Bitcoin Data Analytics: Scalable techniques for transaction clustering and embedding generation

Raj Sanjay Shah, Ashutosh Bhatia, Atith Gandhi, Shray Mathur
Dept. of Computer Science and Information Systems,
Birla Institute of Technology and Science Pilani, Pilani, India
Email: {f20171181, ashutosh.bhatia, f20170062, f20171180}@pilani.bits-pilani.ac.in

***Abstract*—Bitcoin [1] provides pseudo-anonymity to its users, leading to many transactions related to illicit activities. The advent of mixing services like OnionBC [2], Bitcoin Fog [3] and Blockchain.info [4] has allowed users to increase their anonymity further. We tackle the pseudo-anonymity of the Bitcoin blockchain by developing a scalable spark based framework to find patterns in the transaction data. We demonstrate the efficacy of our framework by performing exploratory analysis. Furthermore, we show the capabilities of bitcoin-based graph representations and address the issue of user profiling based on unsupervised learning approaches for analysing Bitcoin transactions and users. We convert the transaction graph of the Bitcoin data to contain only wallet IDs and generate graph embeddings using Variational Graph Autoencoder [5]. Additionally, we use explainable-AI techniques and Kohonen self organizing maps to visualize and understand the results obtained from the unsupervised learning methods.**

***Index Terms*—bitcoin, de-anonymization, Graph Convolutional Networks, Variational Graph Autoencoder, Self Organizing Maps, Apache Spark**

I. Introduction

Bitcoin is a decentralized cryptocurrency built on the blockchain technology. The underlying architecture of the Bitcoin blockchain is built on the principles of pseudo-anonymity and immutability. Bitcoin grew in its popularity through its strong security based on mathematical and cryptographic principles. This was due to growing distrust in the government fiscal policies in countries like Venezuela, where high inflation and fluctuations were rampant. The pseudo-anonymity provided by the Bitcoin architecture enables users to form multiple wallets without linking their 'real-life' identities with the wallets. While the blockchain technology offers anonymity, there are various methods and heuristics to determine a user's identity. This can be done either by linking a person's public key to their identity, when the wallets are recipients of coins or by behavioural analysis and study of transaction patterns. One of the popular heuristic discussed in previous literature is proposed by Ales Janda [6]: Whenever there are multiple inputs (multiple wallets pooling their BTCs) in one transaction, then the users of the wallets must be the same person or at least know the identities of each other, therefore matching real life identity of a person to any wallet id can be extended to the identity for all the wallets pooling their BTCs in the same transaction. Immutability, on the other hand, allows anyone to view the history of all the transactions and thus, enables interested parties to track and trace flows of BTCs and understand usage patterns.

The emergence and proliferation of cryptocurrencies has enabled users to circumvent the fiat currencies and avoid scrutiny while spending or receiving money (BTCs). On one hand, this pseudo-anonymity allows users to hide their wealth and retain their privacy, and on the other hand, it allows users to engage in illegal activities without authorities finding the perpetrators. Because of plausible unlawful behaviour, there is great research interest in profiling wallet ids to real life identities, recognizing usage patterns, and determining the extent and scale of such activities occurring on the Bitcoin blockchain.

In one attempt to tackle and find out such illegal behavioural patterns, Akcora et al. [7] focus on detecting and predicting ransomware using topological analysis. While other such research exists in determining techniques to detect specific behaviour like Money laundering [8] [9], price manipulation [10] and theft suspect identification [11], we extend the previous framework by Shah et al. [12]: from targeting the scalability aspect to generating an embedding which can be used for link-prediction and clustering on a sub-graph of bitcoin transactions. This enables us to move beyond simple clustering algorithms as we obtain vectors for each transaction and each wallet id, which captures associations between the nodes. Figure 1 shows a representation of the framework and its extension in detail.

The paper starts with the background of Bitcoin in section II and a brief explanation about the Variational Graph Autoencoders in Section III. Section IV gives a brief idea about the Kohonen Self Organizing Maps. Section V describes the proposed framework, discusses the results and shows how relevant information can be extracted. Section VI concludes the paper with a discussion on future research prospects and directions.

