Graph analysis techniques and applications to bitcoin forensics

Phetsouvanh, Silivanxay

2019

Phetsouvanh, S. (2019). Graph analysis techniques and applications to bitcoin forensics. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/106803

https://doi.org/10.32657/10220/49668

GRAPH ANALYSIS TECHNIQUES AND APPLICATIONS TO BITCOIN FORENSICS

SILIVANXAY PHETSOUVANH

School of Computer Science & Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Supervisor: Assoc. Prof. Anwitaman Datta

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

15-Aug-2019 .....

Date

Silivanxay Phetsouvanh

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism to the best of my knowledge and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

15 August 2019 .....

Date

..... Assoc. Prof. Anwitaman Datta

Authorship Attribution Statement

This thesis contains material from 4 paper(s) published in the following peer-reviewed conference(s) where I was an author, as well as some yet to be published work.

Chapter 3's Section 3.2.3 and Subsection 3.3.3.1 are published as F. Oggier, S. Phetsouvanh, and A. Datta. Entropic centrality for non-atomic flow networks. In *International Symposium on Information Theory and Its Applications (ISITA)*, 2018.

The contributions of the co-authors are as follows:

- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta initiated problem formalization for split-and-transfer entropic centrality for graphs.
- Assoc. Prof. Frédérique Oggier provided mathematic formulation and analysis.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I designed the algorithms necessary for carrying out the computations.
- Assoc. Prof. Frédérique Oggier and I carried out the algorithm implementation.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I identified necessary datasets to experiment with.
- I conducted the experiments guided by advices from Assoc. Prof. Anwitaman Datta.
- I curated the data and created the plots based on the experiment results.
- Assoc. Prof. Frédérique Oggier, Assoc. Prof. Anwitaman Datta and I carried out the analysis and interpretation of the experiment results.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I wrote the paper.

Chapter 5, except Section 5.3 is published as F. Oggier, S. Phetsouvanh, and A. Datta. Entropy-based graph clustering - A simulated annealing approach. In *International Symposium on Information Theory and Its Applications (ISITA)*, 2018.

The contributions of the co-authors are as follows:

- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta formalized the problem for entropy-based graph clustering.
- Assoc. Prof. Frédérique Oggier provided the mathematical formalism.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I designed entropy-based graph clustering algorithms, and explored several possible choices and came up with multiple preliminary implementations.
- From the choice of algorithms, we zeroed in on a simulated annealing based algorithm, which was finally implemented and fine tuned by Assoc. Prof. Frédérique Oggier and myself.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I identified suitable datasets to be used in the experiments.
- I carried out the experiments, guided by advices from Assoc. Prof. Anwitaman Datta.
- I curated the data and created the plots based on the experiment results.
- Assoc. Prof. Frédérique Oggier, Assoc. Prof. Anwitaman Datta and I analysed and interpreted the experiment results.
- Assoc. Prof. Frédérique Oggier, Assoc. Prof. Anwitaman Datta and I wrote the paper.

Chapter 6 is published as F. Oggier, S. Phetsouvanh, and A. Datta. BiVA: Bitcoin network visualization and analysis. In *IEEE ICDM Demo on Data Mining Workshop (ICDMW)*, 2018, and S. Phetsouvanh, F. Oggier, and A. Datta. EGRET: Extortion graph exploration techniques in the Bitcoin network. In *IEEE ICDM*

Workshop on Data Mining in Networks (DaMNet), 2018.

Note that one of the paper explores the underlying theory and analysis and is more technical in nature, while the other paper is in fact a demo. They thus complement each other. The contributions of the co-authors for both the papers are as follows:

- I started the problem formulation of Bitcoin address aggregation.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta formalized the problem based on network analysis and network visualization.
- Assoc. Prof. Frédérique Oggier formulated the mathematical model.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I designed several algorithms for network analysis.
- Assoc. Prof. Frédérique Oggier and I worked on the implementation of the algorithms and integration with visualization tools.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I came up with related datasets for case study based experimentation and validation.
- Assoc. Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I designed and carried out the experiments.
- Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I did the analysis of the results.
- Prof. Frédérique Oggier and Assoc. Prof. Anwitaman Datta and I wrote the papers.

15-Aug-2019 .....

Date

Silivanxay Phetsouvanh

Acknowledgements

I would like to express my greatest gratitude to my advisor Assoc. Prof. Anwitaman Datta for his continuous support during my Ph.D. studies. His patient guidance, encouragement, and immense knowledge are precious to me and beyond what words can express. I really appreciate having a supervisor who cares so much about my work, who always responded to my queries promptly, and who did not give up on me despite my slow progress.

I would also like to thank all my thesis advisory committee members, especially Assoc. Prof. Frédérique Oggier who guided me with the same care as a co-supervisor. Completing my thesis would have been all the more difficult were it not for her support, encouragement, and for her patiently explaining to me the mathematics and theory on which the ideas of this thesis are founded.

I will also like to thank Dr. Alwen Tiu for his guidance and collaboration during the early phase of my PhD. Even though I eventually decided to pursue a different line of research for my thesis, the experience gained from working with him is invaluable.

Finally, I would like to thank the School of Computer Science and Engineering at Nanyang Technological University Singapore for providing me an opportunity to take this study under NTU research scholarship, and for providing financial support to attend a conference and present some of my works.

Abstract

Flow based networks have been studied over the years, which have considered different interpretations of a flow, for instance where a flow may be based on *transfer* (e.g., package delivery), or *serial replication* (e.g., one-to-one gossip), or *parallel duplication* (e.g., epidemic spread). However, the study of flows which may split while conserving the overall volume at each split is relatively sparse. In this thesis we look at certain aspects of such *split-and-transfer* based flows with volume conservation. Specifically, we (i) design general purpose flow based graph analysis techniques and demonstrate their applicability by studying a wide range of real networks, including the network induced by Bitcoin transactions; and (ii) design analysis techniques which are specific to the Bitcoin network, and showcase the efficacy of these techniques in conjunction with our novel flow-based graph analysis algorithms with case studies.

Pertaining the general purpose graph analysis, we first consider the notion of entropic centrality which measures how central a node is in terms of how uncertain the destination of a flow starting at this node is: the more uncertain the destination (i.e., larger the spread of the flow), the more well connected and thus central the node is deemed. Prior works on entropic centrality implicitly assumed that the flow is indivisible, and at every node, the flow is transferred from one edge to another. In this thesis, we propose a split-and-transfer flow model for entropic centrality, where at every node, the flow can actually be arbitrarily split across choices of neighbours, while conserving the overall volume. This interpretation of volume conserved non-atomic flows better reflects certain applications, including Bitcoin transactions. We consider two variations. In one, we constrain that a down-stream flow cannot revisit a node that has already been visited upstream (no cycles), and show how to relate this model to an equivalent transfer based entropic centrality one for computational tractability. We then relax the 'no cycle' constraint, and explore a Markov process of random walk which simplifies the computation. These entropic centrality models can easily accommodate directed and weighted graphs. Based on the latter variant of (Markov) entropic centrality model, we next design a 2-stage clustering algorithm, which in the first stage is informed by the entropic centrality to detect local communities based on random walks, and then performs a bottom-up iterative aggregation.

We furthermore explore graph clustering by using an entropic distance based approach as an alternative to distribution of flow based approach as described above. To

that end, we revisit a Renyi entropy based measure introduced originally for image clustering, and study its application to graph clustering. To effectuate Renyi entropy based graph clustering, we propose a simulated annealing algorithm, as well as a hierarchical approach. While the simulated annealing approach is practical for a range of graphs, our early results indicate that it (or at least our realization) does not scale for large graphs. This was an immediate motivation for the exploration of a hierarchical agglomerative approach which is naturally amenable to (not explored in this thesis) distribution and parallelization.

Finally, we demonstrate the use of the above novel information theory based algorithms to derive macroscopic information from the Bitcoin network. We augment these macroscopic general purpose graph algorithms with Bitcoin specific graph analysis algorithms that can be used to carry out microscopic analysis of a zoomed in Bitcoin subnetwork. Specifically, we study path confluence based address aggregation and Bitcoin flow traceback/forward mechanisms, which are centred around Bitcoin addresses of potential interest. In addition to that, we designed and developed a visualization aided graph mining tool, BiVA, which enables Bitcoin network data exploration, visualization of subgraphs around nodes of interest, and integrates both standard and new algorithms, including the aforementioned general algorithms for flow based centrality and clustering, and the Bitcoin microscopic graph analysis algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Scope of this Thesis

This thesis involves a subset of topics typically studied under the umbrella on *network science*, with particular emphasis on graph algorithms to study network properties and for mining and visualization, driven extensively by information theoretic tools. The specific network models we study have often been inspired by an immediate objective to understand the Bitcoin ecosystem, nevertheless, the ideas so explored mostly have wider applicability, and contribute to the general body of network science literature, and we illustrate this in our work by also exploring other kinds of networks using the tools we design. To paraphrase [14], network science is the study of a real world phenomenon using a network representation, and the real world phenomenon can be a naturally occurring one (e.g., biological network) or be a man-made artefact (e.g., social, technological), in order to facilitate understanding and arguably prediction or maneuvering of said phenomenon. Network science is driven by ideas from multiple disciplines, including graph theory, statistical mechanics, data mining, social models, inference modeling and information visualization.

A network representation naturally implies a graph. A graph can be considered to comprise a set of vertices with edges which represent relationship among vertices. Mathematically, the relationships can be represented by the notion of relation which is defined as a subset of the Cartesian [15] product of two sets. For example, Figure 1.1 shows how people's relationships are modeled from friendship relation which is a subset of Cartesian product of the set of people (consisting of Alice, Bob, Carol, and Dave) to itself. In this specific instance, based on the graph representation, we

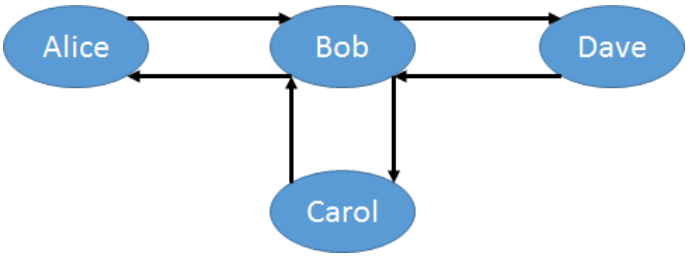


FIGURE 1.1: A graph comprises a set of people, {Alice, Bob, Carol, Dave} as vertices and their relationships modeled from friendship relation {(Alice, Bob), (Bob, Alice), (Bob, Carol), (Carol, Bob), (Bob, Dave), (Dave, Bob)}.

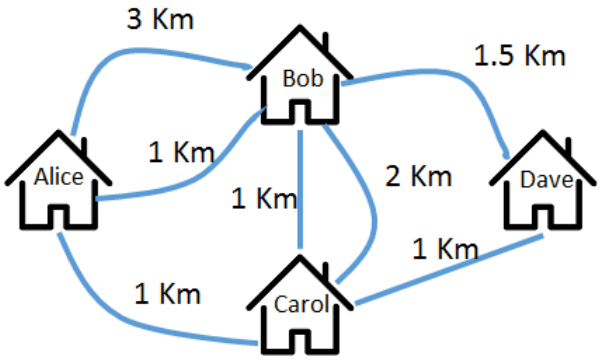


FIGURE 1.2: A graph comprises a set of people's houses, {Alice, Bob, Carol, Dave} as vertices and set of direct paths as links between houses. Each number represents the distance of the corresponding path in kilometres.

are able to observe visually that Bob is the most important person and acts as a connector for everyone in this network comprising friendships as relationships. In fact, objective metrics can be defined to better understand the network properties beyond visual inspection, for instance, the centrality of nodes, influence of nodes, clustering coefficient, network diameter, connectedness, to name some.

Depending on the nature of underlying data and relationships, different graph models may be applicable. One basic distinction is whether the graph has directed edges (digraph), or undirected (or bi-directed) edges. A directed graph contains a set of directed edges which represent asymmetric relationship, for example, our Chapter 1 has references pointing to our Chapter 3, but our Chapter 3 does not necessarily have references pointing back to our Chapter 1; while an undirected graph contains a set of (undirected) edges typically representing symmetric relationship. For instance, a friendship graph, in Figure 1.1, is an example of undirected graph where (undirected) edge {Alice, Bob} can be interpreted as two directed edges, {Alice, Bob} and {Bob, Alice}. Another is whether the graph edges are unweighted, or weighted; where the edge weight could represent certain

information depending on the context - e.g., cost of a link, capacity of a link, etc. Furthermore, one may consider multi-graphs, where there are multiple edges adjacent to a same pair of vertices. For example, the graph in Figure 1.2 captures direct connection between houses of Alice, Bob, Carol, and Dave, and it is an example of an undirected multi-graph where each vertex represents a house, and each undirected edge represents a direct path between two houses. These basic models can then be generalized into more sophisticated models, where there can be multiple kinds of vertices (multi-mode graphs), or multiple kinds of edges. For example, in the context of Bitcoin [16] networks, we will subsequently consider 2-mode graphs that have nodes that represent wallet addresses, and Bitcoin transactions. People can have multiple kinds of relations amongst themselves, for instance, in a bibliographic data set, the relationships could encapsulate co-authorship, cited by, co-participation in scientific events, employment by same employer, as example different relationships. Other, even more complicated models too exist, but are not studied in this thesis, and hence we do not further enumerate them.

As opposed to the graph theoretic classification of graphs described above, there are other ways to classify graphs based on the nature of origin or application of said graphs. Authors in [17] suggest four categories, paraphrased below. Note that it is neither exhaustive nor unique or even mutually exclusive, and other categories could be envisaged.

Biological network graphs: Examples include (i) ecological networks such as water cycle, food web and so on; (ii) neural networks which consist of neurons as nodes and the synapses as links between them. These neural networks are studied in order to analyze and to understand how a brain works based on a connection change when one learns new things; (iii) protein-protein interaction network where proteins are denoted as nodes, and interactions, which perform biological functions in cells, are denoted as links between them. These networks are used to study molecular interaction chains or pathways; (iv) network induced by spread of diseases. For instance, a network in epidemiology where each node represents individual and each directed link represents the transfer of disease infection from one to another. Finding important nodes in this graph can be useful in analyzing, understanding and eventually mitigating the spread of the disease.

