Deep Graph Kernels for Inferring Bitcoin Transaction Dynamics

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF SCIENCE
IN
DATA ANALYTICS

by

PANKAJ KUMAR
(1410120079)

SHIV NADAR UNIVERSITY

BIG DATA ANALYTICS CENTER

SCHOOL OF NATURAL SCIENCES

SHIV NADAR UNIVERSITY, GAUTAM BUDH NAGAR,

U.P., INDIA

MAY, 2016

SHIV NADAR UNIVERSITY

BIG DATA ANALYTICS CENTER

SCHOOL OF NATURAL SCIENCES

SHIV NADAR UNIVERSITY, GAUTAM BUDH NAGAR,

U.P., INDIA

CERTIFICATE

This is to certify that the present work entitled, "**Deep Graph Kernels for Inferring Bitcoin Transaction Dynamics**" submitted by **Pankaj Kumar** to the Big Data Analytics Center, Shiv Nadar University, in partial fulfilment of the requirements for the award of the degree of Master of Science is approved.

....
Dr. Sudeepto Bhattacharya
Associate Professor
Department of Mathematics
Shiv Nadar University

....
Dr. Santosh Singh
(Head, Big Data Analytics Center)
Associate Professor
Department of Mathematics
Shiv Nadar University

....
Dr. Anatoly Dymarsky
Assistant Professor,
Skoltech, Russian Federation

....
Dr. Panagiotis Karras
Assistant Professor
Skoltech, Russian Federation

SHIV NADAR UNIVERSITY

BIG DATA ANALYTICS CENTER

SCHOOL OF NATURAL SCIENCES

SHIV NADAR UNIVERSITY, GAUTAM BUDH NAGAR,

U.P., INDIA

DECLARATION

I hereby declare that the thesis titled, "**Deep Graph Kernels for Inferring Bitcoin Transaction Dynamics**" being submitted to the Big Data Analytics Center, School of Natural Sciences, Shiv Nadar University, in partial fulfillment of the requirements for the award of the degree of Master of Sceience is a record of bonafide work carried out by me.

The matter embodied in the thesis has not been submitted for the award of any degree in any university or institution.

MAY, 2016	Pankaj Kumar
Gautam Budh Nagar, Uttar Pradesh	(1410120079)

Acknowledgments

There are no proper words to convey my deep gratitude and respect for my thesis and research advisor, Dr. Sudeepto Bhattacharya, Shiv Nadar University, India. With his course in graph theory and complex network, he not only inculcated my interest in graph theory, but has inspired me to become an independent researcher. He helped me realize the power of critical reasoning, simple, but concrete mathematical representation of idea and guidance to recover when my steps faltered.

I would like to express my gratitude to my external supervisor at Skolkovo Institute of Science and Technology, Moscow, Russia Federation, Dr. Anatoly Dymarsky and Dr. Panagiotis Karras, whose expertise, understanding, and patience, added considerably to my thesis experience. They taught me how to critically question thoughts and express ideas in simple but scientific way. Their insightful thought-provoking comments and constructive criticisms at different stages of my research allowed me to stay focused to core of research ideas. I am grateful to them for holding me to a high research standard and enforcing strict validations for each research result, and thus teaching me how to do research.

My sincere thanks must also go Dr. George Ovchinnikov, without whose motivation and encouragement I would not have considered working on Bitcoin and graph kernels. It was under his tutelage that I developed a focus and became interested in science behind Bitcoin. He provided me with direction, technical support and became more of a mentor and friend, than a guide. It was though his, persistence, understanding and kindness that I completed my master thesis in utter joy. I doubt that I will ever be able to convey my appreciation fully, but I owe him my eternal gratitude.

There is no way to express how much it meant to me to have been a member of Center for Computational and Data-Intensive Science and Engineering Lab for all meaningful discussion on different plethora of research topics. The brilliant friends, colleagues and and all the other current and former Lab graduates students and visitors that I know inspired me over my stay in Moscow. Also, thanks to active communities of Bitcoin, R, Gephi and Python.

I would like to thank all the people associated with Big Data Analytics Center, Shiv Nadar University, fellow students and teachers who have been source of thoughts, inspiration and smiles. Acknowledgement will be incomplete without expressing sincere gratitude to Dr. Santosh Singh for providing excellent infrastructure, guidance and scholastic path to realize our dreams into reality.

Though only some name appears on the cover of this thesis, a great many people have contributed to its production. I owe my deep gratitude to all those people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

I deeply thank my family for their unconditional trust, timely encouragement, and endless patience. It was their love that raised me up again when I got weary.

In conclusion, I recognize that this research would not have been possible without the financial assistance of Skolkovo Institute of Science and Technology, Moscow, Russian Federation and Shiv Nadar University, India (Teaching Assistantships, Visiting Research Scholarships) , and express my gratitude to those agencies.

Pankaj Kumar

Abstract

Bubbles have fascinated and baffled many canny observers of financial markets. The extreme fluctuations in Bitcoin (BTC/USD) price, hence bubbles, has created interest among media, regulators and researchers to understand underlying cause. The extreme price volatility demands a thorough investigation of transactions network in bitcoin. With readily available blockchain, log of all transactions that were ever verified on the Bitcoin network, we attempt to investigate how structural changes in the network accompany significant changes in the exchange price of bitcoins. The results of this investigation is baseline for understanding bitcoin price formation or forecasting BTC/USD exchange price.

In big data setting, we are trying to understand "How much is a large-scale graph (bitcoin transaction graph) transformed over time or by a significant event (bubbles)?" or "How structurally similar are two large-scale graph (bitcoin transactions graphs)?". In application coming out from dynamic graph, similarity in graph connectivity is great tool to detect anomalies in the bitcoin network that are manifest in the form of bubbles.

Taking help of graph kernels, a polynomial alternative of graph isomorphism problem, we find an algorithm to calculate the similarity index $(0-1)$ between the two graphs. This is quantitative measure of transformation over time. By extending the quite novel framework for graph kernels inspired by latest advancements in natural language processing and deep learning, we propose, deep graph propagation kernels. The unseen deep framework in the literature takes account of attributed graphs with continuous values.The deep graph kernels outperforms its best base graph kernels variants in terms of capturing correlation between newtork structure and market price. This defines a baseline to predict the price of BTC/USD exchange, which leverage on deep learning learning framework to extract feature space of blockchain.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Bubbles have fascinated and baffled many canny observers of financial markets. In the language of the strict orthodoxy of the efficient market theory, bubbles are outcome of sudden changes of the fundamental valuation of assets. The empirical evidence denounce that large price jumps can be explained by news, but destabilising feedback loops of behavioural origin [3]. Although plausible, a clear-cut empirical evidence for such a scenario is still lacking, but it had created enthusiasms among research communities to understand the origin of bubbles in different commodity, Bitcoin is one of them.

Bitcoin is a digital, distributed, cryptographic currency proposed under the pseudonym Satoshi Nakamoto [5]. In short, It can be described as a decentralized accounting system in which accounts are associated with public keys of an asymmetric encryption scheme. To access the former account, knowledge of the corresponding private key allows account holders to create digital signatures to send and receive bitcoins (BTC).

1.1 Bitcoin Bubble

Bitcoin started trading on Mt.Gox (its market share was over 80% on the BTC/USD spot market), the largest online Bitcoin exchange at 5 cents. Over the next two years, BTC exchange rose slowly, but in early April 2013, the price per BTC dropped from \$266 to around \$50 and then rose to around \$100. This major bubble was succeeded by another one on 29 Nov, 2013, when, the price of a bitcoin reached an all-time high of US\$1224.76, then dropped into the \$200-\$300 range. Metaphorically, the price rise/drop is expressed as skippyball. Once skippyball pitched high in the air, the first bounce is big, then lower, and lower, until it is flat on the ground, ready for a new throw upward. Taking market price data from Blockchain the most popular Bitcoin wallet [1], the bubbles can be visualized in the time series plot 1.1 with circles representing bubbles.

[1]Market Price in USD `https://blockchain.info/charts/market-price`

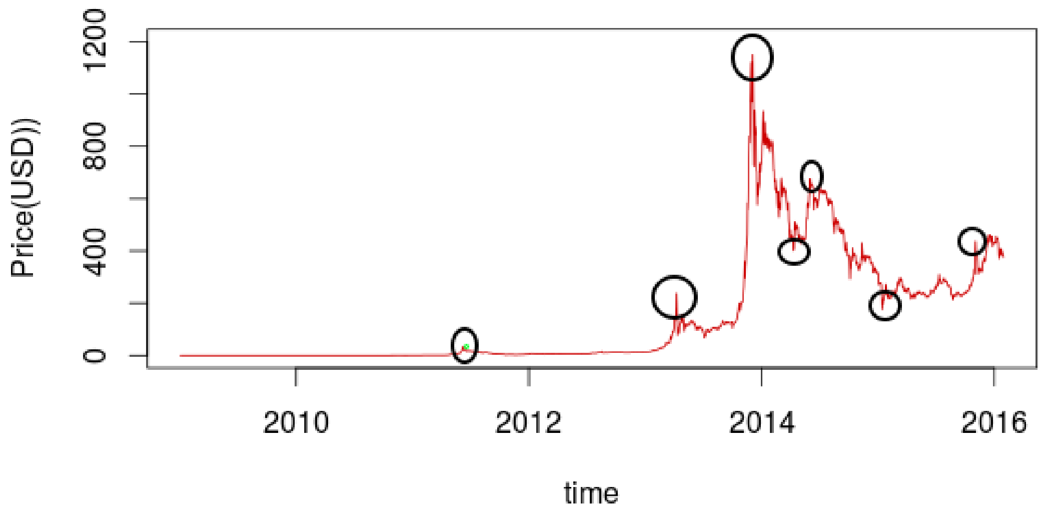


Figure 1.1: BTC Market Price (USD)

As the Bitcoin's complete transaction history is stored in an easily accessible and verifiable public ledger called blockchain, It is quite popular amongst economist to study bubbles. The most popular economic explanation [1, 2] suggests that Bitcoin bubble in April, 2013, was outcome of the financial crisis in Cyprus, which triggered large numbers of people to convert euros into digital BTC. While Bouchaud and Donier [3] believe that bubbles were conditioned by the market liquidity, which triggered mismatch between the aggregate market order flow imbalance that becomes strongly negative and the prevailing liquidity on the buy side. In simple words, because of high price, buyers got too scarce to resist the pressure of a sell-off.

The complex, cryptic and enormous(over 60 million transactions) data-structure of blockchain had limited researchers to do in-depth transaction analysis even in computer science field. Kondor et al. [6] reconstructed the transaction network between users and analyze changes in the structure of the subgraph induced by the most active users. By using unsupervised identification of important features of the time variation of the network and Principal Component Analysis to the matrix constructed from snapshots of the network at different times, they were able to show that structural changes in the network accompany significant changes in the exchange price of bitcoins. The one fall-back about this paper is scalability, which makes it unsuitable for application having nodes greater than 10000 nodes. Other studies [2] uses digital behavioral traces (eg. twitter etc), wavelet coherence analysis and logistic regression to investigates bubbles, which is not a refine method considering the availability of data [4].

1.2 Graph Similarity and Subgraph Matching

The ultimate goal in financial market is predicting long-term asset prices, as they are so difficult to predict, Bitcoin prices are no different. The margin error with Bitcoin can be particularly brutal for extreme predictions, which usually have a lower probability of being realized. While there has been significant research done to analyze Bitcoin transaction network, limited research has been executed to analyze the network's influence on overall Bitcoin price. To achieve the former, one need to efficiently measures the structural change (eg. walks) of a dynamic large-scale graph as well as the similarity between two graphs with reference to price change. Kondor et al. [6] research was one step in the direction of basic graph similarity and subgraph matching algorithms, which is not scalable with increasing size of daily transactions 1.2.