II. Background of Bitcoin

The first mentions of Bitcoin were observed in 2008 when Satoshi Nakamoto released a white paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. Electronic cash is not new, one of the first internet payment service developed by David Chaum called DigiCash was founded in 1989 [13]. It used the concept of Blind Signatures [14] to avoid double-spend. However, it required a server being run by a central

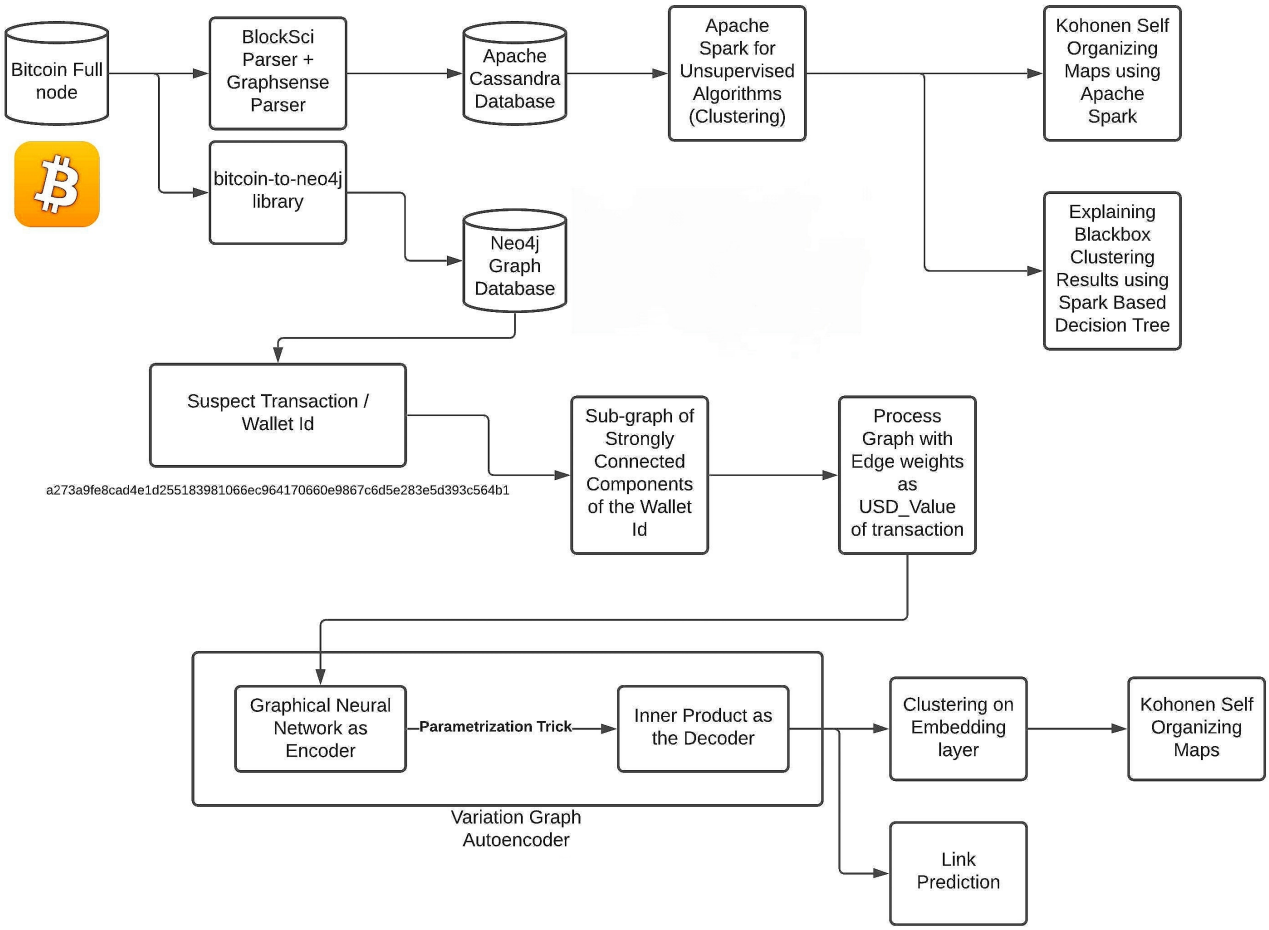

Fig. 1: Bitcoin Analytics Framework

authority and that for everyone to trust them. Another problem was attributing value to digital cash. In the case of DigiCash, to obtain ecash worth $100, one has to take $100 out of their bank account and barter it with the bank that is issuing the ecash. These things were a hassle, so it couldn't gain much popularity, leading to its early demise. So was the fate of other internet payment services at that time.