Social network graphs: These could include friendship networks (see Figure 1.1), business networks where each vertex represents a company and each edge represents goods or services that a company purchases from another, and all social

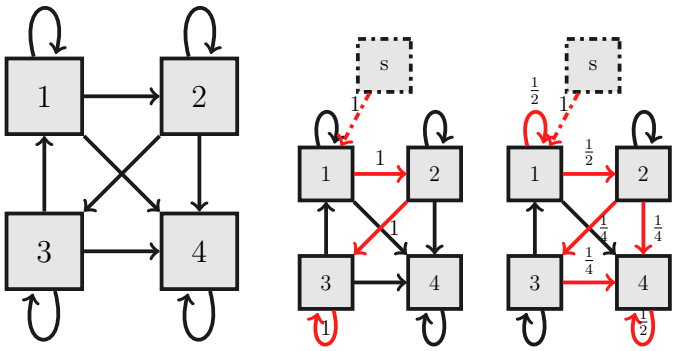


FIGURE 1.3: A small directed connected graph $G = (V, E)$, with $V = \{1, 2, 3, 4\}$, $E = \{(1, 1), (1, 2), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\}$. Two instances of flow propagation are given on the right (each instance will then happen with a given probability). Flow values are shown on the edges.

community and interaction networks. Moreover, social network graphs encompass graphs that capture relationships such as who follows whom, who talks with whom, who influences whom and all interactions in social structure. Social network graphs may be used to observe how communities develop, how topics become hot, and how information flows and so on.

Information network graphs: These comprise (i) preference networks which capture the relationship between users and their preference items; (ii) citation networks where scientific papers (as nodes) cite others. These networks are used to analyze scientific papers' impact, relevance and other aspects; (iii) word classes in dictionary such that a word is denoted as a vertex and relations of word classes are denoted as edges; (iv) the World Wide Web network where the link of the web represents hyperlink to another web. Link graphs are used to analyze popularity of websites, relevance of web pages et cetera.

Technological network graphs: These encompass (i) utility networks (water network, the Internet, power grid, telephone networks and etc.) where each node represents connection point and each link represents a wire/pipe connecting two connection points. Utility networks are studied in order to minimize infrastructure cost while still maintaining high reliability; (ii) transportation networks such as airline routes, networks of roads, railways, pedestrian traffic and so on. One example is road network where nodes represent intersections and links represent roads between them. Such networks are analyzed in order to find best routes between two locations in term of cost/time. They are also used for studying traffic light timings or patterns, and many aspects of transportation.

We are interested mainly in a type of network which can be viewed as an (information) flow network [5, 18]. Existing works consider networks in which a flow may be based on *transfer*, where an item or unit flows in an indivisible manner (e.g., package delivery), or by serial replication, such that both the node that sends the item and the one that receives it have the item (e.g., one-to-one gossip), or parallel duplication, where an item can be transmitted in parallel through all outgoing edges (e.g., epidemic spread). This flow based characterization of network [5, 18] provides the foundations for the thesis, but we build upon these initial interpretations of flow, and add a further concept of non-atomic (but volume conserved) flows. Figure 1.3 shows examples of some variants of flow based networks: on the left we have an underlying graph indicating connectivities, in the middle, we highlight an instance of transfer based flow where a flow starts from node 1 and ends at 3 tracing some of the edges of the underlying graph, while, a case where a flow is actually not just transferred, but is also potentially split among outgoing edges, with the possibility to partly remain at any intermediate node (self-loop is used to represent such partial flow terminations) it encounters is highlighted in the right.

In this context, we design general purpose information flow based graph algorithms (particularly for node centrality and graph clustering), and demonstrate their applicability by studying a wide range of real world networks, including but not limited to (i) a network derived from wire-tapped communication of cocaine dealers [6]; (ii) a network of Maine airports and the flow of passengers through them [19]; (iii) a network representing cross-shareholding among companies listed in Tehran stock exchange [20]; and (iv) sub-networks [21–24] induced by Bitcoin [16] transactions.

After demonstrating the applicability of our general purpose graph algorithms on subnetworks induced by Bitcoin transactions, we look at specific algorithms designed for analysis and forensic of Bitcoin. Early tools for Bitcoin analysis are plenty, e.g. [16, 25–29], exploiting simple heuristics clubbing co-paying addresses, and guessing the address used to collect the change in a transaction (e.g., [30, 31]). The introduction of mixing, where multiple users come together (possibly via a service, or in a decentralized manner) to create multi-input multi-output transactions render the simple heuristics ineffective, by making it difficult to trace the flow of money, since mixing obfuscates the linkage among inputs and outputs of a transaction. It is claimed in [27] that clubbing multiple inputs together in a cluster is surprisingly effective, but false positives are indeed caused by transactions

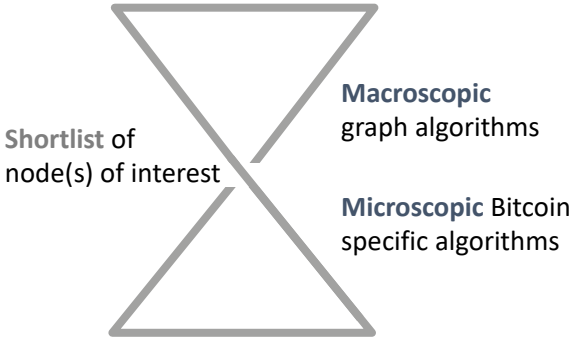


FIGURE 1.4: Thesis topics relationship: Loose coupling between the general purpose graph algorithms (useful for macroscopic analysis) presented in Chapters 3, 4 and 5 with the Bitcoin specific algorithms (useful for microscopic analysis) presented in Chapter 6.

which involve mixing. In addition to the general purpose graph algorithms, we design further algorithms that are specific to Bitcoin sub-network centered around node addresses of interest, to facilitate analysis, outliers detections by average path length, and path confluence based aggregation of Bitcoin wallet addresses that might belong to a single entity coordinating actions of several wallet addresses, as well as visualization tools - that enable us to carry out forensics of Bitcoin subnetworks. We showcase these with case studies of extortion events [32, 33] where perpetrators leverage the anonymous nature of Bitcoin to collect ransoms.

The figure 1.4 puts in context the high level and loose coupling relationship among the topics studied in this thesis. The general purpose algorithms are fundamentally related to each other in their roots in information theoretic principles. Several of them were also inspired by the nature of relationships that occur in the Bitcoin networks, which are induced by (non-atomic) flows of money. These algorithms however are designed by abstracting out the specifics of Bitcoin, and are thus general purpose. These general purpose algorithms designed in this thesis, other third party algorithms as well as any other means (e.g., some sort of side information) may be used separately or in conjunction, to identify Bitcoin subgraphs and nodes of interest. The node(s) of interest can then be investigated further applying Bitcoin specific algorithms and visualization based exploratory techniques designed and discussed near the end of the thesis.

1.2 Research Contributions and Thesis Organization

In this thesis, we make the following contributions.

1. We extend the notion of entropic centrality proposed in [18], which originally considered only the transfer of flow tracing loopless path based on random walks, and define entropic centrality based on a split-and-transfer flow model (similar to the right subfigure of Figure 1.3). We establish that the computation of split-and-transfer flow based entropic centrality for a graph with arbitrary weighted edges can be transformed to the computation of transfer based entropic centrality (which has a simpler mathematical model) over a graph with suitable edge weights. This work is presented in Chapter 3.

2. We revisit the above mentioned idea of entropic centrality based on transfer based flows (represented in the middle subfigure of Figure 1.3), and extend a variation in the literature [5] that simplifies the computation using a Markov model, by relaxing the constraint on the random walks and allowing cycles. Specifically, the original work [5] considered graphs with unweighted, undirected edges; and in our extension we generalize the model to consider all combination variants: un/weighted, un/directed graphs.

   Based on our observations of the probability distributions induced by the random walks, specifically that nodes at the boundary of any cluster often act as hubs and have high centrality, we design an algorithm to determine meaningful node centric local communities. The process is reapplied on the created clusters (instead of the nodes) to effectuate a bottom-up, scalable, hierarchical clustering. Our clustering approach inherits the flexibility of the underlying centrality model, and is applicable to all variations of graphs (un/directed and un/weighted graph), in contrast to most existing graph clustering algorithms that are designed and applicable to only one variant each [7, 34–42]. We present these ideas in Chapter 4.

3. We explore an alternate, Renyi entropy based graph clustering approach, extending an idea that has previously been applied in the context of image clustering [43]. We demonstrate two distinct manners in which to realize this in the context of graph clustering, one based on simulated annealing,

and the other following a classic agglomerative hierarchical approach. This is presented in Chapter 5.

4. One of our underlying motivation for exploring information flow based graph analysis and algorithms was to understand the circulation of money in the Bitcoin network. The above enumerated techniques aide macroscopic analysis, creating shortlist of networks, or identifying community structures. Thus, to close the loop, we propose Bitcoin specific analysis techniques to zoom in and carryout microscopic analysis, and better identify the community around a (set of) node(s) of interest, and demonstrate with case studies how these can be applied for Bitcoin forensics. Specifically, we extend the analysis from [44] to study the expected path lengths in directed graphs. This is used to identify outliers, specifically close-knit nodes. We then design a path confluence algorithm that provides a new way to aggregate suspicious addresses in Bitcoin address subnetworks and a mechanism for estimating the flow of money by traceback/forward to/from a set of nodes of interests. We also design and implement BiVA, a graph mining tool to aide visualization and analysis of information from the Bitcoin network. We describe this in Chapter 6.

An overview of related works is provided in the next Chapter. We wrap the thesis in Chapter 7 where we conclude by summarizing the strengths as well as shortcomings of the body of work presented in the thesis, and indicating some of our planned future work accordingly.

The study of the financial transaction network, specifically network induced by Bitcoin transaction [16], originally motivated our investigations of non-atomic flow-based networks. Accordingly, we design general purpose information flow based graph algorithms, specifically, entropic centrality [45], and entropy-based clustering [46]. While we showcase the applicability of our general purpose algorithms for the analysis of Bitcoin subnetworks, we design further algorithms [47] specific to Bitcoin networks, meant for detecting outliers by using average path length statistics for directed graphs and for identifying path confluences to aggregate Bitcoin wallet addresses, along with visualization tools [48] for exploratory data analysis.

Chapter 2

Literature Review

We will next explore two broad areas of related works (i) network (graph) analysis and (ii) Bitcoin forensics. Below, we present an extensive literature review for network analysis in more detail. Then, we provide related works on the aspects of Bitcoin Forensics. Further, closely relevant and related works are discussed in place, where individual aspects of this thesis is studied.

2.1 Network Analysis

In the wider literature, the term 'network analysis' is used in several distinct contexts, but in this thesis it is confined to a specific meaning, namely, analysis of relations through mathematical graphs. We denote a (directed) graph as $G = (V, E)$. A graph comprises set of vertices V; and a set of edges $E \subseteq V \times V$; furthermore, if the graph is weighted, then each edge $e \in E$ is associated with a numeric value $w \in \mathbb{R}$ indicating the weight.

Development of network analysis emerged out of several research traditions including a now famous experiment [49] from the 1960s of forwarding a letter through knowns contacts, starting from and finally destined to two persons chosen at random, establishing the notion of small-world in a formal manner. In 1970s, network analysis became a distinct research endeavor [50–52]. In 1994, social network analysis was used in [53] to help perceiving various phenomena in social and behavioural science including community elite decision making, group problem solving, world political and economic systems, etc. Over the years, many methods and analysis techniques have been developed (see e.g. [54] for a survey). A framework for graph analysis at different levels has also been developed to obtain more insights into

networks, authors of [55] provides a taxonomy of three groups that illustrates the level of analysis, which we paraphrased below.

Node-level analysis: emphasizes network analysis at the node level including (but not limited to) (1) ego-network analysis, which examines behavior of a particular individual across different settings of network. The term 'ego' refers to a focal node. Egos can be individual people, group, organizations, etc.; (2) centrality measures, which identify important nodes in the network (see Subsection 2.1.1 for more details); (3) PageRank analysis [56], which captures the likelihood that a given node is the destination when links (edges) are followed at random; (4) network bridges analysis, which investigates links between two separated group of nodes.

Group-level analysis: includes (1) cohesive subgroups analysis, which emphasizes on network portion consisting of high density of nodes; (2) cliques analysis, which identifies subgraphs in which every vertex is connected to every other vertex (or some approximation thereof); (3) clustering coefficient, which is a measure of density of cluster of nodes; (4) triadic analysis, which investigates network triads (a subgraph comprises a focal, an alter and a third node), in order to understand the structure of overall network; (5) structural holes analysis [57], which studies the positional importance of focal nodes in ego-network. A structural hole is a hole such that its two neighbours cannot connect directly; (6) brokerage analysis which identifies relations in networks as exchange channel (e.g., exchange information, service, disease and etc.). It is a measure of a focal node's potential to play as subversive role in triads comparing to other focal nodes; (7) transitivity analysis which predicts about the nodes' neighborhood by using property of edges to extend a relation between two nodes; (8) coreness, which is a measure of how strongly the nodes are connected within a network; (9) community detection, which finds clusters where nodes are densely connected to other nodes within the same cluster, and is also referred as clustering techniques (see Subsection 2.1.2 for more on clustering techniques in details)

Network-level analysis: focuses on network properties as a whole, in general. It consists of (1) density analysis, which examines network density defined as the number of links over maximum possible number of links between nodes in the network; (2) distances which are calculated by number of links in shortest path between any pair of nodes in the networks. Measuring average shortest path over

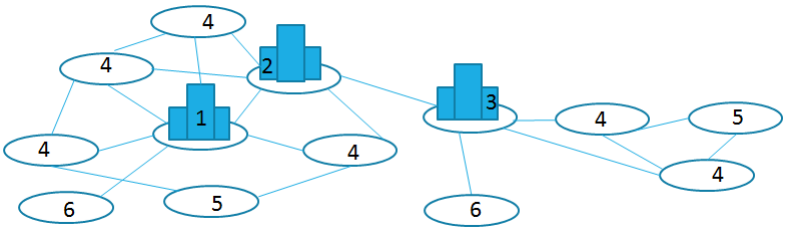


FIGURE 2.1: An example undirected graph along with the nodes' respective degree centrality ranking.

all pair of nodes is crucial for many algorithms specially graph clustering and outlier detection (see Subsection 2.1.3 for more details); (3) reciprocity, which often focuses on directed graph and analyses the ratio of links pointing in both direction to the total number of links in the graph; (4) homophily analysis, which identifies interaction pattern in nodes tending to be more often linked to other similar nodes.