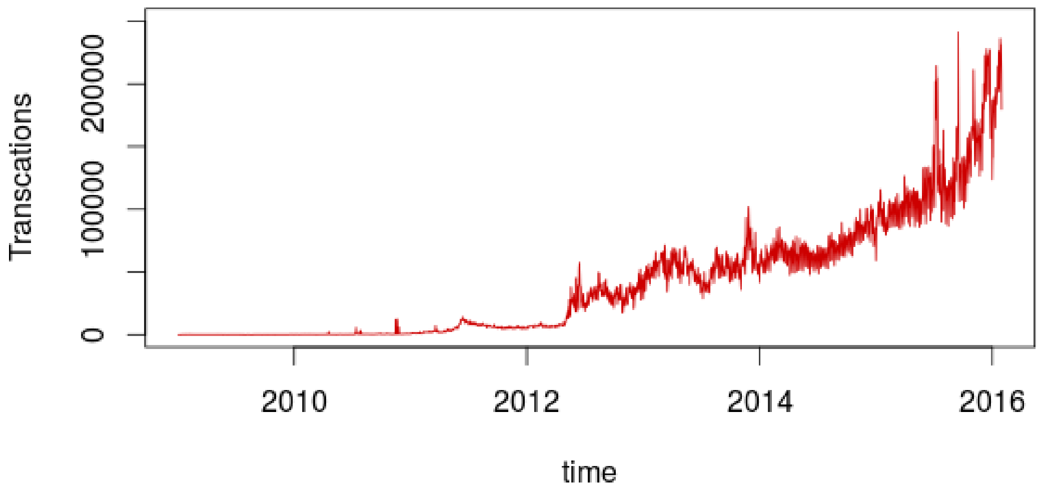


Figure 1.2: Daily Transactions

Most of graph isomorphism algorithms and its generalizations are exponential and, thus, not applicable to the large graphs that are of interest today. The other methods for graph similarity and graph sub-matching, like features extractions, iterative methods and tensors analysis, including some come with their drawbacks [7].

In the recent paper on graph kernels, Yanardag and Vishwanathan [8] used "Deep Graph Kernels", a unified framework to learn latent representations of sub-structures for graphs to measure similarity. As Bitcoin transaction network graph is increasing everyday, It would be interesting to capture its structural dynamics without compromising on the graph features.

1.3 Motivation

The volatile rise and fall of Bitcoin, cryptological mining, complex distributed database, blockchain and evolving network dynamics prompted me to efficiently measures the structural change (eg. walks) of a dynamic large-scale graph as well as the similarity between two graphs with reference to price change.

In simple words, the thesis attempts to address two important questions:

1. How much is a large-scale graph (bitcoin transaction graph) transformed over time or by a significant event (bubbles)?
2. How structurally similar are two large-scale graph (bitcoin transactions graphs)?

1.4 Objective

1. Given complex cryptic distributed blockchain, parsing it to extract transaction attributes in the form transactions graphs.
2. Given two graphs $G_1^{t_1}(n_1, e_1)$ and $G_2^{t_2}(n_2, e_2)$, find an algorithm to calculate the similarity index $(0 - 1)$ between the two graphs. This is quantitative measure of transformation over time.
3. Given a graph time series, where there are T number of graphs, find approximate subgraphs belong to particular events (says bubbles -10-15% rise/fall in BTC/USD) that occur in a subset of the T graphs.

1.5 Contribution

The decentralized paradigm of Bitcoin requires each node of the network to retain the blockchain, which consists of all public, transparent, and permanently transactions recorded since the origin. Therefore, blockchain acts as the central point of potentially interesting information ready to be mined. But ever growing size of blockchain (60 GB as of now) makes the job difficult to parse from the raw blockchain. Adding to the same, the newer bitcoin clients indexed the full blockchain using LevelDB makes the earlier public available software obsolete. This thesis aims to develop open source blockchain parsing tools to extract agent resolved data, which can be used to extend the stucked research in bitcoin transaction dynamics.

To understand how much a large-scale graph (bitcoin transaction graph) transformed over time or by a significant event (bubbles), we use quite novel framework for graph kernels inspired by latest advancements in natural language processing and deep learning, ”deep graph propagation kernels” to have quantitative measure of transformation, which captures correlation between network structure and market price. It outperforms its best base variants in terms of capturing .

1.6 Thesis Organization

The rest of this thesis is organised as follows. In Chapter 2, we introduces Bitcoin, its working in detail by explaining the protocol of blockchain. Then, the data preparation staged is discussed to get the transaction network, which is used to reproduce transaction dynamics in terms of money flow, also a data validation method. With data in hand, Chapter 3 motivates to the graph isomorphism in our context, then reviews the important studies in graph isomorphism problem with advantages and disadvantages. Chapter 4 starts with review of representative graph kernels in the literature. By taking conventional and sophisticated graph kernels, this chapter defines quantitative measure of transformation over time. The chapter concludes with not so smooth correspondence between network structure and exchange price in bitcoin, thus paving way for further investigation to other graph kernels. With graph kernels not giving desired results, Chapter 5 propose a general framework that learns hidden representations of sub-structures used in graph kernels, inspired by deep graph kernels. Then, the framework is demonstrated on propagation kernels, which performs better than normal kernels. Then new graph kernel is used to calculate similarity index, which is plotted to find correlation between network structure and market price. Chapter 6 concludes the thesis.

Chapter 2

Bitcoin Transaction Dynamics

2.1 Introduction

Bitcoin is a decentralized peer to peer electronic payment system in which transactions are performed with no central authority or banks to authorize it. The Bitcoin transactions management and its issuance is carried out collectively by the network. The first Bitcoin specification and proof of concept was published by Satoshi Nakamoto [5] in 2009 cryptographic mailing list. Bitcoin can also be seen as the most prominent triple entry bookkeeping system in existence. Since then, the community has grown exponentially with many developers working on Bitcoin.

2.2 Bitcoin

Bitcoin payments use public keys encryption, where, payers and payees are identified by hashed public keys of their Bitcoin wallets. The public keys are generated by ECDSA (Elliptic Curve Digital Signature Algorithm), based on calculations of elliptical curves over finite space. Suppose, Alice sends "x" coins to Bob, then an unencrypted transaction attaching Bob's public key is broadcast over the Bitcoin network using her private key. The signature on transaction verifies all users for its authenticity (current owner of coin) by looking at complete history of transactions, called *blockchain*. It is essential element in bitcoin architecture, which verifies the legitimacy of ability (sufficient bitcoins) of Alice paying Bob.

2.2.1 Blockchain

The complete record of transactions is in a coded form in a data structure called *blockchain*, which is a sequence of records called *blocks*. Each block contains a group of transactions that have been sent since the previous block, with integrity check all the way back to the first one, the genesis block.

Any users enter the Bitcoin system by trading non-digital currencies at Bitcoin market exchange, or by mining coins, which involves solving a cryptographically hard problem for which he/she is suitably rewarded a fixed number of Bitcoins and transactions is validated on the network. The working of mining, core to *blockchain*, is mathematically explained better by Johannes et al. [9]. The network of peers, called miners, are the agents using computers, who actually add the blocks to the blockchain. Taking example of Alice sending "x" coins to Bob from section 2.2 by broadcasting over network. The miners receives copies of all transactions as they are generated, including Alice's copy. The blockchain is examined to investigate the history of the bitcoins involved in each transaction. If the proposed transaction from the Alice has sufficient bitcoin credit, then it is accepted for incorporation into the block that the miner is currently working on.

Identifying each transaction with a double SHA-256 hash, the transactions is gathered together. Then by using miners hashes,together with the hash that is at the current head of the blockchain, as inputs to the cryptographic problem is solved by miners, with rewards of 25 bitcoins in 10 minutes [9]. At first, miner M computes a block hash h over a unique ordering of the hashes of all the transactions that it is intending to incorporate into its next block B. Then, input of the block solution s_{i-1} is taken at the head of its current version of the *blockchain*. If we concatenate the strings by the symbol +, the cryptographic problem that M has to solve is: compute a SHA-256 hash

$$s_i = (n + h + s_{i-1}), \tag{2.1}$$

such that s_i has at least a specified number x of leading zeros where $x \sim 64$.

Once mined, the new block is communicated by broadcasting newly-discovered blocks via a peer-to-peer network to add the new block at blockchain at each peer using blockchain protocol rules. This makes blockchain as a public ledger, recording every Bitcoin payment ever made. The simplified version of the whole process is represented in below illustration 2.1.

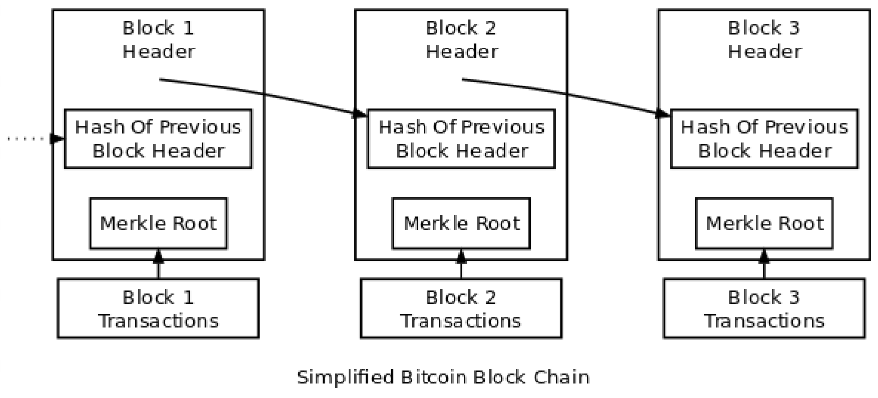


FIGURE 2.1: Simple Blockchain [10]

The transaction data part of a block consists of one or more new transactions. While the merkel root of the merkle tree contains hashed copies of each transaction, and the hashes are then paired, hashed, paired again, and hashed again until a single hash remains. The chaining of blocks together and storing the hash of the previous block's header ensures a transaction cannot be modified without modifying the block that records it and all following blocks. The same goes with the transaction, which are also chained together. A single transaction can create multiple outputs, which are tied to transaction identifiers (TXIDs), the hashes of signed transactions. Also, the outputs of all transactions included in the block chain can be categorized as either Unspent Transaction Outputs (UTXOs) or spent transaction outputs 2.2. The first one of these transactions called a generation transaction, which should collect and spend the block reward (comprised of a block subsidy and any transaction fees paid by transactions included in this block). All these transactions are encoded into blocks in binary rawtransaction format, making it difficult for researchers to extract in simple relational database format, to get transaction graph.

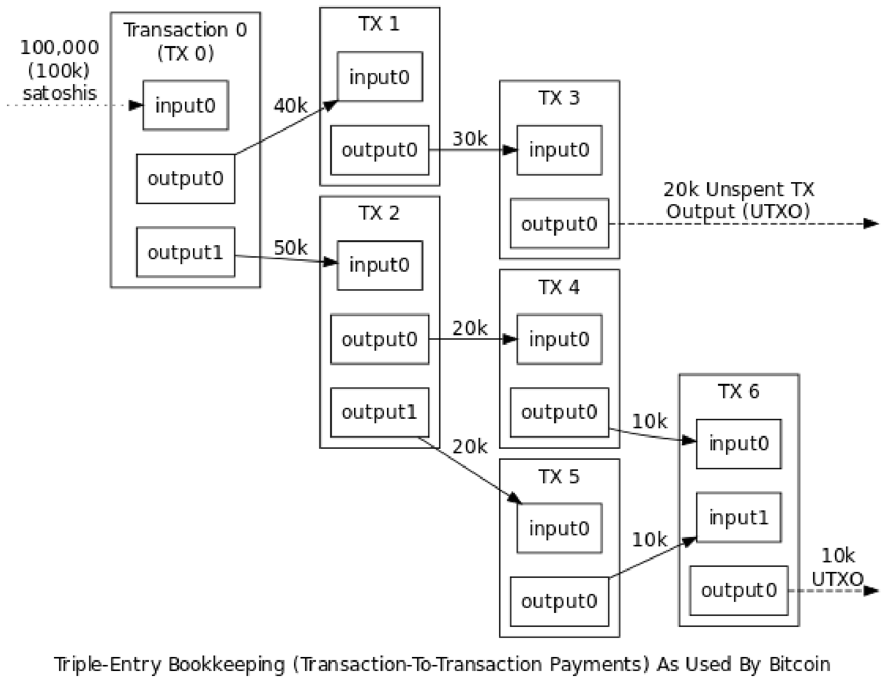


FIGURE 2.2: Blockchain Transaction [10]

2.3 Related Work

The possibility to analyze agent resolved transactions in any market is limited by the scarcity of available data, as this kind of information is usually considered highly sensitive. But with the publicly available blockchain at every machine allows researchers to reconstruct the network of transactions and extract the time and amount of each payment. There are many research studies that concerns bitcoin transaction network using network analysis, machine learning and statistical physics techniques.