The coin in Bitcoin can be thought of as a chain of digital signatures, a payer transfers this coin to the payee by digitally signing the hash of previous transaction and payee's public key and appends it to the end of the coin. In this way chain of ownership can be verified. Double-spending is prevented by announcing the transaction to the public and allowing them to come to a consensus on the particular sequence of transactions. A Timestamp Server ensures the chronological validity of transactions to the payee much like a newspaper timestamps the events of a specific period. Once the validity is confirmed, the payee can use this transaction as a reference to spending the acquired BTCs.

After the broadcast of transactions in the system, users check the validity of these transactions. Finally, the valid transactions are included by *Miners* in the Bitcoin blocks. The privilege of adding the block is earned by the miners at the expense of computational work. It requires solving a cryptography puzzle, the solution of which becomes proof of this computational work. Specifically, to generate a new block, miners must find a nonce value that, when hashed with additional fields, results in a value below a given threshold. If such a nonce is found, miners then include it in a block thus allowing any entity to verify the work done by the miner (PoW). Miner is in turn rewarded with BTCs as Mined coins and Transaction fees for all the transaction in the block

III. Graph Structure and the Variational Graph Autoencoder

In this section, we present the graph structure and the mathematics behind the Variational Graph Autoencoders. We use the implementation of the Graphical Neural network-based variational autoencoder proposed by Kipf and Welling [5].

A. The graph structure

There are two types of nodes in the subgraph generated by finding the strongly connected components of a suspect wallet address: Transactions and Wallet addresses. The two different node types hold no semantic meaning for the link prediction tasks, therefore, we remove all the transaction nodes and recalculate the edge weights as shown in 2. Experimental results show that using log of the (recalculated transaction value in terms of united states dollars *(USD_value)* + 1) as

the edge weights performs better than using the USD_value directly. We add a value of one to every transaction to ensure that the log of those values are always positive in nature. We use USD_value instead of BTCs as the pricing of products and services are defined in USD instead of BTCs. We show results on featureless nodes by using an identity matrix as the feature matrix of size (node_size, node_size) in section V. While some literature [15] use blocks as another type of node, we do not use blocks as the transactions or wallet addresses belonging in the same block show no semantic meaning beyond having a similar time stamp.

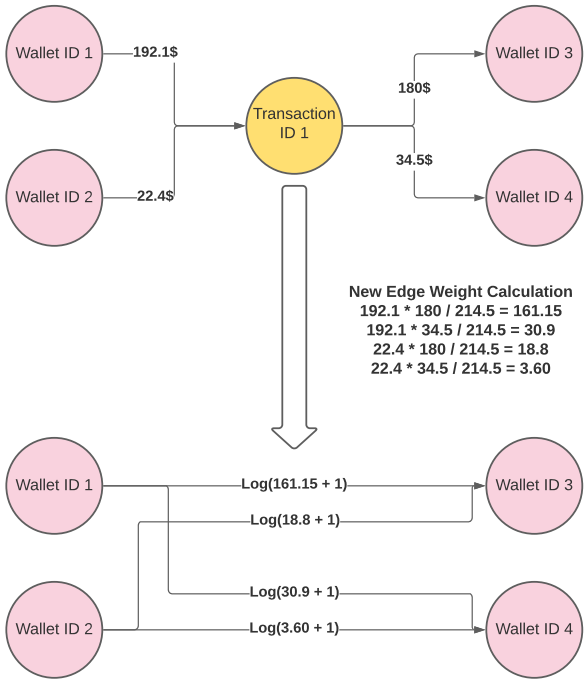

Fig. 2: The graph structure

B. Variational Graph Autoencoder (VGAE)

The VGAE model learns latent variables for the directed graph, which are a part of a distribution rather than a single point. These can be best understood by the set of equations shown below:
Let the feature matrix, X be an identity matrix with of size (node_size , node_size). The adjacency matrix, Adj be a matrix of size (node_size , node_size) consisting of log of the value of bitcoins flowing from one node to another in USD.

$$A = D^{-1/2} Adj D^{-1/2}$$

where D is the degree matrix, and where $D_{ii} = \sum_j A_{ij}$
The Encoder Representation is given by table I. W_0 and W_1 are weight matrices of the respective layers.