In this thesis, we mainly model and study flow based networks, design algorithms (particularly for determining nodes centrality and graph clustering) leveraging on information flow, as well as study some specific network properties, for instance, average length of shortest paths. Accordingly, we would next concentrate on related works directly relevant to the subset of topics we explore in this thesis, and omit from our discussions most of the other network analysis topics listed above. We will wrap this section with a (non-exhaustive) list of applications of graph analysis to forensics and law enforcement.

2.1.1 Centrality Measures

Given a graph to be analyzed, vertex centrality is a notion meant to identify the most important vertices within this graph. What "most important" means is not universally defined, therefore, numerous notions of centrality have been proposed [58], e.g.: the *degree centrality* is just the degree (or in-degree/out-degree) of the vertex depending on whether the graph is directed, possibly normalized to get the fraction of vertices a given vertex is connected to. For instance, Figure 2.1 shows example degree centrality ranking of nodes for a small network in our toy example; the *closeness centrality* is the reciprocal of the sum of the shortest path distances from a given vertex to all others, typically normalized. In the above mentioned example graph, the node which has second highest degree centrality, has in fact the highest closeness centrality; the *betweenness centrality* is the sum of the fraction of all pairs of shortest paths that pass through it. The node which is ranked third

in terms of degree centrality happens to be the most central node in terms of the betweenness centrality in this example network.

Another way to determine centrality is to assign as centrality a (scaled) average of the centralities of the neighbours. This is the idea behind *eigenvector centrality* discussed by [59], which was already debated by [60], who later generalized it to *alpha centrality* ([61]). Alpha centrality introduces an additive exogenous term, which accounts for an influencing factor which does not depend on the network structure. Though *Katz centrality* ([62]) relies on the idea that importance is measured by weighted numbers of walks from the vertex in question to other vertices (where longer walks have less weights than short ones), it turns out that the alpha centrality and Katz centrality differ by a constant term. With these three centralities, a highly central vertex with many links tends to endorse all its neighbours which in turn become highly central. However one could argue that the inherited centrality should be diluted if the central vertex is too magnanimous in the sense that it has too many neighbours. This is solved by Page Rank centrality, which is based on the *PageRank* algorithm developed by [56]. [63] showed that a parametrized random walk model can capture the behaviour of a gamut of centrality measures, including degree centrality (walks of length one) and eigenvector based centrality models (considered as infinite walks), which contain the eigenvector and Katz centralities as particular cases. This parametrized model helps explain and interpret the high rank correlation observed among degree centrality and eigenvector based centralities.

Notwithstanding this high rank correlation among centrality measures, each measure captures the vertex importance subject to a certain interpretation of importance, which is a key rationale behind studying different centrality models in different contexts. Centrality from a flow view point has been investigated in [64], where different types of flows were categorized into parallel duplication (e.g., email virus alert), serial duplication (e.g., person-to-person gossip), or transfer process (e.g. package delivery), and the methods by which the traffic spreads were identified to be paths, geodesic, trails, and walks. Many studies [65–68] enhance classical centrality, namely closeness and betweenness, to take into account model of information spread, and to deal not only with network flow but also current-flow or traffic-flow. The closeness centrality gives the highest weight to vertices which are closer to all others. The betweenness centrality characterizes the number of times a vertex acts as a bridge along the shortest path between two other vertices. It

was introduced in social networks [69] to quantify the control of a human on the communication between other humans. The degree centrality can be interpreted in terms of immediate propagation of a flow from a given vertex. This is of interest if say we are considering the spread of an epidemic. [70, 71] extend network flow based betweenness centrality which assumed that the information spreads only over shortest paths, and relax this assumption to include information spread to all connections between node pairs. The model of flow with parallel or serial duplication was studied by using one of the classical model of information diffusion, specifically the independent cascade model (see [72] for survey), which captures the process of flow spreading by calculating information (influence) probability that a node gets affected from another node at any time step. Due to rising commercial value of social network influencers, many information diffusion studies [73–76] focus on influence maximization (top centrality) problem which aim to determine the set of nodes with the maximum information (or influence) spread. In contrast, one of the central tenets of our centrality model is flow volume conservation, even while the flow may be non-atomic.

The idea to look at flows with transfer process from an information theoretic view point was proposed in [18]. The larger the uncertainty in determining the destination of the flow, the more central the node of origin is deemed. Such a model sometimes assumes an implicit self-loop at each node in order to capture the possible termination of a flow at any node. This gives rise to the notion of *entropic centrality*, which is the focus of this thesis. In [5], the model of [18] was slightly modified so that the flow propagation can be modelled as a Markovian transfer process. Both [18] and [5] assume that the flow is atomic, that is, when it arrives at a graph vertex, it can either stay there, or can go to one of the neighbors, but cannot be split among the neighbors (including the node itself). Our own works on entropic centrality build upon these two works [5, 18], and in particular, we generalize the theory to capture arbitrary splits of the flow, which makes our model amenable to also capture weighted graphs; such that the original works [5, 18] can be seen as special cases of our generalized model.

Other recent works with slightly different interpretations of entropic centrality also exist in the literature. For instance, [77] evaluated entropic centrality to identifying vital nodes in a network by re-defining entropy centrality model based on an analysis of its neighbors' entropy and partitioning the graph into subgraphs. They then extend this work to support weighted networks [78]. In [79], a notion

of mapping entropy was proposed based on local knowledge which considers correlation between a node and its neighbors. An information flow model is naturally related to the process of spreading, and there are other centrality measures used to quantifying spreading in the network. The study in [80] used distance entropy defined as the information entropy of distances between nodes in the network, and the study in [81] determined the spreading capability of nodes based on their topological locations in the network. All these works which look at information spread are similar in spirit, and vary mainly in their interpretation and rendition.

Our specific works encapsulate two specific aspects often not addressed in most of the other works - namely, the possibility of arbitrary (and repeated) splits of the flow at each (intermediate) vertex, and conservation of the total volume of the flow at each stage of split. This model is particularly amenable and applicable to certain applications, this includes (among others) study of the flow of money or cryptocurrency such as Bitcoin - where, indeed, the money can be split arbitrarily at each juncture, but the total amount of money is conserved. Note that, in the context of cryptocurrencies, there is a notion of 'mining' which creates 'new' money. But that is something we can treat 'differently' in application of our graph analysis techniques to Bitcoin network analysis, by decoupling such source of money (creating a new flow, so to say) as extrinsic to the rest of the graph.

2.1.2 Clustering Techniques

While node centrality is one tool for graph analysis, which finds 'important' nodes according to a metric of interest, another crucial one is clustering, the task of grouping nodes by some 'similarity' criteria. Many general clustering algorithms [82–84] are studied in the context of graph clustering. For example, the basic idea of the classic k-mean clustering algorithm is to partition data into k-partitions and to move data into the cluster with nearest mean of data in the cluster. Graph based k-mean clustering algorithm is presented in [85]. The classic hierarchical clustering algorithm treats every data point as an individual cluster initially, then merges clusters with some strategy such as merge two closest clusters. Graph clustering based on hierarchical algorithm has been studied in [35]. Spectral clustering, which is another classic clustering algorithms, relies on the eigenvalue of the similarity matrix of data to cluster, and graph spectral clustering is studied in [35].

There are numerous clustering algorithms for undirected (weighted) graphs (see e.g. [34] for a survey of well-known techniques), but clustering algorithms

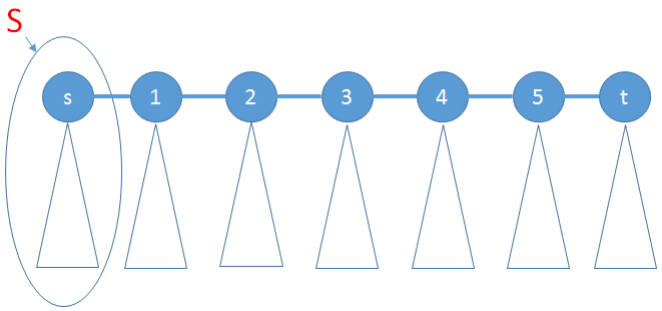


FIGURE 2.2: Consider a graph G= (V, E). Given node $s, t \in V$ and a path from s to t comprising subset of nodes, $\{s, 1, 2, 3, 4, 5, t\}$. When we focus on path s to t, a cut on edge $(s, 1) \in E$ would yield groups S and V-S based on maximum flow minimum cut. In this figure, a circle shows the subset of nodes S after cutting, and triangles represent subtrees (comprising of nodes) which are not part of the path from s to t. The figure has been reproduced from [1].

for directed (weighted) graphs are fewer (see e.g. [35] for a survey). Recently, authors of [86] summarized the main aspects of community detection problem and illustrated the strengths and weaknesses of popular methods in un/directed (weighted) network. As reported in [35], many such algorithms rely on creating an "undirected" representation of the directed graph considered, which may or not maintain directionality. Examples of notable exceptions where the clustering algorithm is designed for directed graphs are the information flow approaches of [36], where clusters are identified as subgraphs where the information flow is larger compared to outside node groups, [37], where an information flow-based cluster is a group of nodes where a random surfer is more likely to get trapped, rather than moving outside the group. We explore two clustering algorithms. One of them (discussed in Chapter 4) considers a similar flow confinement based approach for clustering, and is applicable to all classes (un/directed, un/weighted) graphs. Another is based on an entropic distance based approach (discussed in Chapter 5) which at the moment is applicable to undirected graphs. For this, we have deferred the generalization to directed graphs as future work.

There are also other interpretations of graph clustering based on information flow network. Prominent among these is a maximum flow minimum cut based approach (for example, [1]). The idea is illustrated with an example in Figure 2.2. This graph clustering technique can be applied to any kind of graph such as undirected graph, directed graph and weighted graph. However, it is best suited for graphs where a link represents reference such as citation graph or web. A variation of min-cut flow-based community detection in studies [7, 42] utilizes PageRank

[56] to capture in/out-flow information, and partition clusters by minimizing the average description length. Later, authors of [87] used cluster updating rules in [7] to extend label propagation mechanism [88] to update label of a node to the label of the majority of its neighbors, taking into account label updates which minimize the average description length. A variant of community detection using PageRank is [9, 89] where authors utilized partition graph by using a random walk to perform personalized PageRank diffusion to determine visiting probabilities from a stating node, and then choosing a cut that produces subset of nodes with conductance minimization as local cluster. Community detection using information flow simulation was studied for social network in [90]. The authors find community by treating influence nodes as point of information origin, simulating flows over the network and building clusters around influential nodes. From information theoretic point of view, the entropy based graph clustering was studied in [40] in order to detect clusters with high modularity and low entropy. While, authors of [91] extract cohesive clusters by adding node attributes and compressing information flow.

It is worth to mention clustering algorithms which are designed for undirected graphs and inspire us to explore community detection designed for flow network. These include [5] which used edge removal algorithm to cluster an undirected graph by recomputing total Markov entropy for every graph instance created by removal of each possible edge, to determine which edge to remove, so that the aggregate entropy decreases the most; [38] which studies multiple chained walker model to allow multiple walkers to explore the network, so that confinement probability of random walks is increased within a local community which contains the query (start of the walk) node; and [39] which was an extension of [38] to support multiple query nodes. A variant of community detection using Markov chain is [41] where a random walk starts from a link and transition probability of Markov chain is used as similarity between a link pair. Our flow confinement based clustering algorithm (discussed in Chapter 4) uses similar idea of exploiting the probability distribution gradient as explored in [38], but does so with single (instead of multiple) random walker, and instead augmenting this by choosing the starting (query) node in a manner informed by entropic centrality that is amenable to identifying local communities.

There are several other kinds of related Markov random processes, particularly label propagation based on random walks. Semi-supervised learning was applied

Approach	Parameters	Type of network
Community by label propagation with compression of flow [87]	Label propagation, and average description length	Undirected
Partitioning graph by personalized PageRank [9, 89]	Personalized PageRank vector, conductivity, and query nodes	Undirected
Entropy centrality based clustering [5]	Transition probability matrix for Markov process	Undirected
Clustering by multi-walker chain [38, 39]	Transition probability matrix for Markov process, and query nodes	Undirected, weighted
Entropy based community detection [40]	Modularity, and entropy of partition C	Undirected, weighted
Link clustering by random walk [41]	Transition probability matrix for Markov process, and link distance matrix	Undirected, weighted
Community by mapping of random walks [36]	Transition probability matrix for Markov process, and modularity	Directed, weighted
Modularity maximization [37]	Modularity, eigenvector, and eigenvalue	Directed, weighted
Clustering by map equation [7, 36, 42]	PageRank matrix, Modularity	Directed, weighted
Entropy based partitioning by compressing information flow [91]	Modularity, entropy of partition C, and PageRank matrix	All
Clustering by information flow simulation [90]	Nodes, and percentage k of nodes	All
Clustering by minimum cut tree [1]	Parameter α	All

TABLE 2.1: List of notable graph clustering algorithms; type of network: 'All' refers to (un/directed and un/weighted) graph.

[92] to classify unlabelled data from labelled data by combining data class distribution and random walk. Later, the same authors extend their work [92] by using Gaussian random field [93] model through random walk in [94]. In [95], supervised learning was improved by applying global consistency and allowing every node's label to be continuously influenced by those of its neighbors. Moreover, authors of [96] modified label propagation by partially absorbing random walkers, and then starting a random walk from 'query nodes' with low conductivity. We explore a clustering algorithm which utilizes a flow confinement based approach for clustering through Markov random process with starting nodes based on entropy centrality

ranking nodes. These related works based on random walks are not pertinent to the problems of interest in this thesis. Even though techniques utilize partially absorbing random walkers, the underlying mechanism of label propagation is consistent with a scenario where a node's label gets influenced by its neighbors. In contrast, we are trying to identify sets of nodes where flows tend to get trapped. and hence we do not discuss them any further.