Parsing four (2009-2013) years data from bitcoin's blockchain, Kondor et al. [13] analyze the structure of the transaction network by measuring network characteristics over time, such as the degree distribution, degree correlations, clustering and money movement. Using the same data, they in their different paper [6] analyze changes in the structure of the subgraph induced by the most active users. They were able to show find correlation between structural changes in the network and exchange price of bitcoins. But with exponential increase in the number of transaction over years, they method seems obsolete , as it is difficult to scale with their endorsed methods.

Reid and Harrigan [12] linked addresses belonging to the same entity using external information. The using techniques such as context discovery and flow analysis, they investigate an alleged theft of Bitcoins. By running the Union-Find algorithm, Ron and Samir [11] associated the 3,120,948 addresses with 1,851,544 different entities to understand behaviour of users. Spagnuolo [14] released open source project Bitiodine, which was able to cluster addresses and classify them using a dataset partially obtained in an automatic fashion, using scrapers for major web sources of bitcoin addresses. On the similar line, other paper [15] investigates bitcoin transaction-graph-annotation, which is capable of tracing and clustering user activity.

Most of the previous research employed data collected from old Bitcoin- client Bitcoin 0.8 later, the newer bitcoin clients indexed the full blockchain using LevelDB instead making the publicly available bitcointools obsolete. Adding to the same, the nearing 300,000 transactions/day in 2015 gives really hard time to earlier algorithms to scale, thus motivating us to develop new techniques to parsing data from blockchain, reconstructing network and doing analysis.

2.4 Data

The blockchain is a transaction database of the Bitcoin. Once Bitcoin core is setup at local machine, blockchain is automatically downloaded. Every full node participating in the Bitcoin network has the same copy. As of now there is more than 60 GB of Bitcoin blockchain dataset, which makes it difficult to parse the raw blockchain data. Most of the previous studies [11] employed a forked version of bitcointools [1], but from the bitcoin clients 0.8 version, it indexed the full blockchain using LevelDB instead making the publicly available bitcointools obsolete. Other well know open source blockchain parser like blockparser [2], BitIodine [14] etc are almost undocumented projects, where some appear to not even work. A paper [15]

[1]`https://github.com/gavinandresen/bitcointools`
[2]`https://github.com/znort987/blockparser`

led me to BitcoinArmory project, which requires a dozen of manual interventions to get installed, but still doesn't work.

We then started to look for ready-made SQL database, which pointed to BitcoinABe. This python based library reads the Bitcoin block file, transforms and loads the data into a SQL database A.1. But, Its takes more than two weeks to dump the data, which was not feasible option. We downloaded postgres database dumps of the bitcoin-ruby-blockchain database generated by webbtc [3]. Then, with slight modification to open repository BitIodine [14] code, we parsed through the blockchain, and wrote wrapper classes that extracted the relevant information required to construct the transaction graph. The market price data was scarped from website `https://blockchain.info/` to plot graphs in R.

2.5 Bitcoin Transaction Network

Before getting the transaction network, we first define our graph, which acts as input to further mathematical application.

Definition 2.1. We consider weighted labelled graph. That is, a graph $G = (V, E, \ell)$ is represented by a set of $|V| = n$ vertices, a set of edges E specified by a weighted adjacency matrix $A \in \mathbb{R}^{n\times n}$, and a label function $\ell\colon V \to \mathcal{L}$ with $\mathcal{L} = \big([k], \mathbb{R}^D\big)$, where k is the number of available node labels and D is the dimension of the continuous attributes. Given $V = \{v_1, v_2, ..., v_n\}$, *node labels* $\ell(v_i)$ are represented by nominal values and *attributes* $\mathbf{x}_i \in \mathbb{R}^D$ are represented by continuous vectors. In a transaction graph, addresses are nodes, transactions are edges and weight is BTC.

With the overall transactions records parsed from the blockchain in human readable form, we construct a a weighted directed transaction graph that gives an intuition towards the flow of Bitcoins from one key to the other,the directed edge represents a particular transaction from a source address to a target address and weight represents the value of the transactions. For our experiments, we constructed graphs for more than two years (March, 2013 to December, 2015) to capture major bubble in Bitcoin history. Figure 2.3 represent daily transaction graph for a typical day on April 8, 2013.

2.6 Transaction Dynamics

The validation of the data parsed from our tool is then checked by reproducing the "Mathew Effect" phenomenon from the seminal work of Kondor et al. [13] paper's using their original matlab code, but with our own data. The authors of the above

[3]`http://dumps.webbtc.com/bitcoin/`

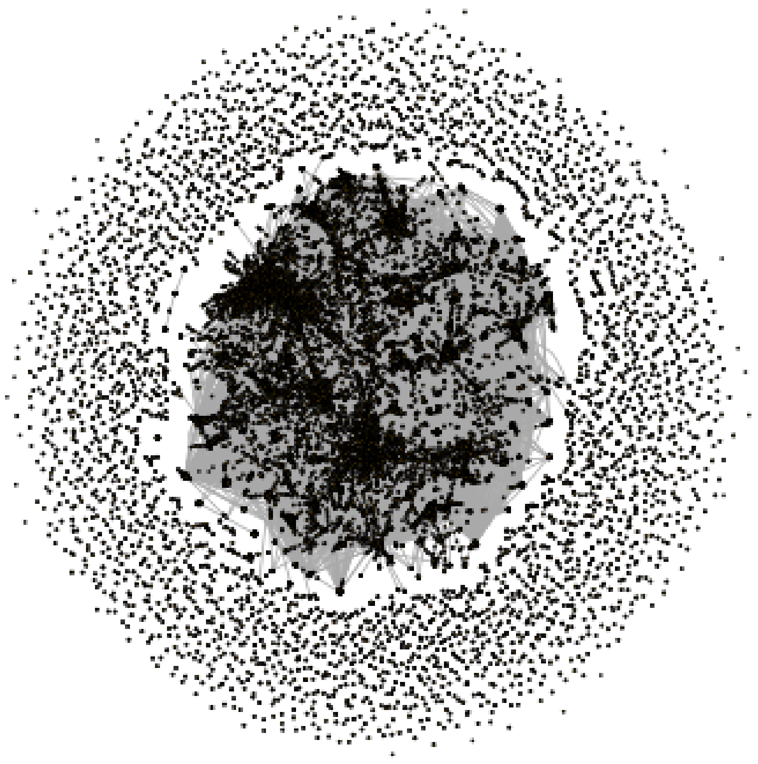


FIGURE 2.3: Daily transaction graph for a typical day (April 8, 2013)

papers are curator of whole blockchain up to 2014.10.19. (326,027 blocks), which is benchmark in quality. The similar figure confirms the high quality of our data.

To capture the transaction dynamics in the bitcoin, we analyze the dynamics of money flow on the transaction network, as discussed in the paper [13]. We try to support popular hypothesis in economics having roots in preferential attachment, called Matthew effect or the "rich get richer phenomenon". It states that the growth of the wealth of each individual is proportional to the wealth of that individual [13].

We assume that the number of bitcoins associated with node n at time t is given by $b_n(t)$. The difference between the balance of each address at the end and at the start of each month is calculated. Then the difference in function of the starting balances is plotted in figure 2.4. The positive correlation between balance and the average growth indicates the "rich get richer" phenomenon in bitcoin.

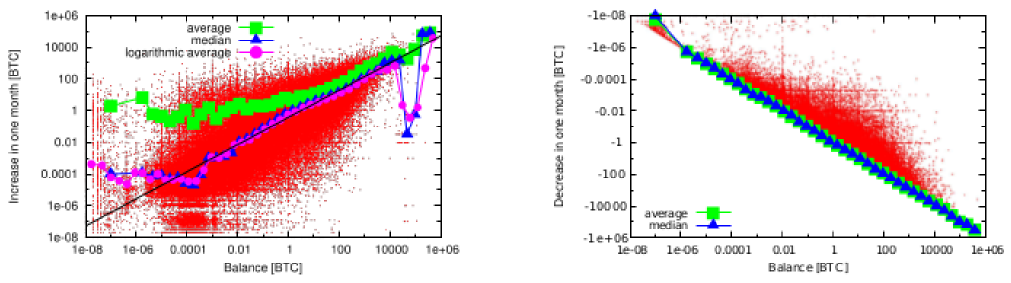


Figure 2.4: Matthew Effect

The figure 2.4 is reproduced from Kondor's paper [13], using our own data where Increase (left) and decrease (right) of node balances in one month windows as a function of their balance at the beginning of each month is represented. The representation in the picture follow the following colors: the raw data (red), the average (green), median (blue) and logarithmic average (magenta). The later three are calculated for logarithmically sized bins. The power-law fit for the double logarithmic data is represented by black line.

2.6.1 Silk Road Arrest

The Silk Road, under the alias of "Dread Pirate Roberts" (DPR) was known as an anonymous marketplace, a Black Market operated as a hidden service only accessible through Tor, where people who use bitcoins were able to buy and sell drugs, art, weapons etc. anonymously, without the risk of being tracked. It was founded by Ross William Ulbricht in February in 2011, but in October 2013 FBI closed the Silk Road and arrested Ulbricht. The FBI seized approximately 173,600 BTC in two phase. At the first go 29,600 BTC held in a so called hot wallet were seized and an additional 144,000 BTC were seized using two addresses [4] controlled by the FBI.

We illustrates the event using transaction network 2.5 realised in gephi. Each vertex represents a user, where address is mapped to the user. Each directed edge between a source and a target represents a flow of Bitcoins from a public-key belonging to the user corresponding to the source to a public-key belonging to the user corresponding to the target.

Although in this scenario involves manual investigation by web scraping , it would have been difficult to find significant links manually, given the millions of nodes involved.

[4] 1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX, 1FfmbHfnpaZjKFvyi1okTjJJusN455paPH

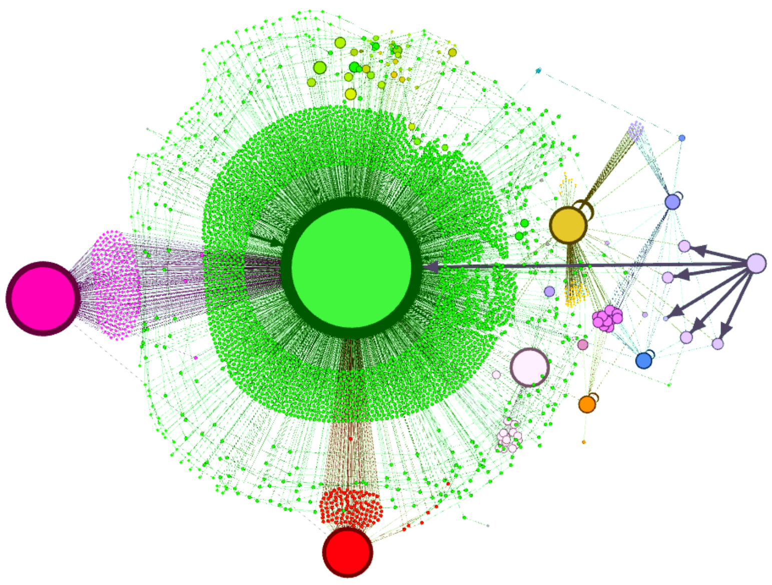


Figure 2.5: Silk Road Arrest

2.7 Summary

This chapter introduces Bitcoin, its working detail by explaining the protocol of blockchain, which contains all the history of transactions in the bitcoin. With enough knowledge about working of bitcoin, the related work in bitcoin transaction network is the discussed with the pros and cons. The data preparation staged is discussed to get the transaction network, an input to our mathematical model in coming chapters. The case study of silk road arrest is illustrated with the transaction graph. Transaction dynamics is reproduced from know studies to validate the parsed data, concludes the chapter.

