare del Dataset

Prendendo successivamente in considerazione il dataset D2(413000, 414000), e di conseguenza il grafo generato da tale dataset, si è preferito creare dei grafici al variare della dimensione del dataset, per vedere quali erano i valori che si sarebbero aggiunti alla distribuzione.

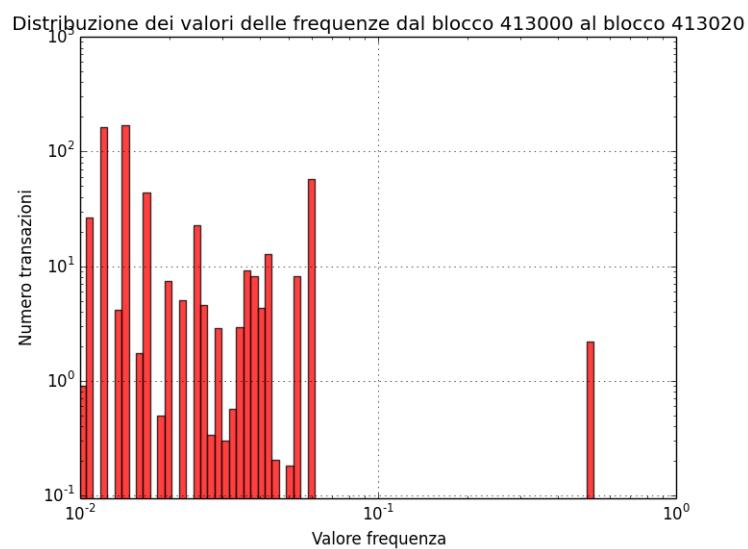


Figura 5.5: 20 blocchi

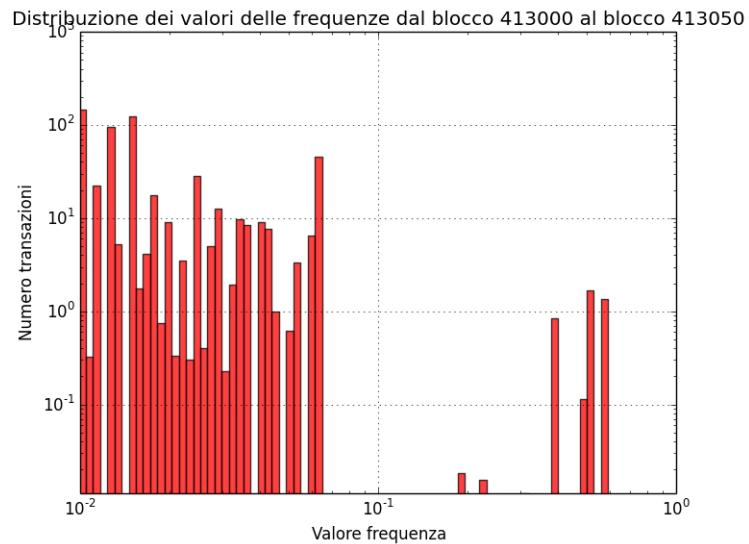


Figura 5.6: 50 blocchi

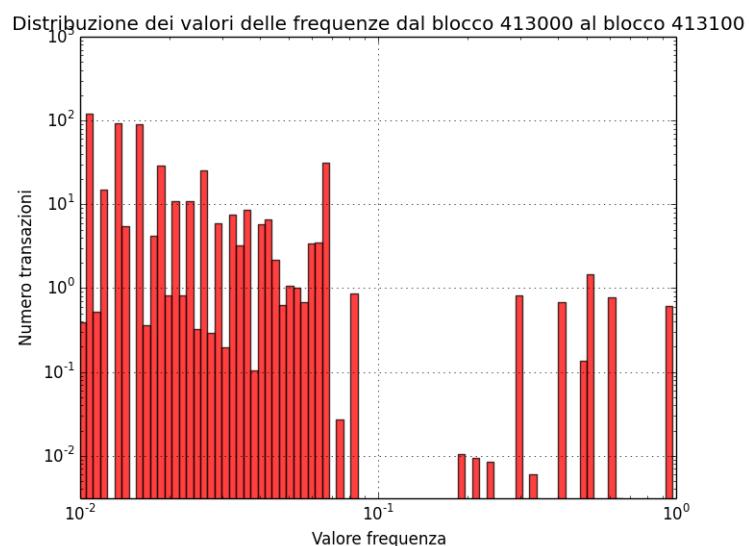


Figura 5.7: 100 blocchi

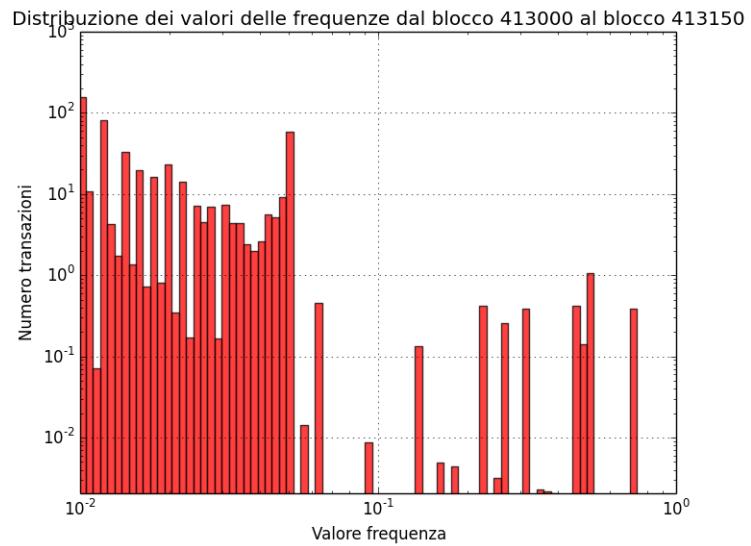


Figura 5.8: 150 blocchi