Table 2.1 summarizes the prominent graph clustering algorithms mentioned above in terms of their applicability to different types of graphs.

Based on the typography of community evaluation mechanisms into four classes [97], our clustering based on flow confinement based approach can be considered as conductance related, and fits in the broader group of techniques which evaluate communities based on combining internal and external connectivity. Our other clustering approach, based on Renyi entropy emphasizes distance between clusters, belong to the category of mechanisms which evaluate clusters based on external connectivity class. One may arguably also use evaluation functions described above, (namely, conductance [98], normalized cut [98], maximum/average/flake out degree fraction [99] for internal and external connectivity class; cut ratio [100], expansion measure [101] for external connectivity class) to evaluate our clustering algorithms. In order to capture our specific community interpretation, we use another well-established evaluation measure, pairwise F-score [102] (for data sets with known ground truth), while the between-cluster entropy function [43] is further used to evaluate the entropic distance based clustering technique.

2.1.3 Average Path Length

An important and widely studied property of a graph is its average path length. In general, an average shortest path in random network has many application including understanding brain network [103], analyzing optical network [104], determining the cost of routing in overlay networks [105], etc. The average path length is a proxy for the efficiency measure of information flow over a network, and is defined as the average number of edges along the shortest paths for all possible vertex pairs. In modelling complex networks, a random undirected network has been studied to classify random network topology in terms of average path lengths [106]. The expected behavior of (power-law) graphs in terms of average path length has been studied in [44]. A minimal path length distribution was established for

random networks in [107, 108] using Weibull distribution [108] model by application of extreme value theory. Statistically, the average shortest path is equivalent to an average hop count of the shortest path for all node pairs. Based on this observation, the authors of [109] develop a general formula of average hop counts in finite-sized tree-like components in random network with Poisson [110] degree distribution.

Due to the frequent occurrence of scale-free networks in real world, the average path length of scale free networks has been studied in [111], which estimates the average path for Barabási-Albert (preferential attachment) model based scale-free networks. Based on the power-law distribution property, a formula for the average path length was estimated in [112]. There are also empirical works, and there an interesting challenge is to improve the (distributed) computation. In [113], average out-degrees of neighbouring nodes is used as variable in dynamic programming to reduce the computation time and memory space required for computing average shortest path of a large scale free network. The algorithm for calculating the average path length was optimized in [114] by using reachable matrix in complex network such as scale-free network and exponential network.

In the context of directed graphs, there are relatively fewer studies on path characteristics. Average path length was studied in [115–117] to describe the structure of directed graph induced by Twitter[1] 'follow' network. In [118], authors estimate average path length in social networks by using random walk to sample a portion of the network in the context of partial/incomplete data sets possibly due to privacy and restricted access.

We are interested in a random directed network with specific (power-law) in-degree/out-degree distributions. For example, a network of wallet addresses induced by flow of Bitcoin consists of directed edges, and has been shown to have power-law in/out-degree distributions [119]. However, prior study [44] in this specific context was confined to undirected graphs. In this thesis, we extended the analysis from [44] to directed graphs to better fit our immediate need (of analyzing Bitcoin wallet address network), but the result is of-course of general value, and adds to the existing network science literature.

There are many other kind of related studies, particularly on optimization of average path length by adding more links [120–123]. These are not pertinent to the problems of interest in this thesis, and hence we do not discuss them any further.

[1] https://twitter.com/

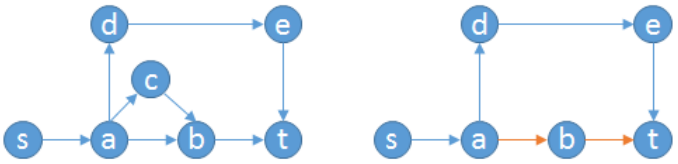


FIGURE 2.3: Superimposed paths on the left, and core paths on the right. Two distinct paths are indicated with differently coloured edges. The Figure has been reproduced from [2].

2.1.4 Application of Graph Analysis to Forensics

Graph analysis has been applied to a wide range of applications. This includes its extensive use in social sciences [5, 40, 53, 57, 67, 115–117, 124], to biology, for instance for functional brain imaging [125], or to perceive functional protein interaction and cancer data [126], to generate business intelligence, such as study of tourism destinations [127], ecological studies such as assessment of forest landscape [128], city planning and understanding urban street [129], and for criminal intelligence [130] to name just a few.

In this thesis, we are interested in network analysis to investigate Bitcoin network in general, and in particular for carrying out studies that aide forensics. Hence, we next review more on different kind applications of graph analysis to forensics and criminal investigations. A more extensive survey on graphical models specifically for forensics analysis can be found in [131].

Graph theory has been used in pattern detection among heroin cutting agents [132] to highlight possible relationship between co-occurrences of cutting agents and location of seizures. In [133], a New York city cocaine dealing dataset [6] is analysed using a variant of degree centrality, which emphasizes on density of links between individuals, to find a focal point or a central person. This is one of the datasets we use (in Chapter 4), in which we model the network as a directed weighted graph and apply our entropic centrality to capture the effect of transitivity in determining the importance of nodes.

In the context of cyber-attacks, when vulnerable nodes in a network get attacked through multiple pathways, graph theoretic analysis can be used to identify a core attack graph [2] and to focus only on that for further study or mitigation. A core attack network/graph is a compact representation of main attack routes and is formally a union of core paths from source node s to destination node t. For example, in Figure 2.3, on the right, node s has two core paths which are (s, a, d, e, t) and (s, a, b, t) to attack target node t, hence a core attack graph

comprises $\{s, a, b, d, e, t\}$. Breadth first search algorithm was adapted for the core attack graph investigation in order to summarize different levels of information.

In [134], application of graph models to analyse information obtained from the image of a virtual machine suspected to be infected is demonstrated, using a case study obtained from the Honeynet Project [135]. Specifically, a PageRank [56] analysis on the graph which comprised network connections and processes as vertices, and network connections and forking to other processes as directed edges was used to identify key pieces of evidence, particularly which network connection was the epicentre of the incident. Furthermore, shortest path was used to determine the actual occurrence of events, specifically that an email was received, an attachment was opened, a malicious file was executed which then accessed the company network.

Similarly, in [136], the PageRank algorithm is customized to quantitatively rank potential important attackers in evidence graph comprising host level network entities as vertices, such as a company network connection on which attackers perform activities such as data tampering, back door installation, account creation, etc. The authors of [136] also develop recursive spectral graph clustering algorithm to determine multi-stage attack from evidence graph.

2.2 Bitcoin Forensics

Above, we discussed about graph algorithms in general, as well as how they are applied in the specific context of forensic studies and related law enforcement contexts.

We will next discuss the specific (and sometimes unique) characteristics of graphs induced by Bitcoin transactions, and accordingly discuss specific techniques that have been applied to understand the Bitcoin ecosystem or to carry out forensics. This includes general analysis, visualization tools, and (wallet) address aggregation and de-anonymization efforts.

Before we delve into the details of such related works, we first summarize some Bitcoin specific terminology so facilitate our presentation.

1	Inputs: ∅ Outputs: 25.0 → addr_A
2	Inputs: 1[0] Outputs: 17.0 → addr_B, 8.0 → addr_A
3	Inputs: 2[0] Outputs: 8.0 → addr_C, 9.0 → addr_B
4	Inputs: 2[1] Outputs: 4.0 → addr_D, 2.0 → addr_B, 2.0 → addr_A
5	Inputs: 3[1], 4[1] Outputs: 11.0 → addr_B
6	Inputs: 3[0], 5[0] Outputs: 3.0 → addr_D, 7.0 → addr_C, 9.0 → addr_B

TABLE 2.2: Transaction examples.

2.2.1 Bitcoin Terminology

The Bitcoin network [16] is a peer-to-peer payment network, where users send and receive a cryptocurrency called Bitcoin, by broadcasting digitally signed messages to the network. Payments are grouped by transactions (see Table 2.2 for a hypothetical toy example), and the transactions are recorded into a public, massively replicated database called the blockchain.

The content of an actual Bitcoin transaction comprises metadata, inputs and outputs [137]. The metadata includes the hash of the entire transaction, which is a unique ID for the transaction (in Table 2.2 the transactions are labelled from 1 to 6 instead). Every input specifies a previous transaction, identified by its hash, and the index of the previous transaction's output that is being spent. For example, the inputs of transaction 5 are 3[1] and 4[1], referring to the second outputs (outputs are labelled from 0) of transactions 3 and 4 respectively. Since the same address is used, we see this transaction is a fund consolidation. Every output contains the value (amount) of the transaction, and a mechanism to identify the recipient address. We next describe some basic terminologies in a layman's language rather than formally (since, that suffices for the purposes of the work described later in the paper), and without going into the technicalities of the cryptographic primitives and incentive mechanisms based on which Bitcoin works.

A Bitcoin address is an identifier which consists of total 26-35 letters and numerals. A Bitcoin payment is assigned to a Bitcoin (wallet) address, which acts as a pseudonym for a user. Essentially, the Bitcoin address is derived from the public key of an asymmetric key cryptographic key pair, and the user needs

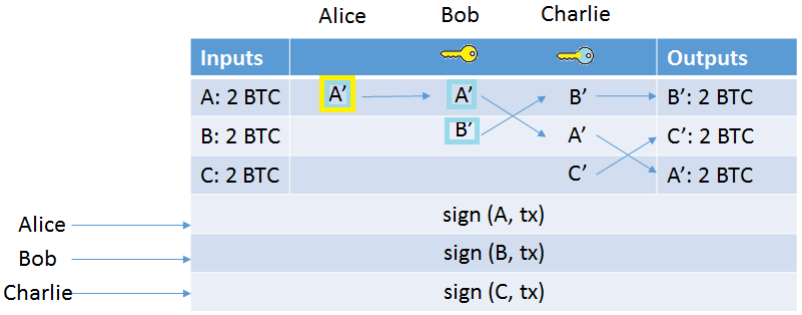


FIGURE 2.4: Example of CoinShuffle system: First, there are three input addresses, then they shuffle their output address obliviously by encrypting ciphertexts with their respective encryption key (colored boxes show encrypted text matching corresponding colored keys), and users next check if all their output addresses receive Bitcoins. Figure reproduced from [3].

to know the corresponding private key in order to be able to control the specific wallet address, particularly to spend the money.

A transaction is a handover of Bitcoin value after broadcasting to the network and being collected to blocks. A transaction input must be (some) previous unspent transaction output. In a transaction, transaction outputs' Bitcoin values are transferred from the cumulation of all the inputs' Bitcoin values. Lastly, all legitimate transactions are recorded in the blockchain, to keep track of which wallet address has been assigned which transaction output, and also whether said output has already been spent or is yet to be spent.

A block is a file in which transaction data is permanently recorded. A new block is connected linearly to the previous block which is the last block in the longest chain (known as blockchain). Also, a new block is filled with new transactions that are continuously being processed by miners. A miner is a node in the peer-to-peer network who needs to solve certain computation intensive puzzles in order to prove the legitimacy of the block, which then is verified by other fellow miners, for a block to be accepted for inclusion in the blockchain.

A blockchain can be viewed as a persistent append-only database that record the Bitcoin transactions in the form of blocks. In the Bitcoin system, all participating machines have a copy of this database. A whole replica of Bitcoin blocks contains every transaction from the beginning of the Bitcoin system till present.

Mixing services are trusted services that mix one's fund with other people's fund. The purpose of mixing is to obfuscate the link between the inputs and outputs of a transaction.

Trustless mixing service is a trustless (typically, decentralized) method

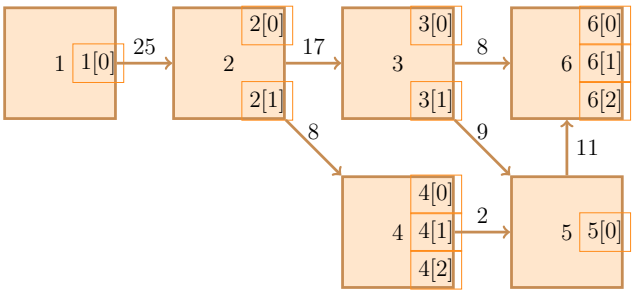
for combining multiple Bitcoin payments or swapping funds from multiple Bitcoin users in a single transaction. Coinjoin [138]/CoinShuffle [3] is a popular approach to do so. Figure 2.4 shows overview of CoinShuffle.

An organization providing mixing service would have access to information to link inputs and outputs of a transaction. This information can also be accessed by third parties, for example by a subpoena or because of a data breach. These are reasons which have prompted the embrace of trustless/decentralized mixing. In our work, we have not considered the possibility of using any side information (including the one described here), and have pursued a line of investigation which only uses (some of) the information that is encapsulated in the blockchain. Naturally, use of such side-channel information can complement and help achieve better results. Of-course, within the context of a single thesis, it was essential to limit the scope sufficiently, so that in-depth study in one aspect could be carried out. The work on Bitcoin forensics presented in this thesis, and its strengths and weaknesses need to be viewed accordingly.

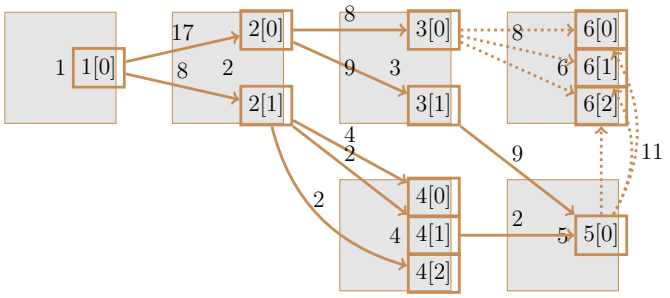
2.2.2 Visualization of Bitcoin Network Data

Bitcoin network can be modelled as a heterogeneous graph, where nodes comprise transactions, identified by their unique hash, and Bitcoin addresses. Typically a user may own many Bitcoin addresses. From each input of a given transaction T, a directed edge is created, linking an output of some previous transaction T' to T. This same edge will also appear as an output of T', from which the transaction amount can be read.