Chapter 3

Graph Isomorphism

The dramatic proliferation of sophisticated networks has resulted in a growing need for supporting effective analysis and measurement of large-scale graphs whose contents change dynamically over time or after an important event. The availability of large scale bitcoin's transactions allows us to have better understanding of the phenomena, modelled using underlying system of interacting agents. In short, It drives our research in direction to measures the structural change of a dynamic large-scale graph as well as the similarity between two graphs. At the core of above problem definition lies a common and critical graph isomorphism questions: By how much is a graph transformed over time or by a significant event? or how structurally similar are graphs?

3.1 Problem Definition

3.1.1 Problem: Graph Similarity

Two graph $G_1 = (V_1, E_1, \ell_1)$ and $G_2 = (V_2, E_2, \ell_2)$ are isomorphic, denoted by $G_1 \cong G_2$, if there is a structure-preserving bijection $f\colon V_1 \to V_2$ so that if $\{e_1, e_2\} \in E_1$ then $\{f(e_1), f(e_2)\} \in E_2$. where V is the set of nodes and E is the set of edges in graph G. Weighted adjacency matrices $A \in \mathbb{R}^{n_i \times n_i}$ represents and the label function ℓ endows nodes with label and attribute information.

3.1.2 Solution: Similarity Index

Similarity Index, intuitively signifying the structural similarity between two given graphs is defined as $SI(G_1, G_2)$, $0 \leq SI(G_1, G_2) \leq 1$. Where, $SI(G_1, G_2) = 0$ indicates that two graphs are structurally complementary and $SI(G_1, G_2) = 1$ means two graphs are completely identical.

3.2 Literature Review

Long in the purview of researchers, graph isomorphism is a well-studied problem in literature. Besides its practical importance, the graph isomorphism problem is a curiosity in computational complexity theory as it is one of a very small number of problems belonging to NP neither known to be solvable in polynomial time nor NP-complete. The classic open problem in the algorithmic theory of random graphs for decades was recently solved by Babai [16]. He showed that it can be solved in in quasipolynomial time. Unfortunately, an algorithm has not yet been created to test such algorithms, thus, it will have to be vetted by other experts in the field before it can be labeled as a success.

But, there are several approaches proposed to solve variations of the problem. The related work comprises five main areas: Subgraph Isomorphism, Graph Edit Distances, Topological Descriptors, Iterative Method and Polynomial Alternatives. We give the related work in each area separately, and by mentioning advantages/disadvantages, we try to justify how our chosen method out of them outweighs others.

3.2.1 Subgraph Isomorphism

Given the query graph Q and a subgraph G^s of G, an isomorphism between Q and G^s involves finding a bijective function $f\colon V(Q) \rightarrow V(G^s)$ such that for any two vertices $v_1 \in V(Q)$ and $v_1 \in V(Q)$, $(v_1, v_2) \in E(Q) \Rightarrow (f(v_1), f(v_2)) \in E(G^s)$.

Subgraph Isomorphism is a fundamental problem in graph big data similarity analysis. Subgraph Isomorphism is an NP-complete problem. Adding to the same, most of the algorithms belonging to subgraph isomorphism are based on a backtracking method. The former computes the solutions by incrementally enumerating and verifying candidates for all vertices in a query graph. Also, excessive runtime in worst case may grow exponentially with the number of nodes. So, for larger graphs with many nodes and for large datasets of graphs, this is an enormous problem [24].

With the above problem for the practitioners, we need polynomial-time similarity measure for graphs.

3.2.2 Graph Edit Distances

With evolving bitcoin transaction graph, where thousands of nodes keeps adding to the network everyday, the inexact graph similarity measure plays a crucial role. The inexact similarity measure consists on finding a distortion or variation between two input graphs, where there may not exist an exact match (two graphs are similar if its topology and labeling is identical).

The Inexact graph similarity has been one of the significant research foci in the area graph isomorphism. As an important way to measure the similarity between graphs, graph edit distance (GED) is the base of inexact graph similarity. The idea of graph edit distance is to define the dissimilarity of graphs by the amount of distortion that is needed to transform one graph into another. The transformation of the graphs involves assigning costs to different types of operations (edge/node insertion/deletion, modification of labels). Using the edit distance, an input graph to be classified can be analyzed by computing its dissimilarity to a number of training graphs [18]. The GED is explained better using mathatically notations and definition below:

3.2.2.1 Problem Definition

Attributed graphs (as defined in Definition 3.1) are powerful data structures for the representation of complex entities. To define GED, we first define attributed graph, a modification of earlier graph definition 2.1:

Definition 3.1. An attributed graph G is a 4-tuple $G = (V, E, \ell_v, \ell_e)$, where :

- V is a set of vertices,
- E is a set of edges, such that $\forall e = (i, j) \in E, i \in V$ and $j \in V$,
- $\ell_v : V \to L_V$ is a vertex labeling function which associates the label $\ell_v(v)$ to all vertices v of V, where L_V is the set of possible labels for the vertices,
- $\ell_e : E \to L_E$ is an edge labeling function which associates the label $\ell_e(e)$ to all edges e of E, where L_E is the set of possible labels for the edges.

The vertices label space (L_V) and respective edges label space (L_E) may be composed of any combination of numeric, continuous, symbolic or string attributes.

With the graph definition 3.1, which allows us to handle arbitrarily structured graphs (directed or undirected, simple graphs or multigraphs) with unconstrained labeling functions, we define GED as the dissimilarity of two graphs by the minimum amount of distortion that is needed to transform one graph into another [19].

Definition 3.2. The graph edit distance $d(.,.)$ is a function

$$\begin{aligned} d \ : \ & \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+ \\ & (G_1, G_2) \mapsto d(G_1, G_2) = \\ & \min_{o=(o_1,\ldots,o_k)\in\Gamma(G_1,G_2)} \sum_{i=1}^{k} c(o_i) \end{aligned}$$

where $G_1 = (V_1, E_1, \ell_{v1}, \ell_{e1})$ and $G_2 = (V_2, E_2, \ell_{v2}, \ell_{e2})$ are two graphs from the set $\mathcal{G}$ and $\Gamma(G_1, G_2)$ is the set of all edit paths $o = (o_1, \ldots, o_k)$ allowing to transform G_1 into G_2. An elementary edit operation o_i is one of vertex substitution $(v_1 \to v_2)$, edge substitution $(e_1 \to e_2)$, vertex deletion $(v_1 \to \varepsilon)$, edge deletion: $(e_1 \to \varepsilon)$, vertex insertion $(\varepsilon \to v_2)$ and edge insertion $(\varepsilon \to e_2)$ with $v_1 \in V_1$, $v_2 \in V_2$, $e_1 \in E_1$ and $e_2 \in E_2$. ε is a dummy vertex or edge which is used to model insertion or deletion. $c(.)$ is a cost function on elementary edit operations o_i that satisfies

- $c(v_1 \to v_2) \leq c(v_1 \to v) + c(v \to v_2)$
- $c(e_1 \to e_2) \leq c(e_1 \to e) + c(e \to e_2)$
- $c(v_1 \to \varepsilon) \leq c(v_1 \to v) + c(v \to \varepsilon)$
- $c(e_1 \to \varepsilon) \leq c(e_1 \to e) + c(e \to \varepsilon)$
- $c(\varepsilon \to v_2) \leq c(\varepsilon \to v) + c(v \to v_2)$
- $c(\varepsilon \to e_2) \leq c(\varepsilon \to e) + c(e \to e_2)$

Moreover, in order to guarantee the symmetry property $(d(G_1, G_2) = d(G_2, G_1))$, the reverse edit path should result in the same cost. So, these costs have to be defined in a symmetric manner so that $c(v_1 \to v_2) = c(v_2 \to v_1)$, $c(e_1 \to e_2) = c(e_2 \to e_1)$, $c(v \to \varepsilon) = c(\varepsilon \to v)$ and $c(e \to \varepsilon) = c(\varepsilon \to e)$ [19].

3.2.2.2 Advantages

The main advantages of algorithms based on GED and its extension is that, It captures partial similarities between graphs. It also allows for noise in the nodes, edges and their labels, which other other algorithms are incapable. One more advantage is that, It is flexible way of assigning costs to different operations [20].

3.2.2.3 Disadvantages

Comparison of the similarity of corresponding nodes and edges in the computation of GED is still not solved well. For example, in attributed graphs, attributes of nodes and edges can be used for comparing the similarity. But the choice of attributes and its available for computing distance remains an open problem [18].

The selection of the cost function for different edit operations is available for limited applications, or under some constrains, so some definitions of costs, which can be applied extensively and easily, are difficult to get.

Some GED based algorithms contains subgraph isomorphism as one intermediate step bring out the problem of NP completeness and excessive runtime in worst case.

3.2.3 Topological Descriptors

Another approach on attacking graph isomorphism is based on features extraction, where each graph is mapped to a feature vector. Topological descriptors are based on a graph representation of the complex network, which includes number of edges, nodes, nodes degree, label distribution, paths, walk etc. The basic idea is to use topological descriptors and graph decompositions to define graph similarity measures using approaches, which derive feature spaces based on topological descriptors and integrate topological decomposition into similarity measures. After that distances and metrics are used on vectors for learning on graphs.

The λ-distance, a spectral method and levenshtein distance which defines the distance between two graphs as the distance between their spectra (eigenvalues) has been studied thoroughly [20]. The study by Li et al.[21] proposes an SVM-based approach on extracted features (average degree, eccentricity, number of nodes and edges, eigenvalues, clustering coefficient, diameter etc.) to perform graph isomorphism. Other techniques includes, computing edge curvatures under heat kernel embedding, comparing graph signatures consisting of summarized local structural features and a distance based on graphlet correlation [20].

Going in the research direction of tracking changes in networks over time, spotting anomalies and detecting events, Lee et al.[24] graph similarity approach is based on random walk with restart (RWR) algorithm with intergraph compression to transform representation of graph. The method is efficient in space requirement and produces results more quickly and accurately over conventional graph transformation schemes. The methods fails in picking Euclidean distance as measurement of matrix distance, as it become weakly discriminant when we have multidimensional and sparse data. Another work by Mheich et al. [22] on measure graph similarity takes account for vertices, edges and spatiality at the same time, which is unseen in literature.

3.2.3.1 Advantages

Representing of graph as topological descriptor has some great advantages. The graph mapping to feature vectors can be Reused known, so act as efficient tools for feature vectors.

3.2.3.2 Disadvantages

Though these methods are powerful and scale well, but depending on the statistics that are chosen, it is possible to get results that are not intuitive. For example, it is possible to get high similarity between two graphs with very different node set size, which is not always desirable. Adding to the same, feature vector transformation leads to loss of topological information or includes subgraph isomorphism as one step.

3.2.4 Iterative Methods

Next approach is based on iterative methods, which is based on the concept to find the "best" correspondence between the nodes of the two graphs. The research direction attempting to solve the graph alignment problem is flooded with the methods span from genetic algorithms to decision trees, clustering, expectation-maximization, similarity flooding algorithm [23], message-passing algorithm for aligning sparse networks and belief propagation [20], to name few of them. The advantages of all enlisted methods are their speed and simplicity, but they do not take into account information about the graph structure. On graph similarly measure with given node correspondence, Koutra et al. [20] proposed DELTACON , a principled, intuitive, and scalable algorithm that assesses the similarity between two graphs on the same nodes using Fast Belief Propagation on real graphs from ENRON e-mail exchange and brain scans. The specific application to node correspondence make its unsuitable for dynamic graphs.

3.2.5 Polynomial Alternatives

Another important approach which work directly on the graphs without doing feature extraction is called graph kernels. It compare substructures of graphs that are computable in polynomial time. It characterize graph features in a high dimensional space and thus better preserve graph structures. Most of graph kernels are instances of the family of R-convolution kernels proposed by Haussler [25], which define graph kernels by comparing all pairs of isomorphic substructures under decomposition, and a new decomposition will result in a new graph kernel. It can be categorised into classes based on comparing all pairs of walks, paths, cycles [27], trees and graphlets in polynomial time [28]. We will do details review of graph kernels in the next chapter 4.

3.2.5.1 Drawback

The graph kernels arising from the R-convolution kernels neglect the relative locations of substructures. As a drawback, the R-convolution kernels cannot establish reliable structural correspondences between the substructures [26]. The basic graph kernels also suffers from prohibitively expensive runtime, $O(n_6)$, tottering and halting problem, which keep swinging with recent research papers [28].