TABLE I: Encoder representation

Layers	Equations
1st layer	$X^1\ (GCN(X, A)) = RELU(AXW_0)$
2nd layer	$\mu\ (GCN_\mu(X^1, A)) = RELU(AX^1W_1)$ $log\sigma^2\ (GCN_\sigma(X^1, A)) = RELU(AX^1W_1)$

We then use the parameterization trick: $z = \mu + \sigma * \epsilon$, where ϵ belongs to normal distribution with mean as zero and standard deviation as 1.
The decoder is represented by the inner product between latent variable Z. The output of the reconstructed adjacency matrix:

$$\hat{Adj} = \sigma(ZZ^T)$$

where σ() represents the logistic sigmoid function.

The loss function of the variational graph autoencoder is defined as:

$$Loss = E_{q(Z|X,A)} logp(A|Z) - KL[q(Z|X, A)||p(Z)]$$

The first term $E_{q(Z|X,A)} logp(A|Z)$ gives the reconstruction error for the adjacency matrix. The second term is the Kulback Leibler Divergence, which compares the output of our latent space with a normal distribution $N(0, 1)$ and therefore regularizes our latent space to a gaussian distribution.

IV. Kohonen Self Organizing Maps

Self organizing maps are a type of Artificial Neural Networks whose training is unsupervised in nature and produces a two-dimensional map. It is a method for dimensionality reduction and thereafter uses the the output to visualise the data. It differs from traditional dimsensionality reduction techniques by using a competitive strategy to learn as opposed to error based learning. We have modified the scala-spark implementation shown by Florent [16] to run the self organizing maps using pyspark. This visualization can be used in conjunction of clustering to visualise similarity between clusters.

The architecture used by us can be described by the following points.

1) We use a hexagonal topology for creating and plotting a 2-d lattice of network nodes. The topology consists of 30 * 30 nodes.
2) We calculate the best matching unit by using Euclidean distance (activation distance).
3) The best matching unit's local neighbourhood is determined by the following exponential decay function: $\sigma(t) = \sigma_0 e^{-(t/\lambda)}$. The initial value of σis 1.5.
4) The decay of learning rate is calculated using this equation : $L(t) = L_0 e^{-(t/\lambda)}$. The initial learning rate is 0.7.

Self organizing maps allows us to visualize high dimensional data and understand the similarity between clusters.