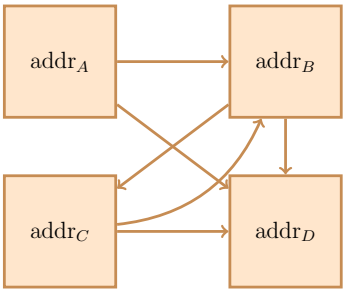
Different networks can be extracted (some are shown in Figure 2.5) from the Bitcoin network: (1) A **transaction only network**, as used e.g. in [139], represents the flow of Bitcoins between transactions over time. Each vertex represents a transaction, and each directed edge between a source vertex to a target vertex connects an output transaction of the source, to the target where it is used as input, as illustrated on Figure 2.5a: we have 6 transactions from Table 2.2, thus 6 nodes. The first transaction has no input, but one output, transaction 2 has one input and two outputs, each of them are inputs to respectively transactions 3 and 4. (2) An **input-output network** can be considered, where nodes are addresses, and there is a directed edge when an output serves as input to a transaction. This is possible without ambiguity if a transaction has one input with possibly several outputs (as in transaction 3 in our example), or possibly several inputs and one



(A) The transaction only network.



(B) The (transaction) input-output network.



(C) The (wallet) address network.

FIGURE 2.5: Different network views to capture Bitcoin transactions from Table 2.2. Dotted lines indicate ambiguity, the amount indicated then refers to the sum of the dotted lines.

output (as in transaction 5 in our example), but not when there are multiple input multiple output transactions (as in transaction 6, see Figure 2.5b), since, then we only know the inputs and outputs of the transaction node, but not how they relate to each other. When there are two outputs, oftentimes one of the outputs is for collecting change, while the other is the intended payee of the transaction (though it is still not clear which is which, and someone simply consolidating funds may also create two outputs as a ruse). This observation is the idea behind mixing, which obfuscates individual pairings in multi-input multi-output transactions. (3) A **(Bitcoin wallet) address network** could be created following e.g. [119], where each node represents a Bitcoin (wallet) address, and there is a directed link between two nodes if there was at least one transaction between the corresponding

addresses. Naturally, this may lead to the creation of 'phantom' links in multi-input multi-output transactions, because, even if there is no actual flow of Bitcoin among a specific pair of addresses, a link will be created. Furthermore, wallet addresses are sometimes aggregated via some heuristic to create a user network, as in e.g. [139].

The overall size of data encapsulating the history of Bitcoin transactions is enormous. Bitcoin blockchain held approximately 185 gigabytes of data, and comprised over 345 million Bitcoin transactions by the end of September 2018. A straightforward visualization of the whole graph is not useful for most practical purposes. In BitIodine [25], Gephi [10] is used to visualize Bitcoin transaction graphs and to aggregate addresses which have high possibility of belonging to same user based on certain heuristics, such as grouping multiple inputs of a transaction and 'change' address guessing [25]. BlockChainVis [140] is another tool used for visualization of Bitcoin flows, and has features to filter out information (for example, transactions of certain amounts), so that a simplified Bitcoin network can be visualized focusing on subset of information that the user may be interested in. In this thesis, we sometime use Gephi to visualize (Bitcoin wallet) address network and Bitcoin transaction input-output network.

2.2.3 Graph Analysis of the Bitcoin Network

Network analysis has been applied to study the Bitcoin network for various purposes.

(1) In [141] the degree of decentralization of the Bitcoin system is studied, given that few entities control most of the computing power (needed for mining and confirming blocks) in the underlying Bitcoin infrastructure network.

(2) The propagation delay in the underlying peer-to-peer communication network, which might cause inconsistencies during finding solution to a proof-of-work [142] and thus lead to forking, was studied in [143].

(3) Properties of the Bitcoin network induced by transactions has been studied in many works. [119, 144] establish that the Bitcoin network has power-law degree distribution and wealth distribution, and also explore degree correlations and clustering. In [26] other network properties such as centrality, clustering coefficient, degree distribution, and shortest path length are studied.

(4) Whether and how robust (or not) the Bitcoin network is in terms of user anonymity has been studied in numerous works, for example, [139, 145–147] (also see [148] for a survey, and in Subsection 2.2.4 next, we discuss some more details).

(5) There are many criminal event specific studies of the Bitcoin network [28, 149–152]. For example, ransom payments from the CryptoLocker ransomware incident[2] has been investigated in [28]. The analysis emphasized on the distribution of timestamps and cluster of addresses which were involve in various suspected transactions (payments of approximately 2฿, 1฿, 0.6฿, 0.5฿, and 0.3฿) during the period CryptoLocker was active. In [149], Bitcoin network related to the WannaCry incident [33] has been studied to trace addresses tied with the extortionists by extending Louvain clustering [8], which treats each node initially as an individual community and then merges two communities such that it results in maximum increment of modularity. Authors of [149] also add their own Bitcoin transactions and then apply their improved Louvain clustering on the network comprising of Bitcoin address which they control, and obtain result with perfect accuracy for tracing the addresses controlled by themselves. Another incident from June 2011[3] was analyzed in [150] using improved K-mean clustering [85] to identify rogue users which resulted in two clusters with one of them containing three suspect address. Authors of [150] also verified their analysis techniques with synthetic data and claimed a 76.5 percent accuracy.

To put things context, in this thesis, we curate data [22] for another extortion incident, which targeted victims of data breach from the Ashley-Madison website, and use it as a case study to demonstrate the efficacy of some of the other aspects of this thesis. Namely, our analysis for this case study uses not only existing graph analysis techniques and known Bitcoin analysis techniques, but also new graph algorithms, some general purpose, and some custom designed for the specific purpose of analysing Bitcoin networks. We also designed and developed a visual analytics tool to make Bitcoin forensics accessible, unencumbered by the knowledge of Bitcoin technicalities and graph analysis.

[2]https://www.ibtimes.com/cryptolocker-virus-new-malware-holds-computers-ransom-demands-300-within-100-hours-threatens-encrypt, Accessed on 09 January 2019.

[3]The dataset can be obtained from http://anonymity-in-bitcoin.blogspot.com/2011/09/. Accessed on 09 January 2019.

2.2.4 Address Aggregation Mechanism in Bitcoin Network

Because of their anonymous nature (wallet *addresses* are used for transactions, but the address owner is hidden, and a single person may control arbitrarily many addresses), cryptocurrencies, and Bitcoin in particular, have become a de facto medium for illegal activities such as online black markets (for selling e.g. drugs or stolen credit card information), carrying out extortion, receiving and laundering money. Studying the latter in terms of detecting Bitcoin flow involving suspicious wallet addresses and in turn singling out new addresses potentially controlled by the (same group of) perpetrators is thus an interesting topic of research.

Analysis and forensics of the Bitcoin network, especially in the early years, mostly relied on the idea of clubbing co-paying addresses together [16, 25–29], and guessing the address used to collect the change in a transaction (e.g.,[30, 31]). More precisely, when inputs from multiple wallet addresses are combined together as inputs to a single Bitcoin transaction, and there are at most two outputs for such a transaction (one for the intended payee, one to collect the change back), then it was assumed that all the input addresses of the transaction belong to a single entity. Likewise, when there are two outputs, the output with the smaller amount is considered as a 'shadow' address to collect 'change', while the output with the larger amount is considered to belong to a third party to whom money is being paid [30]. In [31], it was assumed that in general, the change value is smaller than any of the inputs. If the change was larger than any input, then this behaviour was considered as suboptimal, since, in that case, the transaction could have been carried out with fewer inputs. However, someone deliberately trying to blindside such a heuristic could have two addresses, and create two outputs, one very small, and one large, for each of the addresses, and the heuristic would only club the address receiving the smaller output with the input addresses, but will ignore the other output address. In fact, we witness this particular evasive behaviour in the dataset that we later study (in Section 6.5.2.4, Chapter 6), where we propose a path confluence based address aggregation mechanism which is robust against such an evasion, since it follows all the flows of money.

Furthermore, the heuristic of clubbing co-paying addresses extends to the case where some of the addresses are reused across multiple transactions, since then multiple such sets of aggregated addresses with non-empty intersection can be coalesced together. The proliferation of mixing (a process where multiple users come together to create a multi-input multi-output transactions) complicates forensics,

since it obfuscates the linkage between a particular input of a transaction to a specific output, making it difficult to trace the flow of money. In analyzing the efficacy of the clubbing co-paying addresses heuristic, [27] notes that it is surprisingly effective (while the authors do not discuss this further, we believe that this was so because a very small fraction of transactions representing mixing were present in the data set they studied), but that false positives are indeed caused by transactions which involve mixing. In [153], clustering based on transaction pattern analysis was applied on addresses in order to aggregate addresses. Transaction patterns include Peel transaction (involves only two outputs), Sweep transaction (involves only one output), Distribution transaction (involves more than three outputs), Relay transaction (only involves one input and one output), self-spending transaction (input appears as output), and peeling chain transaction (change address is used as input).

In [154], a de-mixing algorithm was proposed by matching amount of Bitcoin from output addresses to amount of Bitcoin from input address deduced from transaction fees over the time. [145] proposes metrics to quantify the use of well-known anonymizing techniques such as CoinSwap [155], Coinjoin [138] and stealth address mechanism [156]. The studies have found that CoinSwap comprises a large set of node relationships which is potentially vulnerable to statistical analysis, because CoinSwap performs mixing using many normal 2-of-2 multi-signature outputs.

2.2.5 De-anonymization in Bitcoin Network

To wrap up this chapter, even though it is beyond the scope of this thesis, we discuss the issue of de-anonymization beyond address aggregation. In [139], a partial user directory mapping Bitcoin (wallet) users was built with side information. Many organizations and services, that provide mixing services and laundering services, as well as legitimate Bitcoin exchanges, have ability (or even legal requirement) to associate information regarding their users. For example, Bitcoin Faucet[4] provides Bitcoin (wallet) address associated with IP addresses. For de-anonymizing Bitcoin (wallet) address with IP addresses, [146] developed heuristic to capture ownership relationship between each Bitcoin address and each IP address by leveraging anomalous relaying behaviour. The studies of [157, 158] geo-locate users based on IP addresses associated with relay nodes of the transaction.

[4]https://freebitco.in

Even though there is currently no absolute mechanism for de-anonymization of Bitcoin users (see [139, 145] for surveys on both privacy enhancement and deanonymization techniques used in Bitcoin/cryptocurrencies), the use of side (*off-chain*) information [139, 146, 157, 159] to identify real world identities has proven to be effective in certain occasions, and could be used in conjunction with the earlier discussed address aggregation heuristics. For instance, side information regarding a wallet address (associated with a set of wallet addresses allegedly involved in laundering money stolen from the now defunct Bitcoin exchange Mt. Gox) led to the identification (and eventual arrest) of Alexander Vinnik[5].

[5]https://blog.wizsec.jp/2017/07/breaking-open-mtgox-1.html

Chapter 3

A Split-and-Transfer Flow Based Entropic Centrality Measure

As noted in the literature review (Section 2.1.1), there are many centrality measures for identifying important nodes within a graph. The notion of importance would have different interpretations, depending on the type of network being analyzed, and the nature of property being explored.

Centrality using the notion of a flow circulating in the network has been studied in [18], where, higher the uncertainty in terms of where the flow may terminate, i.e., higher the spread of a flow starting at a given node, the higher the centrality of that starting node is deemed to be. It was shown, for instance that betweenness is best suited for geodesics and transfer, while eigenvector based centralities should be used for walks and parallel duplication (as elaborated in Section 2.1.1). Indeed, betweenness [70, 71] is based on shortest paths, suggesting a target to be reached as fast as possible, and thus fitting transfer. Using Katz's intuition [62], eigenvector based centralities [61] count possible unconstrained walks, and they are consistent with a scenario where every vertex influences all of its neighbours simultaneously, which is consistent with parallel duplication. Likewise, influence maximization [72] relies on a diffusion model to select set of nodes which satisfies diminishing marginal gain of the influence spread in the process of parallel duplication.

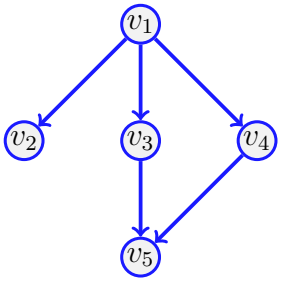
The original work [18] however considers the flow to be atomic, i.e., given a flow, at each node, the whole flow might terminate, or continue as a whole to any one of its neighbors. We consider a flow model where, at each step, the incoming flow can disperse to any arbitrary subset of neighbors, while conserving the overall

volume of the flow. This scenario could typically be motivated by financial transactions, which are transferred, not duplicated. However when transferred, the flow of money is not indivisible. Based on Borgatti's typology [64], a measure of centrality for transfer should be based on paths rather than eigenvectors. This is indeed the approach that we explore. The network induced by flow of money in general, and Bitcoin transactions, have such behaviour, and this is the principal motivation for us to consider this particular model. This specific motivation notwithstanding, we believe that this generalized model of entropic centrality which considers arbitrary splits-and-transfer of flow, can have many other applications, and thus this work is a valuable addition to the literature of graph centrality measures. For instance, entropic centrality can provide a unique perspective not captured by other existing centrality measures - particularly in identifying vertices with relatively low out-degrees which may nevertheless be connected to hub vertices, and thus having high spread in the network.

A priori, it is unclear how or whether the entropic centrality model designed for an atomic flow based on transfer forming a path [18] captures non-atomic flows. Therefore, we generalize and explore an entropic centrality considering split-and-transfer. We show how to relate this generalized model to an equivalent transfer based entropic centrality model for the ease of computation. We also demonstrate that in the special case where the flow split events always choose all possible subsets equally likely, and within the subsets, the flow is split uniformly, the split-and-transfer flow based entropic centrality and the transfer based entropic centrality are identical. We carry out three case studies (an airport network, a cross-shareholding network and a Bitcoin transactions subnetwork) to illustrate the interpretation and insights linked to the generalized notion of entropic centrality. Comparisons with other standard centralities (alpha, Katz, betweenness and PageRank) are given, showing that the entropic centrality captures different features.