3.3 Summary

This chapter review the important studies in graph isomorphism problem under the relevant heading. With the evolving research, the research on graph kernels has gone high, but this chapter **??** review the contemporary best graph isomorphism algorithms for analysis.

Chapter 4

Graph Kernels

Kernels methods offer natural framework to study machine learning questions in different field, where graphs are used to model relationships between structure, with nodes representing objects and edges the relations between them. In this scenario, researcher direction leads to one important questions: “How similar are two graphs to each other?”. To answer the above question, the kernels between graph was first proposed by Gärtner et al. [30] and later extended by Borgwardt et al.[31]. The graph kernels replace the explicit projection in feature space with the evaluation of a symmetric semi-definite positive similarity function. It can utilize infinite possible feature spaces by the learning algorithm with complexity of the kernel function, rather on the size of the feature space. Most kernel functions for graphs associate specific types of substructures to features, so the evaluation is then related to the number of common substructures between two graphs. The most common substructures widely used in the literature includes include random walks, paths, tree structures and rational kernels [28]. Now, we review some of the representative graph kernels.

4.1 Random Walk Graph Kernels

One of the popular graph kernel, random walk graph kernel [30, 32] counts matching walks in two input graphs. The basic idea behind this kernel to perform simultaneous random walks between the vertexes of the graphs, given a pair of graphs and count the number of matching paths.

4.1.1 Computation

Taking two graph G_1, G_2, let $E_\times$ denote the adjacency matrix of their direct product $E_\times = E(G1 \times G2)$, and $V_\times$ denote the vertex set of the direct product $V_\times = V(G1 \times G2)$. With a sequence of weights $\lambda = \lambda_0, \lambda_1, \ldots$ ($\lambda_i \in R$; $\lambda_i \geq 0$ for all $i \geq N$) the direct product kernel is defined as:

$$K_\times(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{k=0}^{\infty} \lambda_k E_\times^k \right]_{ij} \tag{4.1}$$

The graph kernel $K(G_1, G_2)$ is computed efficiently computing matrix power series $\lim_{n\to\infty} \sum_{i=0}^{n} \lambda_i E^i$, which is explained in Gartner et al. [30] by taking exponential and geometric series. The graph kernel is normalised to get similarity index in our case. In simple words, elegant computation involves, computing walk of length k by looking at the k^{th} power of the adjacency matrix, then constructing direct product of the graph G_1, G_2 and counting walk on the product graph $K_\times(G_1, G_2)$. $E^k(i, j) = c$ means that c walks of length k exist between vertex i and vertex j.

4.1.2 Disadvantages

Although random walk kernels and path based kernels are still among the widely adopted graph kernels, It is prohibitively expensive, requiring $O(n_6)$ runtime and suffers from the problem of tottering [34] and halting [35]. Other common disadvantage with them is that walks and paths do not capture information of the substructures present in the graph.

4.1.3 Potential solutions

The computation of the above was greatly improved by Vishwanathan et al. [28] using iterative methods, including those based on Sylvester equations, conjugate gradients, and fixed-point iterations. Other improvements by Kang et al. [33] takes account for just unlabeled graphs with the normalized weight matrix. Tottering problem was solved by Mahe et al.[34] by doing special transformation on the input graphs, but it suffer from adverse computation time ($O(n)$ to $O(n_2)$ and does not show a uniform improvement of classification accuracy. Replacing walk by path Borgwardt et al.[31], solved the tottering and halting problem, but dense matrix representation for connected graphs, may lead to memory problems on large graphs.

But the recent paper by Sugiyama and Borgwart [35] claimed geometric random walk kernels suffers from the problem referred to as halting: Longer walks are downweighted so much that the similarity score is completely dominated by the comparison of walks of length 1. This means that defining kernels for the graph is ongoing process, which keeps the improvements cycles on.

4.2 Shortest Path Kernel on Graphs

The foundation of shortest path graph kernels emerged because, paths do not suffer from tottering, as the number of self-loop-avoiding paths between all pairs of nodes of a given graph is useful for understanding the structure of the graph. But computing the number of such paths between all nodes is however a computationally hard task. Instead, only the number of shortest paths is counted between node pairs in polynomial time avoiding cycles, thus is used as best structural measure in graph kernels [36].

4.2.1 Methodology

The first steps involves computing all-pairs-shortest-paths for G_1 and G_2 via Floyd-Warshall or Dijkstra's algorithm. We then define graph kernel as a function $k(G_1, G_2)$ on pairs of graphs, which can be represented as an inner product $k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle_{\mathcal{H}}$ for some mapping $\phi(G)$ to a Hilbert space $\mathcal{H}$, of possibly infinite dimension. Denoting $D(G)$ as the multi set of shortest distances between all node pairs in the graph G, we define shortest path kernels two given graphs G_1 and G_2 as:

$$K_{\mathrm{SP}}(G_1, G_2) = \sum_{d_1 \in D(G_1)} \sum_{d_2 \in D(G_2)} k(d_1, d_2), \tag{4.2}$$

where k is a positive definite kernel [31, 36].

4.2.2 Advantages

The shortest path graph kernels doesn't suffer from tottering, and has better accuracy on classification benchmarks. Runtime is in $O(n_4)$, which includes computing all-pairs-shortest-paths for G_1 and for G_2: $O(n_3)$. Also, comparing all pairs of shortest paths from G_1 and G_2: $O(n_4)$. It is empirically faster than (fast) random walk kernels (probably due to graph size) [36].

4.2.3 Disadvantages

The complexity of order $O(n_4)$ is too slow for large graphs. The essential step, dense matrix representation for connected graphs, may lead to memory problems on large graphs.

4.2.4 Potential Solution

The above problem of dense matrix representation was solved by new tree-based kernel for graphs [37]. Graphs are decomposed into multisets of ordered Directed Acyclic Graphs (DAGs) and a family of kernels computed by application of tree kernels extended to the DAG domain. They provide richer representation of graph structure than walk-based approach, but their runtime grows exponentially with the recursion depth of the tree.

4.3 Graphlet Kernels

While using graph kernels, a practitioner is faced with some important dilemmas: Which graph kernels to be chosen for particular application? If chosen, how good it capture graph similarity by being better than others? Is it cheap to compute? Unfortunately, all the above questions are too difficult to be answered, and no theoretical justification supports the selection of particular graph kernels.[39].

The one way to look into the solutions of the above problem is efficiently computable representation that adequately captures the topology of the input graphs. Graph kernels do it by counting on the distribution of subgraphs of size k, refereed to as "graphlets".

4.3.1 Description

Using the notation from Yanardag and Vishwanathan paper [39], we define graph as pair $G = (V, E)$ where $V = \left\{v_1, v_2, \dots, v_{|V|}\right\}$ is an ordered set of *vertices* or *nodes* and $E \subseteq V \times V$ is a set of *edges*. Given $G = (V, E)$ and $H = (V_H, E_H)$, H is a *sub-graph* of G iff there is an injective mapping $\alpha : V_H \to V$ such that $(v, w) \in E_H$ iff $(\alpha(v), \alpha(w)) \in E$. Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there exists a bijective mapping $g : V \to V'$ such that $(v_i, v_j) \in E$ iff $(g(v_i), g(v_j)) \in E'$. *Graphlets* are small, connected, non-isomorphic sub-graphs of a large network.

With clear notation, we now define graphlet kernels. Let $\mathcal{G}_k = \{g_1, g_2, \dots, g_{n_k}\}$ be the set of size-k graphlets where n_k denotes the number of unique graphlets of size k. Given a graph G, we define f_G as a normalized vector of length n_k whose i-th component corresponds to the frequency of occurrence of g_i in G:

$$f_G = \left(\frac{c_1}{\sum_j^{n_k} c_j}, \cdots, \frac{c_{n_k}}{\sum_j^{n_k} c_j}\right)^T. \tag{4.3}$$

Here c_i denotes number of times g_i occurs as a sub-graph of G. Given two graphs G and G', the graphlet kernel k_g is defined as:

$$k_g(G, G') := f_G^\top f_{G'}, \tag{4.4}$$

which is simply the dot product between the normalized graphlet-frequency vectors [39].

4.3.2 Problem

The graphlet kernel described above uses induced sub-graphs of k nodes as motifs in the vector representation, and computes the kernel via a dot product between these vectors. As the size of k increases, the sparsity problem pitch in and most higher order graphlets will not occur in a given graph. This affect is diagonal dominance, where a given graph is similar to itself but not to any other graph in the dataset [39]. Also, more numerous lower order graphlets withlower values of k does not provide enough discrimination ability.

4.3.3 Improvement

To tackle the above problems, Yanardag and Vishwanathan [39] proposed smoothing technique based on a novel extension of Kneser-Ney and Pitman-Yor smoothing techniques from natural language processing to graphs. The smoothing algorithm tackles the diagonal dominance problem by distributing the probability mass across graphlets and preserve the dependency.

4.4 Graphs Kernels with Continuous Attributes

A plethora literature on graph kernels (including above) is devoted to discrete attributes graph structures. Out of them, most are not suited for non-discrete node labels since their computational efficiency hinges on avoiding to consider matches between distinct discrete labels. An alternative extend the definition of traditional graph kernels, and consequently derive a graph kernel, which is able to deal with complex and continuous node labels. The same was done by Martino et al. [37] by extended tree kernels to continous attributes. An open challenge to develop a scalable kernel on graphs with continuous-valued node attributes was taken by Feragen et al. [38]. By taking convolution kernel counting sub-path similarities, they presented GraphHopper kernel between graphs with real-valued edge lengths and any type of node attribute, including vectors. The detail description of graph kernels with contionus contributes is discussed in the next section.

4.5 Propagation Kernels

Recent work on graph kernels tries to capture structural information encoded in node labels, attributes, and edge information to come up ”propagation kernels”, which can be used to construct kernels for many graph types, including labeled, partially labeled, unlabeled, directed, and attributed graphs [29]. Propagation kernels leverage the power of continuous node label distributions as graph features and hence, enhance traditional graph kernels to efficiently handle partially labeled graph in a principled manner.

4.5.1 Method

Taking the help of propagation kernels [29], We define a kernel $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ among graph instances $G^{(i)} \in \mathcal{X}$. The input space $\mathcal{X}$ comprises graphs $G^{(i)} = (V^{(i)}, E^{(i)}, \ell)$, where $V^{(i)}$ is the set of nodes and $E^{(i)}$ is the set of edges in graph $G^{(i)}$. Weighted adjacency matrices $A^{(i)} \in \mathbb{R}^{n_i \times n_i}$ represents and the label function ℓ endows nodes with label and attribute information. The similarity between two graphs $G^{(i)}$ and $G^{(j)}$ is to computed by comparing all pairs of nodes in the above two:

$$K(G^{(i)}, G^{(j)}) = \sum_{v \in G^{(i)}} \sum_{u \in G^{(j)}} k(u, v),$$

where $k(u, v)$ is an arbitrary node kernel determined by node labels and, if present, node attributes. It is defined in terms of the nodes' corresponding probability distributions $p_{t,u}$ and $p_{t,v}$, which we update and maintain throughout the process of information propagation.