V. The Data Analytic Framework

The full node of the Bitcoin blockchain consists of around 310 GB of the block data. The files containing the transaction data and the wallet Ids are stored in BerkeleyDB 4.8 format and the blockchain indexes(headers) are stored in LevelDB. We use the BlockSci v-0.5 parser [17] and then use the graphsense library [18] to convert the data into the CassandraDB. We also create a neo4j graph database using the bitcoin-to-neo4j library [19]. We use both of these databases at the same time, allowing us to have more flexibility. The main advantage of using the above mentioned libraries is that they let us update our data in an incremental fashion.

```
Cluster Centers:
[-0.02171836 -0.00333789 -0.00209213]
[26.24360722  0.12227284  0.13861157]
[-8.50495906e-02 -4.29836695e-02  4.45184680e+02]
[-7.06100631e-02  1.63030607e+02  6.65573662e-01]
cluster:  0
+-------+-----------------+------------------+------------------+--------------------+--------------------+----------------------+
|summary|      input_count|      output_count|   input_total_usd|  scaled_input_count| scaled_output_count|scaled_input_total_usd|
+-------+-----------------+------------------+------------------+--------------------+--------------------+----------------------+
|  count|          2718372|           2718372|           2718372|             2718372|             2718372|               2718372|
|   mean|1.940876009611635|2.4413031034751684|18446.952727438354|-0.02167313767564...|-0.00337015923313...|  -0.00208175412503...|
| stddev|5.529908358995659| 6.550164091945536| 325991.1751504966|  0.5072776762385027|  0.4267893001421959|   0.35000318159599525|
|    min|              1.0|               1.0|            0.0379|-0.10798295780269665|-0.09728120240550055|  -0.02188744020611...|
|    max|            145.0|            1153.0|      1.30615648E8|  13.101636557774633|    74.9636282438763|    140.21470740581086|
+-------+-----------------+------------------+------------------+--------------------+--------------------+----------------------+

cluster:  1
+-------+------------------+------------------+------------------+------------------+--------------------+----------------------+
|summary|       input_count|      output_count|   input_total_usd|scaled_input_count| scaled_output_count|scaled_input_total_usd|
+-------+------------------+------------------+------------------+------------------+--------------------+----------------------+
|  count|              2251|              2251|              2251|              2251|                2251|                  2251|
|   mean|288.26254997778767|  4.36961350510884|149487.99556708123|26.243607224704835| 0.12227283767778305|    0.1386115695281551|
| stddev|157.71724042019437|25.916914095828528| 808712.1416616305|14.467949562483332|  1.6886693941615012|    0.8682806289041615|
|    min|             146.0|               1.0|           11.0453|13.193370026632808|-0.09728120240550055|  -0.02187562201803...|
|    max|            1100.0|             592.0|       1.8005196E7|100.70709931733262|  38.410529112483864|     19.30954344618217|
+-------+------------------+------------------+------------------+------------------+--------------------+----------------------+

cluster:  2
+-------+-----------------+------------------+--------------------+--------------------+--------------------+----------------------+
|summary|      input_count|      output_count|     input_total_usd|  scaled_input_count| scaled_output_count|scaled_input_total_usd|
+-------+-----------------+------------------+--------------------+--------------------+--------------------+----------------------+
|  count|               12|                12|                  12|                  12|                  12|                    12|
|   mean|            1.25|1.8333333333333333| 4.146631226666667E8|-0.08504959058815266|-0.04298366953058605|    445.18468032530336|
| stddev|0.621581560508061|0.3892494720807615|2.6676465067316618E7|  0.0570198327236826|0.025362343208217914|    28.641412280515702|
|    min|              1.0|               1.0|         3.77178912E8|-0.10798295780269665|-0.09728120240550055|     404.9394405979058|
|    max|              3.0|               2.0|         4.34792032E8| 0.07548397991365513|-0.03212416295560315|     466.7962540296267|
+-------+-----------------+------------------+--------------------+--------------------+--------------------+----------------------+

cluster:  3
+-------+------------------+------------------+------------------+--------------------+--------------------+----------------------+
|summary|       input_count|      output_count|   input_total_usd|  scaled_input_count| scaled_output_count|scaled_input_total_usd|
+-------+------------------+------------------+------------------+--------------------+--------------------+----------------------+
|  count|                54|                54|                54|                  54|                  54|                    54|
|   mean|1.4074074074074074|2504.6111111111113|640297.7960907407|-0.07061006308269906| 163.03060673146263|    0.6655736618684439|
| stddev|1.5962044428394255| 1868.074100755093|747461.7971556883| 0.14642537054849247|  121.7181778782312|    0.8025186786273375|
|    min|               1.0|            1256.0|          198.4932|-0.10798295780269665|  81.67480330721573|  -0.02167436699524...|
|    max|              12.0|            6830.0|       2377905.25|  0.9010851996372382|  444.8601412009438|     2.531170596976847|
+-------+------------------+------------------+------------------+--------------------+--------------------+----------------------+
```

Fig. 3: K-means algorithm: Clusters

A. Top down approach to clustering using Apache Spark Technology

After obtaining our data in the NoSQL format, we are equipped to run queries and different algorithms on our data. We have chosen Apache spark, CassandraDB, and Neo4j GraphQL technologies for our analytics as all of them support distributed and scalable features required to tackle a large dataset of the size of the Bitcoin blockchain. The only draw back is that the different database forms require around 1200 GB of space (4 times the size of the blockchain). To appropriately demonstrate the efficacy of our framework, we solve the following problem statement: *What is the percentage of transactions in the network used by the mixing services around the world?*

Mixing services are used to obfuscate money trails, and flows and therefore act as money laundering services. Intermediate transactions of the mixing services are characterized by large number of inputs and large number of outputs for effective mixing [9] [20]. The number of intermediate transactions are far greater than the end points of a mixing service graph for an efficient mixing.