3.1 System Model

Consider the scenario where we have entities involved in say selling or distributing a particular item. This can be represented by a graph whose set V of vertices contains the entities, and there is a directed edge between a pair of nodes (u, v) if u sends some item(s) to v (one could also put an undirected edge if what matters is that they are in business together, irrespectively of which of the two



$V = \{v_1, v_2, v_3, v_4, v_5\}$
v_1 sells to $\{v_2, v_3\}$ 1/3 of the time, and to $\{v_3, v_4\}$ otherwise.
Also v_2 gets 2/3 of the goods reaching $\{v_2, v_3\}$, while v_3 gets remaining 1/3. Similarly v_4 gets 3/4 of the goods for $\{v_3, v_4\}$, while in this case the remaining 1/4 goes to v_3. Then, each of v_3, v_4 keep half of what they obtain for themselves, and forward the other half to v_5.

FIGURE 3.1: A motivating example: suppose we have a scenario with the given information. We would like to build a graph whose node centrality captures these nuances.

is buying or selling). Also, weights could be given to the edges if one wishes to encapsulate, say the amount of goods being transferred from one node to another. The graph furthermore may have self-loops, that is directed edges from a node v to itself to capture the possibility that an item at v stays at v.

Consider the graph from Figure 3.1 depicting a seller v_1 whose direct customers are v_2, v_3, v_4. Say we further know that when v_1 distributes a new batch of items, he does so to either customers $\{v_2, v_3\}$ or $\{v_3, v_4\}$, and in fact, the pair $\{v_3, v_4\}$ is preferred (they receive 2/3 of the batches, versus 1/3 for the group $\{v_2, v_3\}$). Furthermore, v_2 receives a higher volume than v_3 (say 2/3 of the batch goes to v_2), while v_4 takes 3/4 of the batch shared with v_3. Once v_3, v_4 obtain the items, they typically keep half for themselves (self loops to represent such termination of parts of the flow are not depicted in the figure), and distribute the other half to v_5. We are interested in a model of node centrality with the flexibility to capture these nuances.

Our starting point is the notion of entropic centrality as proposed in [18]. In [18], a (directed) graph is built whose edges are unweighted, representing that two entities are related to each other. To define the centrality of $u \in V$, the probability $p_{u,v}$ that a random walk constrained such that it does not revisit any vertex (thus, only forming paths), starting at u terminates at v is computed. To model the stoppage of flow/walk at any node, a edge (self-loop) to itself is added on every vertex. The process of computing $p_{u,v}$ is thus to consider a constrained random walk to start at node u, and at every node w encountered in the path, to choose an outgoing edge uniformly at random among the edges leading to unvisited nodes (or choosing the self-loop to terminate the walk). Then the entropic centrality $C_H(u)$

of u is defined to be

$$C_H(u) = -\sum_{v \in V} p_{u,v} \log_2 p_{u,v}. \tag{3.1}$$

In this chapter we revisit and generalize the original concept of entropic centrality to make it more flexible. To do so, we first interpret the "transfer" centrality proposed in [18] as having (1) an underlying graph that models the scenario to be studied, where every edge has a probability which is that of being chosen uniformly at random among the other outgoing edges of a given node, and (2) an indivisible flow which starts at a node u, and follows some path where the probability to choose an edge at every node in this path is given by the probability attached to this edge, taking into account unvisited neighbours, to reach node v (as in the *transfer* model). Note that since the flow is indivisible, the self-loop in this model represents the probability for this flow to stop at a given node.

In our generalization, we will similarly assume that we have (1) an underlying graph, only now the probability attached to each edge depends on the scenario considered and could be arbitrary, (2) the flow used to measure centrality can split among neighbours, by specifying which subsets it goes to with which probability, at every node it encounters (as per a *flow* in the traditional network analysis sense, flow conservation applies, meaning that the amount of flow that goes out of u is the same amount of flow that reaches all of its neighbours). Again, a self-loop in this model is an artefact introduced to capture the effect of the flow on nodes, even if none of the flow actually remains in the node [5] (a zero probability would otherwise render zero contribution to the entropic centrality calculation). For instance, in a road network, vehicles do not just stop at road intersections, which may nevertheless be of interest. While the underlying phenomenon may have self-loops, they may or not be directly used to determine the self-loops needed for the mathematical model. This should be determined based on the scenario being modelled.

The above motivates the notion of a *split-and-transfer* entropic centrality. Since propagation of flow is an indicator of spread over the network, we will also consider a scaled version of entropic centrality, where a multiplicative factor is introduced to incorporate additional information, which may suggest an a priori difference of importance among the nodes, for instance, if the data suggests that some nodes handle volume of goods much larger than other nodes.

Let us illustrate these generalizations on the example of Figure 3.1, considering the centrality of v_1. We thus have a divisible flow starting at v_1 and propagating among its neighbours. We allow the flow of items to split among different paths

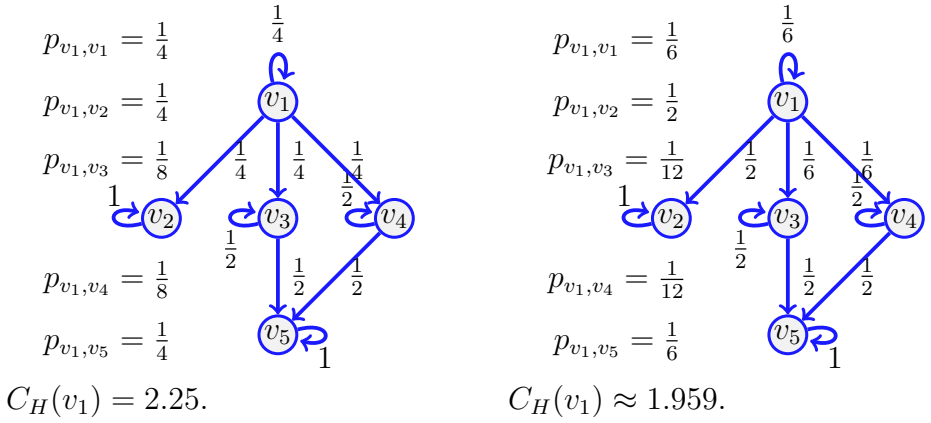


FIGURE 3.2: The transfer entropic centrality $C_H(v_1)$ of v_1 is computed using (3.1) for a uniform distribution on the left (meaning that the choice of an edge at a given vertex is chosen uniformly at random among choices of unvisited neighbours), and for some non-uniform distribution on the right.

instead of following one. This is done by choosing, at every node, a subset of the outgoing edges. For v_1, there are 15 such subsets: four containing one node $\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}$ (these correspond to the case of no splitting), six with two nodes $\{v_1, v_2\}$, $\{v_1, v_3\}$, $\{v_1, v_4\}$, $\{v_2, v_3\}$, $\{v_2, v_4\}$, $\{v_3, v_4\}$, four with three nodes $\{v_1, v_2, v_3\}$, $\{v_1, v_2, v_4\}$, $\{v_1, v_3, v_4\}$, $\{v_2, v_3, v_4\}$ and with all the nodes we have $\{v_1, v_2, v_3, v_4\}$ (as done in [45]). We then choose the probability $q(x)$ with which any subset x appears (in contrast to [45] where it was chosen to be uniformly at random). In our example, this corresponds to saying that the pair $\{v_3, v_4\}$ receives the batch 2/3 of the time, versus 1/3 for $\{v_2, v_3\}$. Finally, within a given subset, this is further refined to indicate which weight (or quantity of goods) goes to which nodes conditioned on the fact that the subset is chosen (e.g., 2/3 of the batch goes to v_2 and v_3 receives the other 1/3, while v_4 takes 3/4 of the batch shared with v_3).

3.2 The Notion of Split-and-Transfer Entropic Centrality

We first summarize the idea of flow transfer entropic centrality before we generalize and study the split-and transfer entropic centrality.

3.2.1 The Transfer Entropic Centrality

Consider the network shown on the left of Figure 3.2, and assume that at any node, the probability of an indivisible flow going to another node is uniform at random (including the option to remain at the current node). For a flow starting at v_1, there is then a probability $\frac{1}{4}$ to go to v_4, and a probability $\frac{1}{2}$ to continue to v_5, so the probability to go from v_1 to v_5 following the path (v_1, v_4, v_5) is $\frac{1}{8}$. But since it is also possible to reach v_5 from v_1 using v_3 instead, an event of probability $\frac{1}{8}$, we have that the probability p_{v_1,v_5} for an indivisible flow to start at v_1 and stop at v_5 is $p_{v_1,v_5} = \frac{1}{4}$. By repeating the same reasoning, $p_{v_1,v_1}, p_{v_1,v_2}, p_{v_1,v_3}$, and p_{v_1,v_4} are computed, and the formula given in (3.1) for the transfer entropic centrality $C_H(u)$ of $u = v_1$ tells us that $C_H(v_1) = \frac{3}{4}\log_2 4 + \frac{2}{8}\log_2(8) = 2.25$.

For a point of comparison, we also provide on the right of the same figure a scenario where probabilities are assigned non-uniformly. Suppose for example that we change the probability to go out of v_1, such that the edge (v_1, v_2) is chosen with a probability $\frac{1}{2}$, while the probability is $\frac{1}{6}$ for using the edges to the other nodes (including a probability $\frac{1}{6}$ that the flow just stays at v_1 itself). The resulting probabilities are provided on Figure 3.2 (right subfigure). There is no complication in computing $C_H(v_1)$ using the same formula (3.1) with non-uniform probabilities, it can be done from the moment one has information about the graph considered that justifies the probability choice. We observe in doing so here that the change in probability reduces slightly the centrality of v_1. This is consistent with the interpretation of entropic centrality: the underlying notion of entropy is a measure of uncertainty [18], the uncertainty of the final destination of a flow, knowing that it started at a given node. Imagine the most extreme case where the edge (v_1, v_2) is chosen with a probability 1, then even though v_1 has three potential outgoing neighbours, two of them are used with probability 0, so the centrality of v_1 would reduce considerably, as expected, since there is no uncertainty left regarding the destination of a flow at v_1.

The notion of transfer entropic centrality captured by (3.1) assumes that there is no vertex repetition in the paths taken by the flow. Since Figure 3.2 does not provide an illustration of this case, we consider a slightly more complex example in Figure 3.3. Again for the centrality of v_1, a flow leaves v_1, it can go to either v_2, v_3 or v_4. When reaching v_4, the flow cannot go back to v_1, since v_1 is already visited (and going back would not give a path anymore), there the probabilities to stay at v_4 and to go to v_5 from v_4 are modified. On the left, when probabilities are

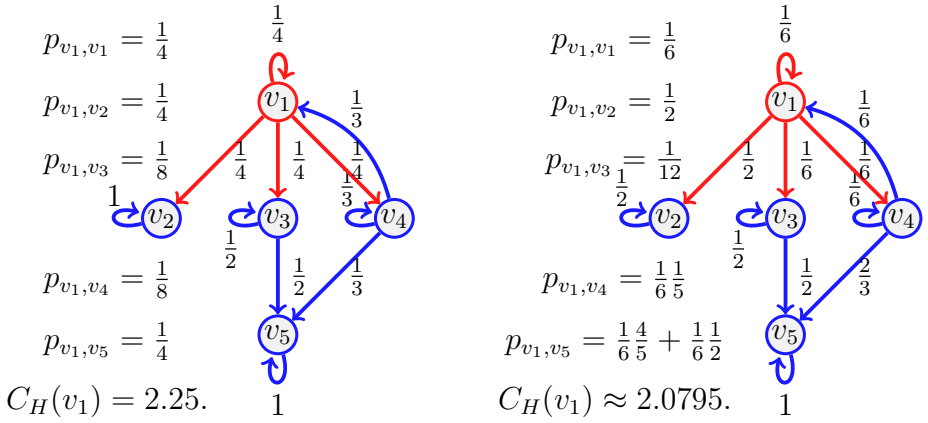


FIGURE 3.3: An example of transfer centrality involving already visited neighbours. If probabilities are uniform at random (on the left), they are scaled according to the number of unvisited neighbours. If not (on the right), they are scaled proportionally to the existing probabilities.

uniform, since now only two outgoing edges of v_4 are available, namely edges (v_4, v_4) and (v_4, v_5), each is assigned a probability of $\frac{1}{2}$. On the right, when probabilities are not uniform, we distribute the probability of going to some visited node(s) proportionally to the rest of the available edges. Since $\frac{4}{6}$ is going to v_5 while $\frac{1}{6}$ is staying at v_4, we have 4 and 1 out of 5 respectively leaving and staying, thus obtaining the renormalized probabilities as $\frac{4}{6} + \frac{4}{5}\frac{1}{6} = \frac{4}{5}$ and $\frac{1}{6} + \frac{1}{5}\frac{1}{6} = \frac{1}{5}$.

The examples of Figures 3.2 and 3.3 illustrate diverse cases that can be encountered with an indivisible flow. By definition of indivisibility, the choice of an edge for the flow at a node u to go to among its neighbours corresponds to choosing subsets containing one node only in the list of all subsets of neighbours. We can thus set a probability 0 to all subsets which contain more than one node. Therefore, the definition of entropic centrality in (3.1) as well as its variation where edges are assigned non-uniform probabilities are particular cases of the proposed split-and-transfer framework, that we discuss next.

3.2.2 The Split-and-Transfer Entropic Centrality

To model the situation described in the introductory motivating example (see also Figure 3.1), where we are interested in the centrality of $u = v_1$, we consider a flow leaving v_1 which can actually split before being transferred to some neighbours. To model the choice of splitting among possible neighbours, we first define a probability $q(x)$ over the set $\mathcal{E}_u = \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\},$

$p_{v_1,v_1} = 0$

$p_{v_1,v_2} = \frac{1}{3}(\frac{2}{3}) = \frac{2}{9}$

$p_{v_1,v_4} = \frac{2}{3}(\frac{3}{8}) = \frac{1}{4}$

$p_{v_1,v_3} = \frac{1}{3}(\frac{1}{6}) + \frac{2}{3}(\frac{1}{8})$

$p_{v_1,v_5} = \frac{1}{3}(\frac{1}{6}) + \frac{2}{3}(\frac{1}{8} + \frac{3}{8})$

$C_H(v_1) \approx 1.9076.$

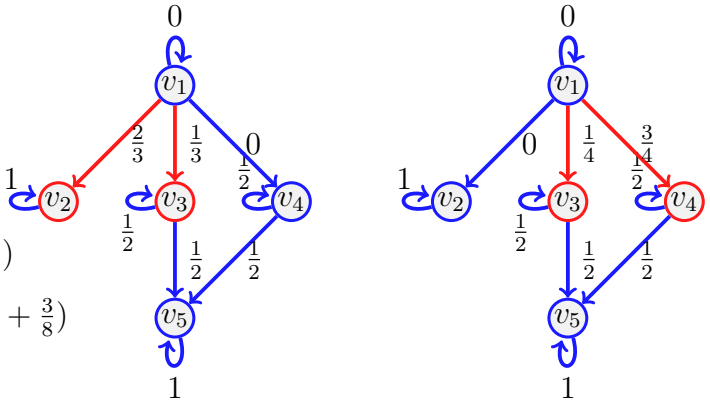


FIGURE 3.4: An example of split-and-transfer entropic centrality: on the left in red, the event corresponding to choosing $\{v_2, v_3\}$, on the right, the event $\{v_3, v_4\}$. The probabilities $p_{u,v}$ are computed by summing over both events, weighted by the respective event probability.