The kernel contribution of iteration t is defined by

$$K(G_t^{(i)}, G_t^{(j)}) = \sum_{v \in G_t^{(i)}} \sum_{u \in G_t^{(j)}} k(u, v). \tag{4.5}$$

The propagation kernels between labeled and attributed graphs are defined as:

$$k(u, v) = k_l(u, v) \cdot k_a(u, v), \tag{4.6}$$

where $k_l(u, v)$ is a kernel corresponding to label information and $k_a(u, v)$ is a kernel corresponding to attribute information. If no attributes are present, then $k(u, v) = k_l(u, v)$. The t_{MAX}-iteration propagation kernel is now given by

$$K_{t_{\text{MAX}}}(G^{(i)}, G^{(j)}) = \sum_{t=1}^{t_{\text{MAX}}} K(G_t^{(i)}, G_t^{(j)}). \tag{4.7}$$

If node kernel is defined in the form

$$k(u, v) = \begin{cases} 1 & \text{if } \textit{condition} \\ 0 & \text{otherwise,} \end{cases} \tag{4.8}$$

where *condition* is an equality condition on the information of nodes u and v, we can compute K efficiently by *binning* the node information, *counting* the respective bin strengths for all graphs, and *computing a base kernel* among these counts. That

Algorithm 1 The general propagation kernel computation [29].

```
given: graph database $\{G^{(i)}\}_i$, # iterations $t_{\text{MAX}}$, propagation scheme(s), base kernel $\langle\cdot,\cdot\rangle$
$K \leftarrow 0$, initialize distributions $P_0^{(i)}$
for $t \leftarrow 0 \ldots t_{\text{MAX}}$ do
    for all graphs $G^{(i)}$ do
        for all nodes $u \in G^{(i)}$ do
            quantize $p_{t,u}$, where $p_{t,u}$ is $u$-th row in $P_t^{(i)}$    ▷ bin node information
        end for
        compute $\Phi_{i\cdot} = \phi(G_t^{(i)})$    ▷ count bin strengths
    end for
    $K \leftarrow K + \langle\Phi, \Phi\rangle$    ▷ compute and add kernel contribution
    for all graphs $G^{(i)}$ do
        $P_{t+1}^{(i)} \leftarrow P_t^{(i)}$    ▷ propagate node information
    end for
end for
```

is, we compute count features $\phi(G_t^{(i)})$ for each graph and plug them into a base kernel: $\langle\cdot,\cdot\rangle$

$$K(G_t^{(i)}, G_t^{(j)}) = \langle\phi(G_t^{(i)}), \phi(G_t^{(j)})\rangle. \tag{4.9}$$

The above mathematical equations can be put in the form of algorithms 1.

4.5.2 Advantages

There are two major benefit of using propagation kernels: The shelf propagation schemes as discussed above can be used to naturally construct kernels for labeled, partially labeled, unlabeled, directed, and attributed graphs. Also, by leveraging existing efficient and informative propagation schemes, propagationkernels can be considerably faster than state of the art approaches [29].

4.6 Recent Graph Kernels

Other graphs kernels includes, Optimal Assignment Kernels and Edit-Distance Kernel, which are not positive definite in general; Subtree Kernel runtime grows exponentially with the recursion depth of the subtree like patterns; Cyclic Pattern Kernel restrict their attention to scenarios where the number of simple cycles in a graph dataset is bounded by a constant and Graphlet Kernel's common solutions not feasible on labeled graphs [28].

Bai et al.[26] developed a novel graph kernel by aligning the Jensen-Shannon (JS) representations of vertices, which addresses the drawback of neglecting the relative locations between substructures that arises in the R-convolution kernels.

To solve the problem of graph structure at multiple different scales, Kondor and Pan [40] introduces, Multiscale Laplacian Graph kernels (MLG kernels), which has the Feature Space Laplacian Graph kernel (FLG kernel) with the property that it can lift a base kernel defined on the vertices of two graphs to a kernel between the graphs.

In the recent KDD 2015 paper by Yanardag and Vishwanathan [8] used language modeling and deep learning to learn latent representations of sub-structures for graphs. Their framework leverages the dependency information between sub-structures by learning their latent representations. Using three popular graph kernels, namely Graphlet kernels, Weisfeiler-Lehman subtree kernels, and Shortest-Path graph kernels, their models proves to be improves the classification accuracy, robust to random noise and computationally efficient. We discuss deep graph kernels in the next chapter with scholastic detail.

4.7 Methodology

Here, we present a method to efficiently measure the change of a dynamic graph (bitcoin transaction graph) whose connectivity structure evolves over time with respect to changes in the exchange price of bitcoins. and quantative measure of the similarity of large-scale graph datasets along with justification for our method.

In simple word, given two graph G_1, G_2, we need to find similarity index $SI(G_1, G_2)$, that lies between 0 and 1.

4.7.1 Graph Dataset

We parsed the transaction data from blockchain in the form 4.1, which is automatically downloaded at the local machine, once Bitcoin Core is set-up. Once the data is loaded at SQL database, we extract the daily transaction for the month of March, 2003 to May, 2003 to capture maximum variation, as this period have two major bubbles. The resulting transaction network for the April, 2013 consists of 64,782 addresses with 100,952 edges.

Table 4.1: Transaction Data Format

$Transaction_{From}$	$Transaction_{To}$	$Value$	$Timestamp$

4.7.2 Graph Kernel Calculation

We use widely used random walk graph kernels (RWGK) to calculate similarity index. As RWGK suffers from various problem, which is then corrected by propagation graph kernels (PGK), we intend to use it too, so that fair comparison can be done.

4.7.2.1 Similarity Index: RWGK

We take simple case of random walk graph kernels as defined in section 4.1. Taking two graph G_1, G_2, let $E_\times$ denote the adjacency matrix of their direct product $E_\times = E(G1 \times G2)$, and $V_\times$ denote the vertex set of the direct product $V_\times = V(G1 \times G2)$. With a sequence of weights $\lambda = \lambda_0, \lambda_1, ...$ ($\lambda_i \in R$; $\lambda_i \geq 0$ for all $i \geq N$) the direct product kernel is defined as:

$$K_\times(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{k=0}^{\infty} \lambda_k E_\times^k \right]_{ij} \tag{4.10}$$

The graph kernel $K(G_1, G_2)$ is computed efficiently computing matrix power series $\lim_{n\to\infty} \sum_{i=0}^{n} \lambda_i E^i$, which is explained in Gärtner et al.[30].

The similarity index $SI_{RWGK}(G_1, G_2)$ is calculated by normalising graph kernels. The method is implemented in Python by converting readily available Matlab code [32].

4.7.3 Similarity Index: PGK

Propagation graph kernel is calculated by method as discussed in section 4.5.

$$K(G_t^{(i)}, G_t^{(j)}) = \langle \phi(G_t^{(i)}), \phi(G_t^{(j)}) \rangle. \tag{4.11}$$

The similarity index $SI_{PGK}(G_t^{(i)}, G_t^{(j)})$ is calculated by normalising graph kernels. The method is implemented in Python by converting readily available Matlab code [29].

Note that graph similarity reflecting overall graph structure ends up lying in between 0 and 1, as governed by normalised version of Eq. 4.10 and 4.11.

4.8 Results and Discussion

To investigate the connection between the network structure and macroscopic properties (i.e. the exchange price) in the Bitcoin network, we plot similarity index and BTC/USD exchange price. This in turn, offer an accurate and efficient similarity measure for dynamically changing large scale graphs and structurally comparable graphs.

We first plot the similarity index calculated by popular RWGK and BTC/USD exchange price.

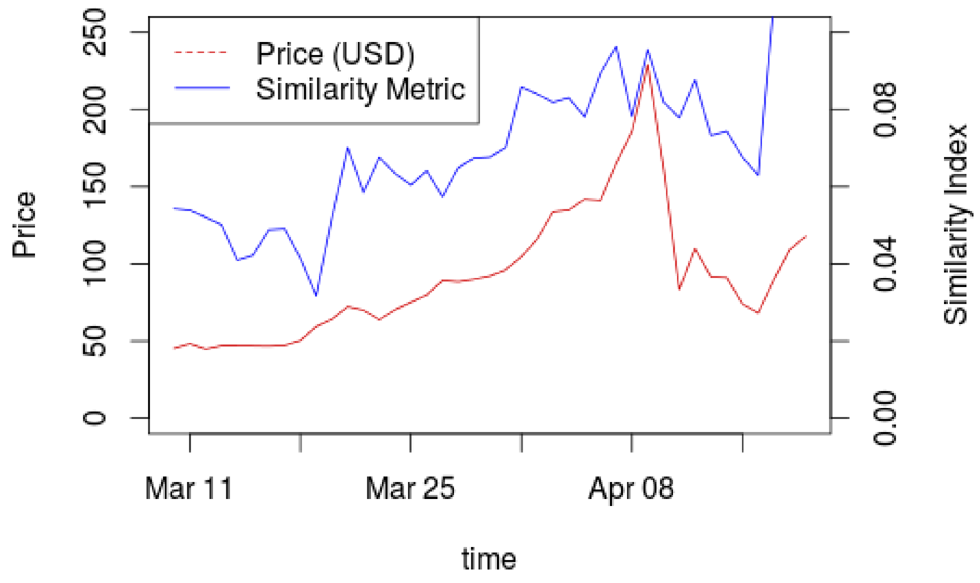


FIGURE 4.1: Daily transactions price correlation with network structure. The similarity index was calculated by RWGK

Although random walk kernels are still among the widely adopted graph kernels, one common disadvantage with them is that walks and paths do not capture information of the substructures present in the graph. The weak correlation as seen in the figure 4.1 support our reasoning. Possibility of tottering [34] and halting [35] problem cannot be ruled out. This in turns focus our direction to use other graph kernels, free from above problem and can be scalable.

Next we use propagation kernels, whose design is motivated by the iterative information propagation. They not only capture structural information, but can often adapt to the aforementioned issues of real world data. Adding to the same, they didn't suffer from the problem of halting and tottering [29]. Figure 4.2 plot the correlation between network structure and BTC/USD exchange price.

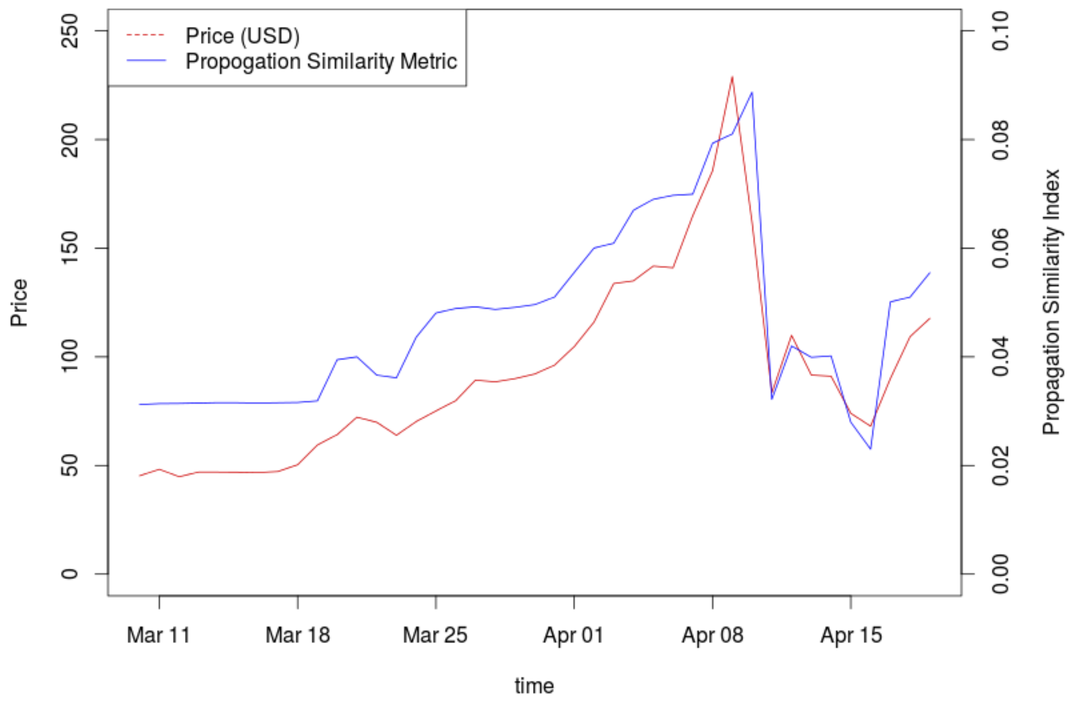


FIGURE 4.2: Daily transactions price correlation with network structure. The similarity index was calculated by PGK

As expected figure 4.2 infer that the time-varying contribution of large scale transaction network correspondence with the market price of bitcoins, but there is still hope for the improvement, as graph is not smooth.

4.9 Conclusions

The purpose of this thesis is to offer an accurate and efficient similarity measure for dynamically changing large-scale graphs and structurally comparable graphs. By defining quantitative measure of transformation over time, in terms of similarity index using diffemt kernels, we infer that there is correspondence between network structure and exchange price in bitcoin. The not so smooth graph direct our research to use other graph kernels. Our analysis of real-world transaction data from bitcoin showcases a successful application of the method, which can further be employed in analyzing data from various fields.