To estimate the upper bound on the number of transactions involved in mixing services, we run the Principle Component Analysis to remove correlation between the data and subsequently run the K means algorithm using the Spark framework on 2,722,114 transactions with the value of K as 4 using a cluster. These transactions are the all the transactions occuring in the first ten days of 2020. The features selected are input_count, output_count,input_total_usd, fee_per_kwu and fee_per_kb. We also normalise these features to give equal weight-age to all the features. We get the output as seen in figure 3.

1) Cluster 0 : This cluster represents the majority of the transactions. These transactions maintain a proportionality with the number of inputs and number of outputs to the value of the transaction.
2) Cluster 1 : This cluster represents those transactions with a large number of inputs.
3) Cluster 2 : This cluster represents transactions with less number of inputs, less number of outputs but large transaction value.
4) Cluster 3 : This cluster represents transactions with less number of inputs, large number of outputs.

Cluster 0 is the only cluster that contains transactions with both large number of inputs and large number of outputs and thus has a higher probability of having transactions that are part of the mixing services. We filter cluster 0 to have only those records with more than threshold **t** input_count and

output_count. We set **t** to a value of 5, then to 10 and finally to 30. The results of our analysis are seen in table II.

TABLE II: Percentage of transactions used in mixing services based on different threshold values

Case	Threshold value	Number of transactions	Percentage of total transactions
1	5	10,515	0.386%
2	10	3,601	0.13%
3	30	763	0.028%

The percentage of transactions used in mixing services ranges from 0.028 to 0.386%. Figure 4 shows the SOM output of the clustering done in figure 3. We observe that cluster one consists of a wide variety of transactions and cluster zero occupy a particular area of the maps.

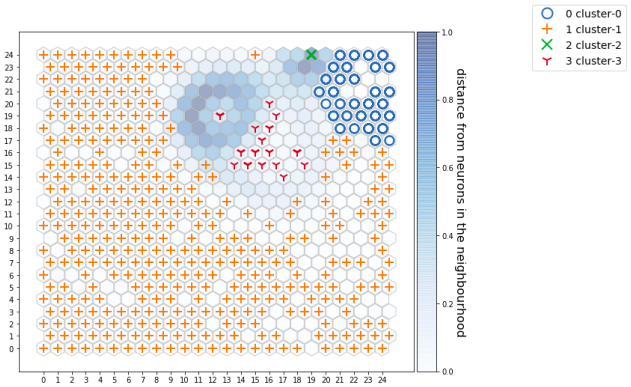

Fig. 4: Self organizing map output

1) Explainable K-Means Clustering: Explainable AI talks about understanding results of black box algorithms like Deep networks and unsupervised learning methods like K-means clustering. We understand the decision boundaries of the clustering and realize the different parameters on which the clusters are formed. We extended the work done by Frost et al. [21] to work in a spark based framework. We modify the traditional distributed version of decision tree so that we maintain a trade off between depth of the tree and the accuracy, and the explainability of the clustering in a low cost manner. The tree formed in the clustering in the above step is shown in figure 5. We have limited the depth of the tree to a maximum of length five considering the trade of between the Accuracy and the construction cost. We observe that only two of the classes are represented in the decision tree. When we build the tree to a length of nine, then we observe that all the clusters have representation in the tree.

B. Embedding Generation and Link Prediction

We process the Neo4j graph database to obtain the adjacency matrix as defined in Section III. Thereafter we construct a sub-graph consisting of the nodes and edges that are a part of the strongly connected components of a particular suspect address. Then we use Variational Graph Autoencoders with the configuration described in Section III for link prediction and embedding generation. We also implement a simple Autoencoder and a Structural Deep Network Embedding [22] to compare the results for link prediction.