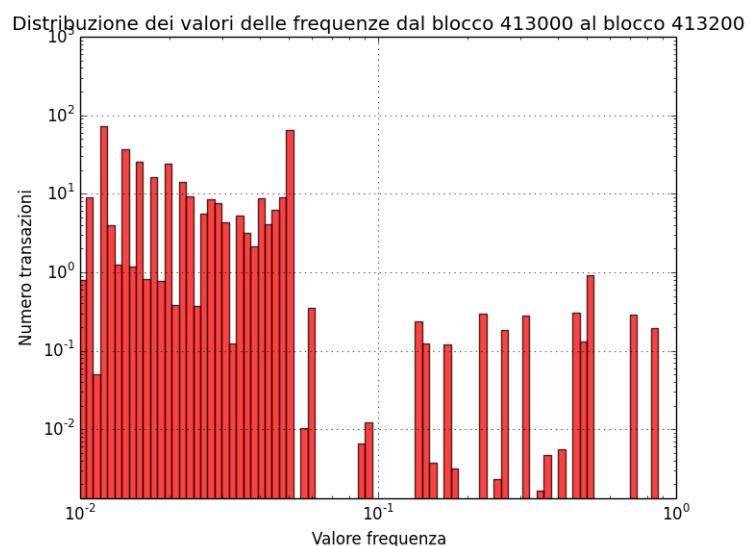


Figura 5.9: 200 blocchi

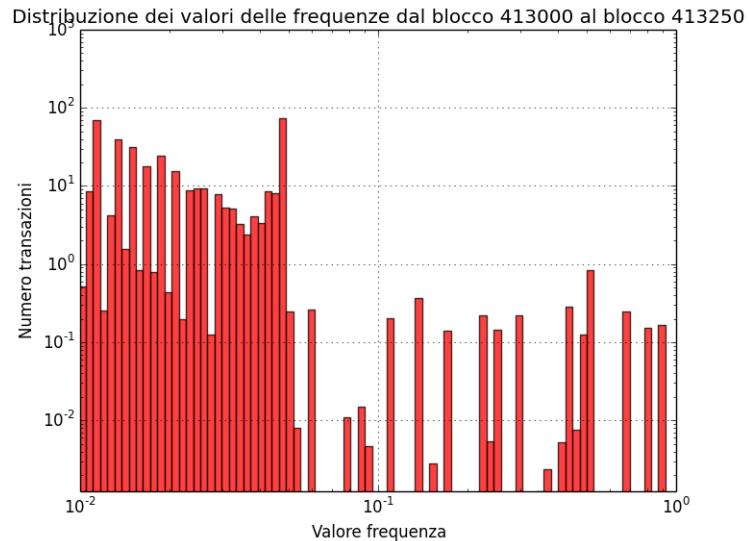


Figura 5.10: 250 blocchi

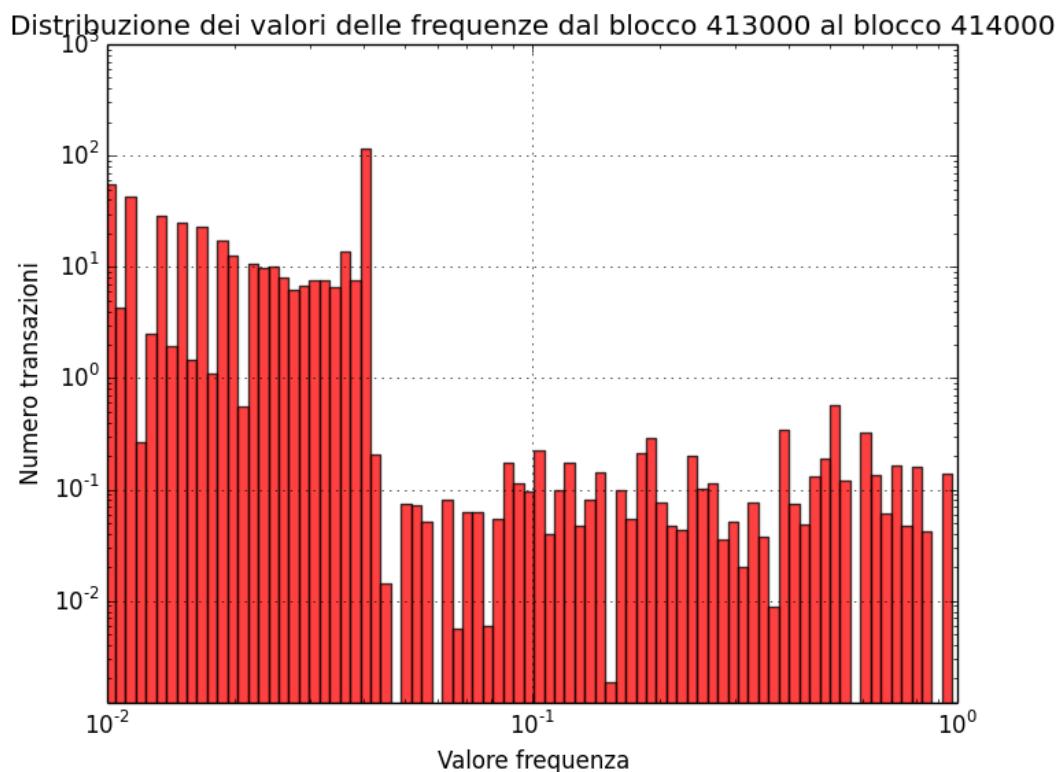


Figura 5.11: 1000 blocchi

Conclusioni e sviluppi futuri

Bibliografia

- [Ant14] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies.* " O'Reilly Media, Inc.", 2014.
- [Blo] Blockchain.info. Bitcoin block explorer - blockchain.
- [DDP17] Giuseppe Di Battista, Valentino Di Donato, and Maurizio Pizzonia. Long transaction chains and the bitcoin heartbeat. In *Workshop on Large Scale Distributed Virtual Environments (LSDVE 2017)*, 2017. To Appear.
- [Mix] Coin Mixer. Coin mixer - <https://coinmixer.se>. <https://coinmixer.se/it/>.
- [Mös13] Malte Möser. Anonymity of bitcoin transactions. In *Münster bitcoin conference*, pages 17–18, 2013.
- [Wik11a] Wikipedia. Bitcoin — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].
- [Wik11b] Wikipedia. Tor(software) — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].
- [Wik11c] Wikipedia. Varianza — wikipedia, l'enciclopedia libera, 2011. [Online; controllata il 9-aprile-2011].