$\{v_1, v_2, v_3, v_4\}$ } such that, for our example, $q(\{v_2, v_3\}) = \frac{1}{3}$, $q(\{v_3, v_4\}) = \frac{2}{3}$, and $q(x) = 0$ for other choices of x. This represents the fact that 1/3 of the time, v_1 sends the goods to the pair $\{v_2, v_3\}$, while for the rest of the time, it sends it to the pair $\{v_3, v_4\}$. We compute the path probabilities for each event, for $q(\{v_2, v_3\}) = \frac{1}{3}$ on the left of Figure 3.4, and for $q(\{v_3, v_4\}) = \frac{2}{3}$ on the right. We have further information in the introductory example: when v_1 deals with $\{v_2, v_3\}$, there is a bias of $\frac{2}{3}$ for v_2 compared to $\frac{1}{3}$ for v_3, and the bias is of $\frac{3}{4}$ for v_4 in the other case. The corresponding probabilities are attached to the edges $\{(v_1, v_2), (v_1, v_3)\}$ and $\{(v_1, v_3), (v_1, v_4)\}$ respectively. Now that the edge probabilities are defined, we can compute the path probabilities. For example, from v_1 to v_5, we sum up the path probabilities for both events, weighted by the respective event probability: $\frac{1}{3}(\frac{1}{6}) + \frac{2}{3}(\frac{1}{8} + \frac{3}{8})$.

Now that we understand how the computation works on our motivating example, let us provide a general formula. We let a flow start at a node whose centrality we wish to compute, and at some point of the propagation process, a part f_u of the flow reaches a node u. Let $\mathcal{N}_u$ be the neighborhood of interest given f_u, that is, the set of outgoing neighbors which have not yet been visited by the flow. Every outgoing edge (u, v) of u exactly corresponds to some outgoing neighbor v, so in what follows, we may refer to either one or the other. Let $\mathcal{E}_u$ denote the set of possible outgoing edge subsets (where every edge (u, v) is represented by v). We

attach a possibly distinct probability $q(x)$ to every choice x in $\mathcal{E}_u$. Then

$$\sum_{x \in \mathcal{E}_u} q(x) = 1.$$

Every x in $\mathcal{E}_u$ corresponds to a set of edges (u, v) for v a neighbour. We further attach a weight $\omega_x(u, v)$ to every edge in x, with the constraint that

$$\sum_{(u,v) \in x} \omega_x(u, v) = f_u.$$

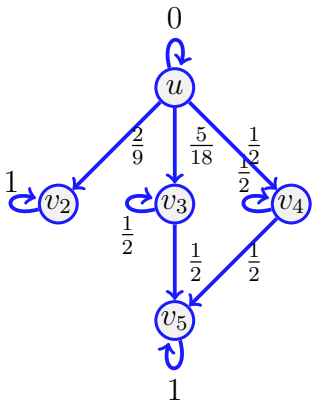
For example, we could choose all edges with equal weight, that is $\omega_x(u, v) = \frac{f_u}{i}$ for every (u, v) in x containing i edges, to instantiate the special case where the flow is uniformly split among all edges.

For a given node u, we compute the expected flow from u to a chosen neighbour v. Every such choice of x comes with a probability $q(x)$, and every edge (u, v) in x has a weight $\omega_x(u, v)$, which sums up to

$$f_{uv} = \sum_{x \in \mathcal{E}_{u,v}} q(x)\omega_x(u, v), \tag{3.2}$$

where $\mathcal{E}_{u,v}$ contains the sets in $\mathcal{E}_u$ themselves containing v.

Example 1. Consider the running example, with $u = v_1$. The set of neighbours of u is $\mathcal{N}_u = \{u, v_2, v_3, v_4\}$. We assign the following probabilities: $q(\{u\}) = q_1$, $q(\{v_2\}) = q_2$, $q(\{v_3\}) = q_3$, $q(\{v_4\}) = q_4$, $q(\{u, v_2\}) = q_5$, $q(\{u, v_3\}) = q_6$, $q(\{u, v_4\}) = q_7$, $q(\{v_2, v_3\}) = q_8$, $q(\{v_2, v_4\}) = q_9$, $q(\{v_3, v_4\}) = q_{10}$, $q(\{u, v_2, v_3\}) = q_{11}$, $q(\{u, v_2, v_4\}) = q_{12}$, $q(\{u, v_3, v_4\}) = q_{13}$, $q(\{v_2, v_3, v_4\}) = q_{14}$, $q(\{u, v_2, v_3, v_4\}) = q_{15}$, with $\sum_{i=1}^{15} q_i = 1$. We write down explicitly the terms involved in the sum (3.2) for two nodes, v_2 and v_3.



$$
\begin{aligned}
f_{u,v_2} = \; & q_2 f_u + q_5 \omega_{\{u,v_2\}}(u,v_2) + \\
& q_8 \omega_{\{v_2,v_3\}}(u,v_2) + q_9 \omega_{\{v_2,v_4\}}(u,v_2) + \\
& q_{11} \omega_{\{u,v_2,v_3\}}(u,v_2) + q_{12} \omega_{\{u,v_2,v_4\}}(u,v_2) + \\
& q_{14} \omega_{\{v_2,v_3,v_4\}}(u,v_2) + q_{15} \omega_{\{u,v_2,v_3,v_4\}}(u,v_2).
\end{aligned}
$$

$$
\begin{aligned}
f_{u,v_3} = \; & q_3 f_u + q_6 \omega_{\{u,v_3\}}(u,v_3) + \\
& q_8 \omega_{\{v_2,v_3\}}(u,v_3) + q_{10} \omega_{\{v_3,v_4\}}(u,v_3) + \\
& q_{11} \omega_{\{u,v_2,v_3\}}(u,v_3) + q_{13} \omega_{\{u,v_3,v_4\}}(u,v_3) + \\
& q_{14} \omega_{\{v_2,v_3,v_4\}}(u,v_3) + q_{15} \omega_{\{u,v_2,v_3,v_4\}}(u,v_3).
\end{aligned}
$$

Then

$$f_{u,u} + f_{u,v_2} + f_{u,v_3} + f_{u,v_4} = f_u \sum_{i=1}^{15} q_i = f_u.$$

By setting $q_8 = \frac{1}{3}$ and $\omega_{\{v_2,v_3\}}(u,v_2) = f_u \frac{2}{3}$, we find $f_{u,v_2} = f_u \frac{2}{9}$. Also, adding up $q_{10} = \frac{2}{3}$ and $\omega_{\{v_2,v_3\}}(u,v_3) = f_u \frac{1}{3}$, $\omega_{\{v_3,v_4\}}(u,v_3) = f_u \frac{1}{4}$, we find $f_{u,v_3} = f_u \frac{1}{9} + f_u \frac{1}{6} = f_u \frac{5}{18}$. Similarly $f_{u,v_4} = f_u \frac{1}{2}$ and indeed $f_u \frac{2}{9} + f_u \frac{5}{18} + f_u \frac{1}{2} = f_u$.

We repeat the computations for f_{v_3,v_5} and f_{v_4,v_5}. For that, we need to know what is f_{v_3} and f_{v_4}, but in this case, since both v_3 and v_4 only have one incoming edge, we have that $f_{v_3} = f_{v_1,v_3}$ and $f_{v_4} = f_{v_1,v_4}$:

$$f_{v_3,v_5} = \tfrac{1}{2} f_{v_3} = f_u \tfrac{1}{2} \tfrac{5}{18}, \;\; f_{v_4,v_5} = \tfrac{1}{2} f_{v_4} = f_u \tfrac{1}{2} \tfrac{1}{2}, \;\; f_{v_5} = f_{v_3,v_5} + f_{v_4,v_5} = f_u \tfrac{7}{18}.$$

It is true that by setting $f_u = 1$, we have $f_{u,v_2} = \frac{2}{9} = p_{u,v_2}$ as computed in Figure 3.4, but this is true because $p_{v_2,v_2} = 1$. If we consider v_3 instead, we find $f_{u,v_3} = \frac{5}{18} = 2p_{u,v_3}$, this is because we have computed what reaches v_3, but since v_3 has an outgoing edge, we need to distinguish what stays from what continues.

Notice that by setting $f_u = 1$ and $f_{v_3} = f_{v_4} = 1$, we get

$$f_{u,v_2} = \tfrac{2}{9}, \;\; f_{u,v_3} = \tfrac{5}{18}, \;\; f_{u,v_4} = \tfrac{1}{2}, \;\; f_{v_3,v_5} = \tfrac{1}{2}, \;\; f_{v_4,v_5} = \tfrac{1}{2}.$$

We can then assign to edge (v_i, v_j) the probability f_{v_i,v_j} (obtained with $f_u = 1$) as reported on the graph edges.

The property of flow conservation observed in the example holds true in general, which we shall prove next. Indeed, when v varies in $\mathcal{N}_u$, the sets $\mathcal{E}_{u,v}$ appearing

in the summation $\sum_{v\in\mathcal{N}_u}\sum_{x\in\mathcal{E}_{u,v}} q(x)\omega_x(u,v)$ may intersect, so for each choice x, one can gather all the $\mathcal{E}_{u,v}$ that contains x. For this x, we find a term in the above sum of the form

$$q(x)\sum_{(u,v)\in x}\omega_x(u,v) = q(x)f_u.$$

Then

$$\sum_{v\in\mathcal{N}_u}\sum_{x\in\mathcal{E}_{u,v}} q(x)\omega_x(u,v) = \sum_{x\in\mathcal{E}_u} q(x)f_u = f_u.$$

This shows that the flow from u to v is conserved over all the neighbours $v\in\mathcal{N}_u$ given f_u. Thus, by setting $f_u = 1$, the quantity

$$f_{uv} = \sum_{x\in\mathcal{E}_{u,v}} q(x)\omega_x(u,v)$$

becomes a probability, and in fact, putting this probability on the edge (u,v) in the context of the transfer entropic centrality gives the same result as the above computations using the split-and-transfer model, as in fact already illustrated on the figure in Example 1, since the probabilities displayed on the edges of the graph have been computed in this manner.

Definition 3.1. The split-and-transfer entropic centrality $C_{H,p}(u)$ of a vertex u is given by

$$C_{H,p}(u) = -\sum_{v\in V} q_{uv}\log_2(q_{uv})$$

where $q_{uv} = \sum_{x\in\mathcal{E}_{u,v}} q(x)\omega_x(u,v)$ is computed from (3.2) with $f_u = 1$ and the standard convention that $0\cdot\log_2 0 = 0$ is applied. The index p in $C_{H,p}$ emphasizes the dependency on the choice of the probability distribution p. Then we have $C_{H,unif}$ when p is uniform as a particular case. We also alternately use the notation C_H without indicating the distribution when the context of the particular probability distribution is either clear, or not the aspect of emphasis.

We notice the similarity with (3.1), which shows that the split-and-transfer entropic centrality is equivalent to the transfer entropic centrality, assuming the suitable computation of edge probabilities.

The notion of split-and-transfer entropic centrality characterizes the *spread* of a flow starting at u through the graph. Now two nodes may have the same spread, but one node may be dealing with an amount of goods much larger than the other.

In order to capture the *scale* of a flow, we also propose a scaled version of the above entropy.

Definition 3.2. The scaled split-and-transfer entropic centrality is accordingly given by $F(f_u)C_H(u)$ where F is a scaling function.

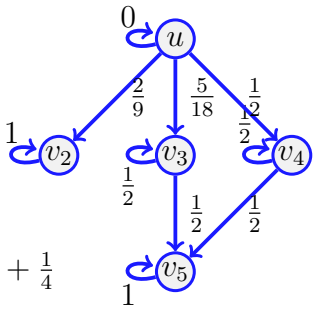
The scaling function F may depend on the nature of the underlying real world phenomenon being modelled by the graph, with $F(f_u) = f_u$ being a simple default possible choice (other standard choices are $F(f_u) = \sqrt{f_u}$ or $F(f_u) = \log(f_u)$). We use the default choice in the example below.

Example 2. Continuing with the same example, we use the edge probabilities as obtained in Example 1 to compute the transfer entropic centrality from Definition 3.1. The scaled entropic centralities of $u = v_1$ and v_3 are simply $f_u C_H(v_1) \approx f_u 1.9076$ and $f_{v_3} C_H(v_3) = f_{v_3}(\frac{1}{2}\log_2(2) + \frac{1}{2}\log_2(2)) = f_{v_3}$. Without the scaling factor, C_H is a measure of spread, and it makes sense that $C_H(v_1) > C_H(v_3)$.

$p_{v_1,v_1} = 0$

$p_{v_1,v_2} = \frac{2}{9}$

$p_{v_1,v_4} = \frac{1}{4}$



$p_{v_1,v_3} = \frac{5}{36}$

$p_{v_1,v_5} = \frac{5}{36} + \frac{1}{4}$

$C_H(v_1) \approx 1.9076.$

However if v_1 is actually distributing some items in overall small amounts, while v_3 is not only getting this item from v_1 but also producing it and furthermore sending it only to v_5 but in large amounts, then the scaling factor could be used to refine the analysis and account for this extra information. In this example, from the moment $f_{v_3} \geq 1.9076 f_{v_1}$, v_3 will be deemed more important than v_1 as per the scaled entropic centrality measure.