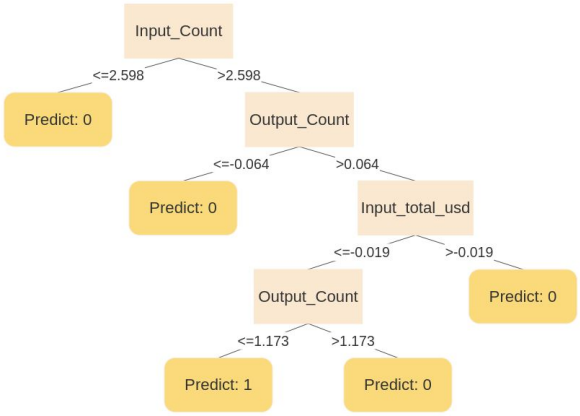

Fig. 5: Explainable K-means Clustering

1) Link Prediction: The model recreates a modified version of the adjacency matrix with all the edge weights set to one or zero, in accordance to the presence or the absence of an edge. While the model tries to reconstruct the modified adjacency matrix, we use the original adjacency matrix for the message passing abilities of the Graphical Neural Networks. This allows us to capture the graph topology accurately. The size of the first hidden layer is 32 and the size of the second hidden layer is 16, thus the latent space consists of 16 dimensions. Experimental results work best on these sizes of the hidden layers. We run three models: Graph Autoencoder (GAE), Structural Deep Network Embedding(SDNE), Variational Graph Autoencoder(VGAE) for 200 epochs and the results for them are seen in table III. From this table, we observe that Variation Graph Autoencoder gives competitive performance as compared to Graph Autoencoder and Structural Deep Network Embedding.

TABLE III: Link Prediction

Model	GAE	SDNE	VGAE
Test set: Area under the curve (ROC)	82.5780%	61.4553%	**93.4462%**
Test set: Average Precision	76.3090%	58.3181%	**91.4099%**

The receiver operating characteristic curve for the variational graph autoencoder can be observed in the figure 6.

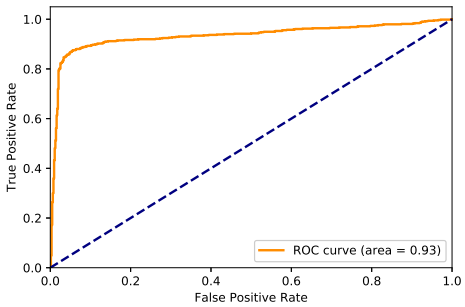

Fig. 6: ROC Curve for VGAE

2) Embedding Generation: We use the latent space formed in the bottleneck framework of the Variational Graph Autoencoder. We judge the quality of the node embeddings

with the help of self organizing maps. We cluster the output of the embeddings using K-means algorithm and thereafter visualize these clusters using Self organizing maps. Based on the observations of the result of the elbow test in figure 7, we set the value of K as 17. The outcomes of the self organizing maps are seen in figure 8. The most notable observation from

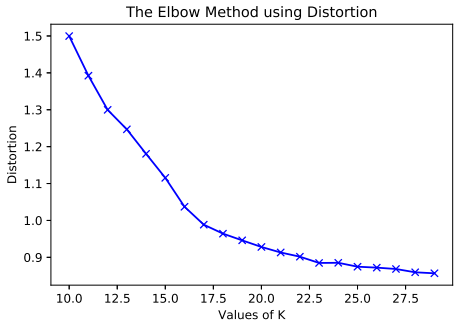

Fig. 7: K Means clustering: Elbow Test

the visualizations of the embeddings in figure 8 is that the clusters are distinct and well separated. Analysing the sub-graph consisting of the nodes of a particular cluster, gives insights to interested persons about the structure and the properties of the graph that lead to the following embedding generation. These properties can be linked to the real-life identities and patterns of suspect users of the Bitcoin blockchain.