4.10 Summary

This chapter review of representative graph kernels in the literature. By taking conventional and sophisticated graph kernels, this chapter defines quantitative measure of transformation over time. The chapter concludes with not so smooth correspondence between network structure and exchange price in bitcoin, thus paving way for further investigation to other graph kernels.

Chapter 5

Deep Graph Kernels

Over recent years, graph kernels are one the promising candidate for problem related to graph isomorphism. Graph Kernels are based on the comparison of graph-substructures via kernels. The graph-substructure including walks, path, subtrees, graphlets and cyclic patterns have been widely used to construct kernels. However, kernels on these substructures are either computationally expensive, NP-hard to determine, limited in their expressiveness or applicable to just a small subset of graphs rather all graphs. Existing graph kernels have difficulties with reaching at least one of these goals.

An emerging line of research, consisting in designing meaningful kernels on structured data, produced promising results in cheminformatics, bioinformatics, computer vision and natural language processing. The graph kernels together with techniques from natural language processing and computer vision have recently received an increased attention, and showed good performances for classification of small molecules [34]. In very recent years, lots of research has been developed around deep learning architectures that allow for higher level abstractions for representing and understanding graph data [41].

Deep learning is method based on learning representations of data, which attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations. By using deep learning, Perozzi et al.[41] proposed DeepWalk, which uses local information obtained from truncated random walks to learn social representations of vertices of graphs. Motivating from DeepWalk, Yanardag and Vishwanathan [8] work focuses on learning similarities between structured objects, such as graphs and strings, instead of nodes of the single graph. However, the adoption of deep learning in graphs isomorphisms has not been adequately investigated yet.

5.1 Background

Deep Graph Kernels [8] is not so deep learning frame work to learn latent representations of sub-structures for graphs, which are not independent. The framework leverage on the dependency information between sub-structures by learning their latent representations. To motivate our discussion of using deep graph kernels, we first try to explain the problem associated with general graph kernels representation.

5.2 Representation Problem

Graph Kernels are produced using R-convolution, where a graph is to recursively decomposed into into “atomic” sub-structures to define a kernels between them. Mathematically, It can be expressed as:

$$K(G_1, G_2) = \Big\langle \Phi(G_1), \Phi(G_1) \Big\rangle_{\mathcal{H}} \tag{5.1}$$

Where, $\Phi(G)$ is denote a vector which contains counts of atomic sub-structures.

The representation in the equation 5.1 suffers from two basic problems. First, the sub-structure are not independent. Second, the dimension of the feature space often grows exponentially, as substructure grows, which leads to ”diagonal dominance” [42], that is, a given graph is similar to itself but not to any other graph in the dataset. But, we need a kernel matrix where all entries belonging to a class are similar to each other, and dissimilar to everything else.

5.3 Deep Framework

To alleviate the representation problem, we define alternative graph kernel as:

$$K(G_1, G_2) = \Phi(G_1)^T \mathcal{M} \Phi(G_1) \tag{5.2}$$

where $\mathcal{M}$ represents a $|\mathcal{N}| \times |\mathcal{N}|$ positive semi-definite matrix that encodes the relationship between sub-structures and $\mathcal{N}$ represents the vocabulary of sub-structures obtained from the training data. Now, the important task is design $\mathcal{M}$, which can be computed by learning latent representations of sub-structures or by edit-distance relationship between sub-structures.

5.4 Methodology

The $\mathcal{M}$ is computed by learning the latent representations of sub-structures by techniques such as Neural language models, Continuous bag-of-words (CBOW) and Skip-gram model (SG) [8].

5.4.1 Skip-Gram

First, we review the representative Skip-Gram [43] [1]. architecture in language modelling before transforming the ideas to learn representations of sub-structures.

The SG model maximizes co-occurrence probability among the words that appear within a given window. More precisely, SG model tries to minimize the following objective function:

$$\mathcal{L}_{SG} = \sum_{t=1}^{T} \log P(w_{t-c}, \cdots, w_{t+c} | w_t) \tag{5.3}$$

where, $P(w_{t-c}, \cdots, w_{t+c}|w_t)$ is computed as $\prod_{-c \leq j \leq c, j \neq 0} P(w_{t+j}|w_t)$. Furthermore,

$$P(w_{t+j}|w_t) = \frac{\exp(v_{w_t}^{\intercal} v'_{w_t})}{\sum_{w=1}^{\mathcal{V}} \exp(v_{w_t}^{\intercal} v'_{w_t})} \tag{5.4}$$

Hierarchical softmax or negative sampling are two efficient algorithms that are used in training SG and CBOW [8].

5.5 Deep Graph Kernels

The language modeling and deep learning techniques was used to compute $\mathcal{M}$ with intuition that different sub-structures compose graphs in a similar way that different words form sentences when used together. The list of decomposed sub-structures for each graph is then treated as a sentence that is generated from a vocabulary $\mathcal{V}$ where vocabulary $\mathcal{V}$ simply corresponds to the unique set of observed sub-structures in the training data.

The linear co-occurrence relationship of sub-structure is created using modified random sampling scheme, taking account of neighborhoods. That is, whenever we randomly sample a graphlet G, we also sample its immediate neighbors. Then is corpus is generated according to particular graph kernels, for example, for shortest graph kernels, whenever shortestpath sub-structure is generated, all possible

[1]The google tool, *Word2Vec* provides a fast, multi-threaded implementation of both SG and CBOW:https://code.google.com/p/word2vec/

shortest-path sub-structures are also collected that share the same source node, and treat them as co-occurred.

5.5.1 Corpus Generation

Most of the graph kernels [28] are designed for unlabeled graphs or graphs with a complete set of discrete node labels , but graphs with continuous node attributes is catching up fast [38]. But these graph kernels can only handle graphs with complete label or attribute information in a principled manner. These may be efficient, but for specific graph and if flexible, computation is memory and/or time consuming. The propagation kernels by [29] overcomes all the above problems. We exploit the propagation kernels in the deep framework to have similarity index between two graph. The propagation kernels measure the similarity between two graphs by comparing node label or attribute distributions after each step of an appropriate random walk.

The corpus is generated for propagation kernels [29], which is explained in section 4.5. Since, the propagation kernels are based on monitoring how information spreads through a set of given graphs. We use early-stage distributions from propagation schemes such as random walks to capture structural information encoded in node labels, attributes, and edge information. In simple words, propagation kernels intuitively count common sub-distributions induced after each iteration of running inference in two graphs. We collect all possible sub-distributions that share the same source node, and treat them as co-occurred. Therefore, sub-distributions which have similar labels will acquire similar representations. We differ from earlier studies in using propagation kernels, which is efficient than random walk, shortest path, graphlet and Weisfeiler-Lehman subtree kernel.

5.5.2 Model Building and Training

Once corpus is generated, the model is build by using Skip-gram algorithms and trained with negative sampling as explained in the base reference [8].

If s represent an arbitrary sub-structure from a vocabulary $\mathcal{V}$, and ϕ_s represent learned vector representation of s. Matrix $\mathcal{M}$ is calculated such that, each entry on the diagonal is $\mathcal{M}_{ii}$ computed as $\langle \phi_i, \phi_i \rangle$ where ϕ_i corresponds to learned d-dimensional hidden of sub-sequence i and $\mathcal{M}_{ii}$ where i =j and $1 \leq i \leq |\mathcal{V}|$ (resp. j). After computing the $\mathcal{M}$ matrix, it is plugged in Equation 5.2 to get deep graph kernels. The similarity index $SI_{DGP}(G_1, G_2)$ is calculated by normalising the 5.2.

5.5.3 Experiments

We compare our framework with the representative instances of major families of graph kernels as discussed in [8]. The similarity index is calculated as discussed in section 5.5.2. The Matlab code of the propagation kernels was obtained from [29], which was then coded in python. The correlation between transaction network structure and market price is depicted by figure 5.1. It shows better correlation with the network, as compared to similarity index calculated by other graph kernels without using deep framework.

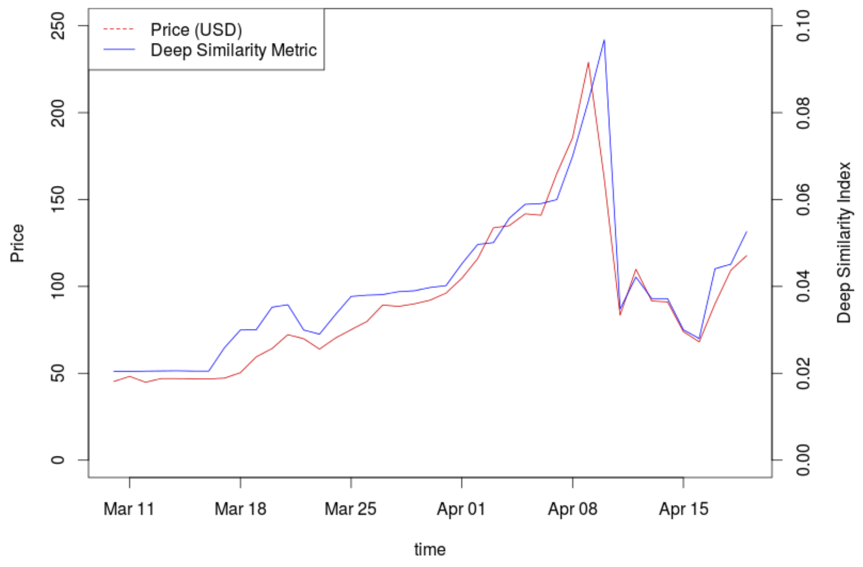


FIGURE 5.1: Deep Similarity

5.6 Result and Discussions

As per Bitcoin's "Metcalfe's Law", there is strong correlation between market cap (transactions/day) and the square of the number of transactions. As our graph is weighted network, where weight represents the transactions between two agents, our results too supports Metcalfe's Law. We are able to show how structural changes in the network accompany significant changes (Quantitative Measure) in the exchange price of bitcoins.

5.7 Conclusions

In this study, we extend the quite novel framework for graph kernels inspired by latest advancements in natural language processing and deep learning. We extend the seminal work [8] by introducing propagation kernels, which takes account of attributed graphs with continuous values. As expected, our deep graph kernels outperforms its best base variants in terms of capturing correlation between newtork structure and market price.

5.8 Summary

This chapter propose a general framework that learns hidden representations of sub-structures used in graph kernels, inspired by deep graph kernels [8]. Then, the framework is demonstrated on propagation kernels, which performs better than normal kernels. The above framework is applied to derive deep variants of string kernels. Then new graph kernel is used to calculate similarity index, which is plotted to find correlation between network structure and market price.

Chapter 6

Conclusion and Future Scope

6.1 Conclusions

1. The enormous blockchain (60 GB as of now) and the newer bitcoin clients indexing the full blockchain using LevelDB had made earlier public available software obsolete. The thesis develops an open source blockchain parsing tools to extract agent resolved data, which can be used to extend the stucked research in bitcoin transaction dynamics.

2. The validation of the data parsed from our tool is then checked by reproducing the "Mathew Effect" phenomenon from prominent paper using their original matlab code, but with our own data. The transaction dynamics in the bitcoin as money flow on the transaction network, support popular hypothesis in economics having roots in preferential attachment, called Matthew effect or the "rich get richer phenomenon".

3. By using the agent resolved data, the case study of silk road arrest is illustrated with the transaction graph with details. It paves the way to understand the important events in bitcoin based on transaction.

4. By defining quantitative measure of transformation over time, in terms of similarity index using different kernels, we infer that there is correspondence between network structure and exchange price in bitcoin. It also concludes that choice of graph kernels as per the graph structure plays important role on problem related to graph isomorphism.

5. We extend deep graph kernels [8], involving propagation kernels, unseen in literature, to solve graph isomorphism problem, which gives extremely agreeable results.

6. By defining quantitative measure of transformation over time, in terms of similarity index using different kernels, we infer that there is correspondence between network structure and exchange price in bitcoin, which is first step in price forecasting.