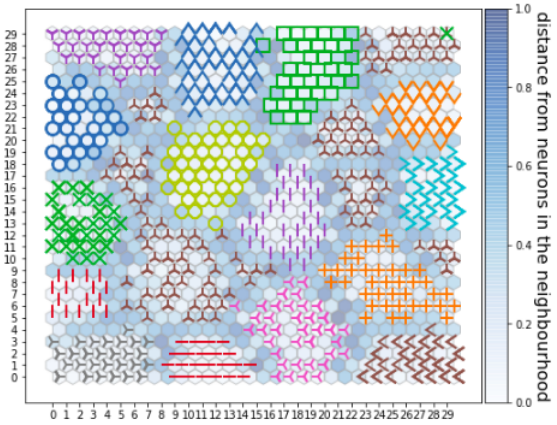

Fig. 8: Self organizing map output for the generated embeddings

VI. Conclusion and Future Research Prospects

The proposed framework provides a scalable and fault-tolerant architecture to perform exploratory and investigatory analysis of the Bitcoin blockchain. The framework consists of a top-down approach using distributed computation technologies for overall analysis which supports various generic algorithms and querying methods. At the same time, the framework provides granularity for a detailed analysis of particular portions of the Bitcoin transaction graph with the help of graph embeddings and predicts associations between the nodes of the graph by using link prediction methods. Further research can be done in forming homogeneous transactions as well as wallet-id graphs. The graphs can be used for generating embeddings which take into account certain node features. A combination of an experimental approach coupled with mathematical probabilities to compute proportions of BTCs transmitted from one transaction to another can be used to form different heuristics to adequately assign weights to edges of the homogeneous graphs.

References

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] OnionBC. Onionbc. [Online]. Available: http://6fgd4togcynxyclb.onion/

[3] B. Fog. Bitcoin fog. [Online]. Available: http://www.bitcoinfog.info/

[4] Blockchain.info. Blockchain.info. [Online]. Available: https://blockchain.info/q/

[5] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[6] WalletExplorer. smart bitcoin block explorer. [Online]. Available: http://www.WalletExplorer.com

[7] C. G. Akcora, Y. Li, Y. R. Gel, and M. Kantarcioglu, "Bitcoinheist: Topological data analysis for ransomware detection on the bitcoin blockchain," 2019.

[8] J. Crawford and Y. Guan, "Knowing your bitcoin customer: Money laundering in the bitcoin economy," in *2020 13th International Conference on Systematic Approaches to Digital Forensic Engineering (SADFE)*, 2020, pp. 38–45.

[9] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the bitcoin ecosystem," in *2013 APWG eCrime Researchers Summit*, Sep. 2013, pp. 1–14.

[10] P. Timothy, "To the moon: A history of bitcoin price manipulation," 2020.

[11] Y. Wu, A. Luo, and D. Xu, "Identifying suspicious addresses in bitcoin thefts," *Digital Investigation*, vol. 31, p. 200895, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287619302233

[12] R. S. Shah and A. Bhatia, "Bitcoin data analytics: Exploring research avenues and implementing a hadoop-based analytics framework," in *Web, Artificial Intelligence and Network Applications*, L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, Eds. Cham: Springer International Publishing, 2020, pp. 178–189.

[13] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[14] S. Bag, S. Ruj, and K. Sakurai, "Bitcoin block withholding attack: Analysis and mitigation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1967–1978, Aug 2017.

[15] A. Sharma and A. Bhatia, "Bitcoin's blockchain data analytics: A graph theoretic perspective," 2020.

[16] FlorentF9. Spark-scala based som implementation. [Online]. Available: https://github.com/FlorentF9/sparkml-som

[17] H. Kalodner, S. Goldfeder, A. Chator, M. Möser, and A. Narayanan, "Blocksci: Design and applications of a blockchain analysis platform," 2017.

[18] B. Haslhofer, R. Karl, and E. Filtz, "O bitcoin where art thou? insight into large-scale transaction graphs." in *SEMANTiCS (Posters, Demos)*, 2016.

[19] in3rsha. Bitcoin-to-neo4j. [Online]. Available: https://github.com/in3rsha/bitcoin-to-neo4j

[20] G. D. Battista, V. D. Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia, "Bitconeview: visualization of flows in the bitcoin transaction graph," in *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, Oct 2015, pp. 1–8.

[21] N. Frost, M. Moshkovitz, and C. Rashtchian, "Exkmc: Expanding explainable k-means clustering," 2020.

[22] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1225–1234. [Online]. Available: https://doi.org/10.1145/2939672.2939753