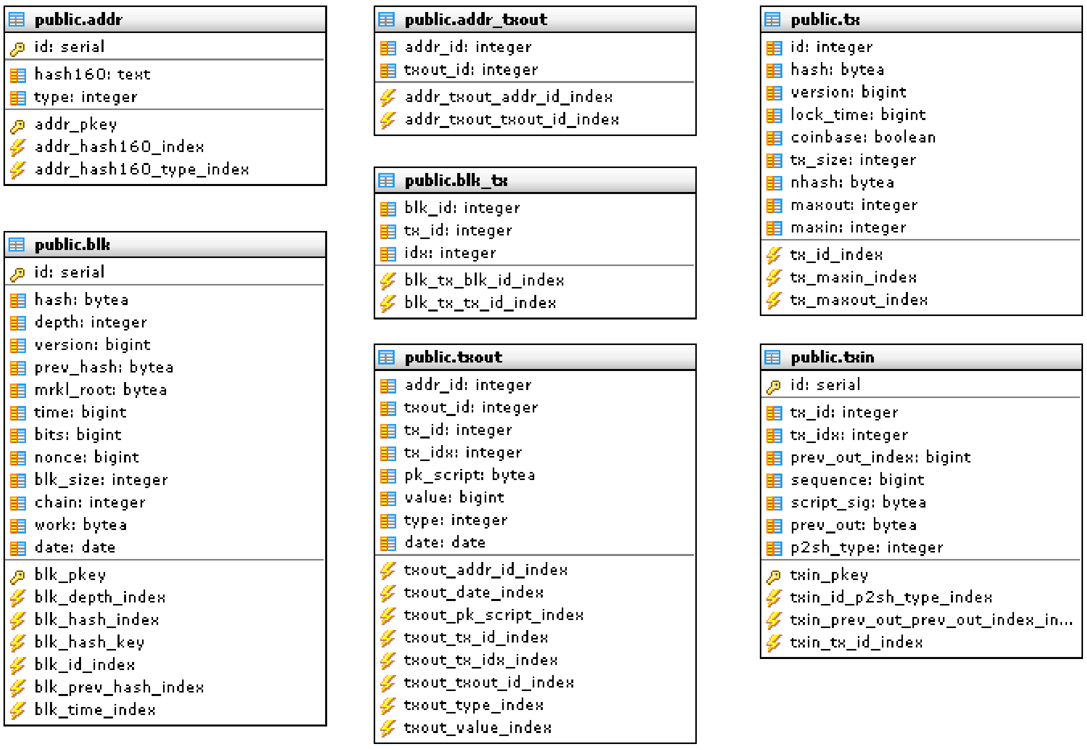
6.2 Future Work

1. On transactions data front, the future work would be to transform the data set into a simplified one indexed by user entity rather than address to do some other meaningful studies, like identifying cluster, market player and linking events to predict bubbles.

2. The possible extension of our work would be to leverage on blockchain network features, as a basis to conduct deep learning learning prediction on the price change of Bitcoin.

Appendix A

Blockchain SQL Schema



```
public.addr
id: serial
hash160: text
type: integer
addr_pkey
addr_hash160_index
addr_hash160_type_index

public.blk
id: serial
hash: bytea
depth: integer
version: bigint
prev_hash: bytea
mrkl_root: bytea
time: bigint
bits: bigint
nonce: bigint
blk_size: integer
chain: integer
work: bytea
date: date
blk_pkey
blk_depth_index
blk_hash_index
blk_hash_key
blk_id_index
blk_prev_hash_index
blk_time_index

public.addr_txout
addr_id: integer
txout_id: integer
addr_txout_addr_id_index
addr_txout_txout_id_index

public.blk_tx
blk_id: integer
tx_id: integer
idx: integer
blk_tx_blk_id_index
blk_tx_tx_id_index

public.txout
addr_id: integer
txout_id: integer
tx_id: integer
tx_idx: integer
pk_script: bytea
value: bigint
type: integer
date: date
txout_addr_id_index
txout_date_index
txout_pk_script_index
txout_tx_id_index
txout_tx_idx_index
txout_txout_id_index
txout_type_index
txout_value_index

public.tx
id: integer
hash: bytea
version: bigint
lock_time: bigint
coinbase: boolean
tx_size: integer
nhash: bytea
maxout: integer
maxin: integer
tx_id_index
tx_maxin_index
tx_maxout_index

public.txin
id: serial
tx_id: integer
tx_idx: integer
prev_out_index: bigint
sequence: bigint
script_sig: bytea
prev_out: bytea
p2sh_type: integer
txin_pkey
txin_id_p2sh_type_index
txin_prev_out_prev_out_index_in...
txin_tx_id_index
```

Figure A.1: SQL Schema of the Blockchain

Bibliography

[1] Lo, S and Wand, C. Bitcoin as Money?. *Federal Reserve Bank of Boston, Current Policy Perspective*. 14-4, Boston. http://www.bostonfed.org/economic/current-policy-perspectives/2014/cpp1404.pdf, 2015.

[2] Kristoufek,L. What are the main drivers of the bitcoin price? evidence from wavelet coherence analysis. *PLoS ONE*, 10(4), 04, 2015.

[3] Donier, J and Bouchaud, J. Why Do Markets Crash? Bitcoin Data Offers Unprecedented Insights. *PLoS ONE*, 10(10): e0139356, 2015.

[4] Ali, R, Barrdear, J, Clews, J and Southgate, J. The economics of digital currencies. *Bank of England Quarterly Bulletin*, page Q3, 2014.

[5] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

[6] Kondor, D, Csabai, I, Szüle, J, Pósfai, M and Vattay, G. Inferring the interplay between network structure and market effects in bitcoin. *New Journal of Physics*, 16(12):125003, 2014.

[7] Zager, L and Verghes, G. Graph similarity scoring and matching. *Applied Mathematics Letters*, 21(1):86 – 94, 2008.

[8] Yanardag, P and Vishwanathan, S.V.N. Deep graph kernels. *In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1365–1374, New York, USA, 2015.

[9] Johannes G, Keeler P, Krzesinski, A and Taylor, P. Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. *CoRR, abs/1505.05343*, http://arxiv.org/abs/1505.05343, 2015.

[10] https://bitcoin.org/en/developer-guide#block-chain

[11] Ron, D and Shamir, A. Quantitative Analysis of the Full Bitcoin Transaction Graph. *IACR Cryptology ePrint Archive*, p. 584, 2013.

[12] Reid, F and Harrigan, M. An Analysis of Anonymity in the Bitcoin System. *Security and Privacy in Social Networks*, pp 197-223, 2013.

[13] Kondor, D, Csabai, I, Pósfai, M and Vattay, G. Do the Rich Get Richer? An Empirical Analysis of the Bitcoin Transaction Network. *PLOS ONE*, 9(5): e97205., 2014.

[14] Spagnuolo, M. Bitiodine: Extracting intelligence from the bitcoin network. *Master's thesis, Politecnico di Milano, Italy*, 2013

[15] Fleder, M, Kester, M and Pillai, S. Bitcoin transaction graph analysis. *CoRR*, abs/1502.01657, http://arxiv.org/abs/1502.01657, 2015.

[16] Babai, L. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, http://arxiv.org/abs/1512.03547, 2016.

[17] Lee, J, Han, W.S, Kasperovics, R and Lee, J.H. An in-depth comparison of subgraph isomorphism algorithms in graph databases. *VLDB*, pages 133–144, 2012.

[18] Gao,X, Xiao, B, Tao, D and Li, X. A survey of graph edit distance. *Pattern Analysis and Applications*, Volume 13, Issue 1, pp 113-129, 2010.

[19] Lerouge, L, Abu-Aisheh, Z, Raveaux, R, Héroux, P and Adam, S. Graph edit distance : a new binary linear programming formulation. *CoRR*, abs/1505.05740, http://arxiv.org/abs/1505.05740, 2016.

[20] Koutra,D, Vogelstein, J.T, and Faloutsos, C. DELTACON: A principled massive-graph similarity function. *CoRR*, abs/1304.4657, http://arxiv.org/abs/1304.4657, 2015.

[21] Li, G, Semerci, M, Yener, M and Zaki, M.J. Effective graph classification based on topological and label attributes. *Stat. Anal. Data Min.*, 5(4):265–283, August 2012.

[22] Mheich, A, Hassan, M, Gripon,V, Khalil, M, Berrou, C, Dufor, O and Wendling, F. A novel algorithm for measuring graph similarity: Application to brain networks. *In Neural Engineering (NER), 2015 7th International IEEE/EMBS Conference on*, pages 1068–1071, April 2015.

[23] Melnik, S, Molina, H.G, and Rahm, E. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. *In Proceedings of the 18th International Conference on Data Engineering, ICDE '02,*, IEEE Computer Society, pages 117–, Washington, DC, USA, 2002.

[24] Lee, J, Kim, G and Yoon, S. Measuring large-scale dynamic graph similarity by ricom: RWR with intergraph compression. *In 2015 IEEE International Conference on Data Mining, ICDM 2015*, Atlantic City, NJ, USA, November 14-17, 2015, pages 829–834, 2015.

[25] Haussler, D. Convolution kernels on discrete structures. *Technical Report*, UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

[26] Bai, l, Zhang, Z, Wang, C, Bai, X and Hancock, E.R. A graph kernel based on the Jensen-Shannon representation alignment. *In Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*, Qiang Yang and Michael Wooldridge (Eds.). AAAI Press 3322-3328, 2015.

[27] Aziz,F, Wilson, R.C. and Hancock, E.R. Backtrackless walks on a graph. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(6):977–989, June, 2013.

[28] Vishwanathan, S.V.N, Schraudolph, N.N, Kondor, R and Borgwardt, K.M. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August, 2010.

[29] Neumann, M, Garnett, R, Bauckhage, C and Kersting, K. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2015.

[30] Gärtner, T, Flach, P and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In Sch ölkopf, B and Warmuth, M.K. editors, *Proceedings of the Annual Conference on Computational Learning Theory*, pages 129–143. Springer, 2003.

[31] Borgwardt, K.M, Ong, C.S, Sch önauer, S, Vishwanathan, S. V. N, Smola, A.J, and Kriegel, H.P. Protein function prediction via graph kernels. *In Proceedings of Intelligent Systems in Molecular Biology (ISMB)*, Detroit, USA, 2005.

[32] Kashima, H, Tsuda, K and Inokuchi, A. Marginalized kernels between labeled graphs. *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press, 2003.

[33] Kang, U, Tong, H and Sun, J. Fast Random Walk Graph Kernel. *SIAM International Conference on Data Mining (SDM)*, Anaheim, California, USA, 2012.

[34] Mahé, P, Ueda, N, Akutsu, T, Perret, J.L, and Vert, J.P. Extensions of marginalized graph kernels. *In Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04*, pages 70–, New York, NY, USA, 2004.

[35] Sugiyama, M and Borgwardt, K. Halting in random walk kernels. *In Advances in Neural Information Processing Systems*,pages 1630–1638. Curran Associates, Inc., 2015.

[36] Kriegel, H.P and Borgwardt, K.M. Shortest-Path Kernels on Graphs. *In Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM '05)*,. IEEE Computer Society, Washington, DC, USA, 74-81, 2005.

[37] Martino, G.D.S, Navarin, N and Sperduti, A. A Tree-Based Kernel for Graphs. *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp 12-24, 2012.

[38] Feragen, A, Kasenburg, N, Petersen, J, Bruijne, D. M and Borgwardt, K. Scalable kernels for graphs with continuous attributes. *Advances in Neural Information Processing Systems 26*, page 216–224, Curran Associates, Inc., 2013.

[39] Yanardag, P and Vishwanathan, S. V. N. A Structural Smoothing Framework For Robust Graph Comparison. *NIPS 2015*, 2134-2142, 2015.

[40] Kondor, R and Pan, H. The Multiscale Laplacian Graph Kernel. *CoRR*, arXiv:1603.06186 http://arxiv.org/pdf/1603.06186v1.pdf, 2016.

[41] Perozzi, B, Al-Rfou, R and Skiena, S. DeepWalk: online learning of social representations. *In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '14)*, ACM, New York, NY, USA, 701-710, 2014.

[42] Kandola, J, Graepel,T and Taylor, J.S. Reducing kernel matrix diagonal dominance using semi-definite programming. *In Proc. Annual Conf. Computational Learning Theory, volume 2777 of Lecture Notes in Computer Science*, pages 288–302, Washington, DC, 2003.

[43] Mikolov, T, Sutskever, I, Chen, K, Corrado, G.S, and Dean, J. Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems*, pp. 3111–3119, 2013.