3.2.3 The Split-and-Transfer Entropic Centrality in the Uniform Case

We show how the framework presented in the Section 3.2.2 simplifies to the results of [45] in the case where the distribution $q(x)$ is uniform (that is, any possible subset of edges is chosen uniformly at random among all possible choices of subsets of edges) and the edges are chosen with equal weight, that is $w_x(u, v) = \frac{f_u}{i}$ for every edge (u, w) in the set x containing i edges.

Recall that $\mathcal{N}_u$ is the set of outgoing neighbors of u which have not yet been visited by the flow. Let $d'_{out}(u) = |\mathcal{N}_u|$ be its "effective" out-degree (since the

out-degree would count every outgoing neighbour). There are $2^{d'_{out}(u)} - 1$ choices of edge subsets for a flow at u to split-and-transfer to. Let $\mathcal{E}_u$ denote this set of possible outgoing edge subsets (where every edge (u, w) is represented by w):

$$|\mathcal{E}_u| = 2^{d'_{out}(u)} - 1.$$

Then $\mathcal{E}_u$ can be partitioned among sets $\mathcal{E}_{i,u}$ of i outgoing edges, where i is at least 1, and at most $d'_{out}(u)$:

$$\mathcal{E}_u = \sqcup_{i=1}^{d'_{out}(u)} \mathcal{E}_{i,u} = \sqcup_{i=1}^{d'_{out}(u)} (\mathcal{E}_{i,u,v} \sqcup \mathcal{E}_{i,u,\bar{v}})$$

where we have further distinguished subsets containing or not a given neighbour v. Thus we have:

$$\sum_{i=1}^{d'_{out}(u)} |\mathcal{E}_{i,u}| = \sum_{i=1}^{d'_{out}(u)} \binom{d'_{out}(u)}{i} \tag{3.3}$$

$$|\mathcal{E}_{i,u,v}| + |\mathcal{E}_{i,u,\bar{v}}| = \binom{d'_{out}(u) - 1}{i - 1} + \binom{d'_{out}(u) - 1}{i} \tag{3.4}$$

since once v is fixed, there are $\binom{d'_{out}(u)-1}{i}$ subsets of $i+1$ edges containing v (given v, there is one edge from u to v, then among the $d'_{out}(u)-1$ neighbors left, choose any i of them). Also note the consistency with (3.3): $\binom{n-1}{k} + \binom{n-1}{k-1} = \frac{(n-1)!}{k!(n-1-k)!}\frac{n-k}{n-k} + \frac{(n-1)!}{(k-1)!(n-k)!}\frac{k}{k} = \frac{n!}{(n-k)!k!}$.

In the uniform case, where $q(x) = \frac{1}{2^{d'_{out}(u)}-1}$ for all x, using (3.3), we indeed have that

$$\sum_{i=1}^{d'_{out}(u)} \sum_{x \in \mathcal{E}_{i,u}} q(x) = \frac{1}{2^{d'_{out}(u)}-1} \sum_{i=1}^{d'_{out}(u)} \binom{d'_{out}(u)}{i} = 1.$$

From (3.4), there are $\binom{d'_{out}(u)-1}{i-1}$ choices of edges that bring flow from u to v, for $i = 1, \ldots, d'_{out}(u)$. Every such a choice x comes with a probability $q(x)$, and every edge (u, v) in x has a weight $\omega_x(u, v)$, which sums up to

$$f_{uv} = \sum_{i=1}^{d'_{out}(u)} \sum_{x \in \mathcal{E}_{i,u,v}} q(x)\omega_x(u, v),$$

where $q(x) = \frac{1}{2^{d'_{out}(u)}-1}$ and $\omega(u,v) = \frac{f_u}{i}$. Using (3.4), we get

$$\begin{aligned} f_{uv} &= \frac{1}{2^{d'_{out}(u)}-1} \sum_{i=1}^{d'_{out}(u)} \binom{d'_{out}(u)-1}{i-1} \frac{f_u}{i} \\ &= \frac{f_u}{2^{d'_{out}(u)}-1} \sum_{j=0}^{d'_{out}(u)-1} \binom{d'_{out}(u)-1}{j} \frac{1}{j+1} \\ &= \frac{f_u}{d'_{out}(u)} \end{aligned}$$

since

$$\frac{1}{i+1}\binom{n-1}{i} = \frac{(n-1)!}{(i+1)!(n-1-i)!} = \frac{n!}{(i+1)!(n-1-i!)n}$$

which implies that

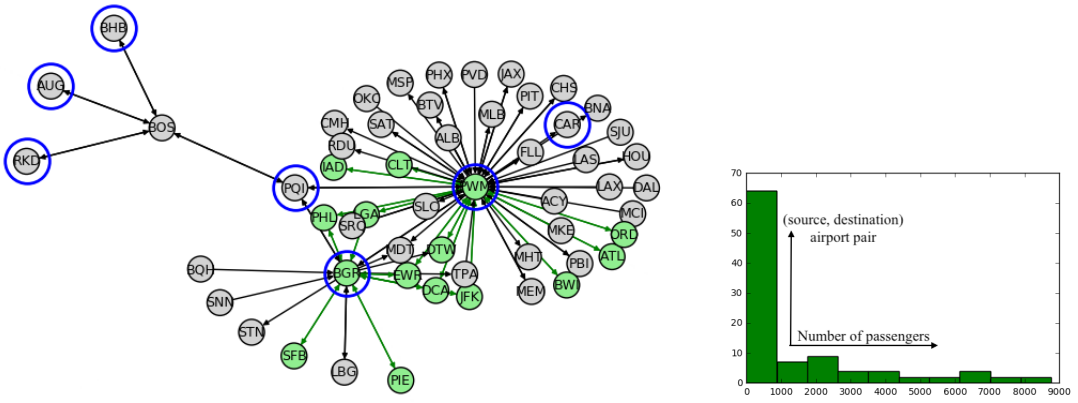
$$\sum_{i=0}^{n-1} \frac{1}{i+1}\binom{n-1}{i} = \frac{1}{n}\sum_{i=0}^{n-1}\binom{n}{i+1} = \frac{1}{n}\sum_{j=1}^{n}\binom{n}{j} = \frac{1}{n}(2^n-1).$$

3.3 Case Studies

The split-and-transfer model studied in this chapter captures many real world scenarios. We carry out studies with three distinct scenarios to illustrate this point: (1) Maine airport network induced by connecting flights; (2) Organizational cross-shareholding of entities listed in the Tehran stock market; and (3) Bitcoin (wallet) address network; which is a network inferred from the Bitcoin transaction log. Furthermore, we consider two graph models of Bitcoin address network: one which reflects the weights and co-occurrence of specific subset of edges; and a simplified model where we discard the actual amount aggregated over transactions and instead focus only on the existence of edges, to reflect connectivity among the nodes.

3.3.1 Maine Airport Network

As a first scenario, we consider a network of airports, with their connecting flights. We choose Maine airports since we would like to start with a small network, easy to visualize and understand. On Figure 3.5a, the Maine flight network is represented for the period of January 2018 (according to `https://`



(A) Maine flight network. (B) Distribution of passengers.

FIGURE 3.5: On the left, the circled nodes are Maine airports, the other nodes are airports from/to which there were flights connecting Maine during January 2018. Green nodes have the highest centralities. On the right, the frequency of airport source-destination pairs is given as a function of the volume of passengers.

`www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=292` using only passengers, this includes flights with one or two persons, such as those offered by Dassault Falcon Service, see also [19] for accessing the data used). The circled nodes are airports in Maine (PQI = Presque Isle/Houlton, PWM = Portland, AUG = Augusta/Waterville, BGR = Bangor, RKD = Rockland, CAR = Caribou, BHB = Bar Harbor).

There is a directed edge from a node u to another node v if there is at least one flight from airport u to airport v. Most pairs u, v of airports have both edges (u, v) and (v, u) since flights are usually going in both directions. Nodes that are not circled (meaning that they are not located in Maine) have outgoing edges only if these are connections to Maine (connections to airports outside Maine are not considered). Edges are weighted, the weight is the number of passengers. Since the graph is directed, every node has an in-degree and an out-degree (see Table 3.1a for the in-degrees and out-degrees for the 7 Maine airports). Note that the in/out-degrees do not sum up to be equal, since the degrees are contributed from edges with airports outside Maine also.

The node PWN stands out for both its high in-degree and out-degree: PWN corresponds to Portland International Jetport, cited as the busiest airport of Maine[1]. By collecting how many passengers are going from one given departure airport to a given destination, weights are attributed to edges as shown in Figure 3.5b.

[1]https://en.wikipedia.org/wiki/Portland_International_Jetport

node	in	out
PQI	3	3
PWM	35	33
AUG	1	1
BGR	12	14
RKD	1	1
CAR	1	0
BHB	1	1

(A) In-/out-degrees.

node	uniform	$C_H(u)$	passengers	$f_u C_H(u)$	between.
PWM	5.2542242	3.6768458	55996	2.79512	0.55450
BGR	4.8991375	3.6262334	16051	0.79018	0.15583
PQI	4.5812700	2.2985885	781	0.02437	0.10901
RKD	2.5449986	2.1780402	283	0.00837	0
AUG	2.5449986	2.1764987	275	0.00813	0
BHB	2.5449986	2.1715783	274	0.00808	0
CAR	0	0	0	0	0

(B) Entropic and betweenness centralities for the Maine airports network.

TABLE 3.1: Characteristics of the Maine airport network: in-/out-degrees and centralities.

We thus have an underlying graph to be analyzed. Nodes and edges are given as described above, and the edge probabilities can be chosen to be inversely proportional to the out-degree (the edge probability is then uniform) to capture how well connected an airport is, or proportional to the corresponding edge weights (yielding a non-uniform edge probability) to incorporate the number of passengers as information for the analysis. For the uniform case, the out-degree includes a self-loop that represents the probability that a flow stops at a node. In the non-uniform case, self-loops do not come with a natural interpretation (passengers are not staying in the airport, and while some transit and other do a round-trip, this information is not captured in the data we have). Thus, to model the probability that a flow leaving from u stops at v (meaning that u influences v), a self-loop is introduced. In this case, we propose the following choice: for a node u with out-degree d, and set of neighbours $\mathcal{N}_u$, define a self-weight $w(u,u)$ such that $\frac{w(u,u)}{w(u,u)+\sum_{w\in\mathcal{N}_u} w(u,w)} = \frac{1}{a}$ for a a chosen constant. Then by construction, the probability assigned to the self-loop is $\frac{1}{a}$.

The entropic centralities $C_{H,unif}$ for the uniform case and $C_{H,p}$ for the non-uniform case with $1/a = 1/5 = 0.2$ are shown in Table 3.1b. For the scaled entropic centrality given in the last column, the value f_u is chosen to be the normalized number of passengers. For example, $f_u = \frac{55996}{73660} = 0.76020$ for PWM. We expect the way the flow propagates from Maine airports to the rest of the network to be

an indicator of how each Maine airports connects Maine both within, and outside Maine. The uniform case looks at these connections just in terms of how many there are connecting a given airport to others, the non-uniform case takes into account how frequently used each of these connections actually are, while the scaled entropic centrality further incorporates the number of passengers at each airport.

That PWM has the highest centrality for the three flavours of entropic centrality is consistent with the fact that it is the busiest airport of Maine, in terms of both numbers of connections and passengers. At the other end of the centrality range, CAR has entropy 0 because it has no outgoing edge.

We observe that the node ranking in the 2nd column follows that of their out-degree: this makes sense, because a high out-degree gives a first advantage when it comes to spread. However there are interesting subtleties. While the out-degrees are spread apart (PWN has 33, BGR 14 and PQI 3), the differences in entropic centralities are much smaller, which is explained by the fact that both BGR and PQI are having flights to PWN, and thus they inherit some of the influence of PWN, something that the out-degree itself cannot capture.

When non-uniform edge probabilities are considered, the gap among the different centralities reduces, this is because some of the connections are used most of the time, while others are barely used, thus the actual spread factor is considered less. The case of RKD, AUG and BHB furthermore illustrates the impact of considering non-uniformly split flows. All three have identical roles as per the graph structure, and thus have identical entropic centrality in the uniform case. However, when we consider the non-uniform case, the BOS-RKD, BOS-AUG and BOS-BHB edges have edge probabilities of 0.126500732, 0.128257687 and 0.134114202 respectively. Consequently, a flow originating from any one of these could continue to go to the other two (but not to itself), and to PQI. Note that the probabilities for an actual flow instance will have to be renormalized based on the edges that they can take, while maintaining the relative ratios among those edges. The edge BOS-PQI has probability of 0.411127379. Thus, the flow starting at BHB will experience the largest relative skew and corresponding skewed concentration of flow, and thus, it has the lowest entropic centrality, and conversely, among these three, RKD has the highest entropic centrality.

The scaled entropic centrality given in the last column weights the centrality with the normalized number of passengers. There is a clear hierarchy, and in fact the gap between PWM and BGR is larger than in the uniform case, which

TABLE 3.2: Relative rankings using different centralities. The top nodes as per the non-uniform entropic centrality $C_{H,p}$ from the entire network of 55 nodes (with their relative ranking) are shown in the first column. For the alpha, PageRank and Katz centralities, we used a graph with flipped edge direction.

vertex ($C_{H,p}$ rank)	out	$C_{H,unif}$	alpha/Katz ($\alpha = 0.1$)	PageRank (UW, $\alpha = 0.85$)	PageRank (W, $\alpha = 0.85$)
PWM (1)	1	1	1	1	1
ACY (2)	6	5	5	4	12
ALB (2)	6	5	5	4	18
DAL (2)	6	5	5	4	17
LAS (2)	6	5	5	4	14
LAX (2)	6	5	5	4	11
MCI (2)	6	5	5	4	21
MKE (2)	6	5	5	4	20
OKC (2)	6	5	5	4	18
SJU (2)	6	5	5	4	15

makes sense since PWM had more than 3 times more passengers than BGR over the chosen period. Thus, while the entropic centrality helps capture the network effect, the weighted entropic centrality allows us to capture the cumulative effect of local characteristics as well as the network.

The betweenness centrality of u is the ratio of the number of shortest paths through u to the total number of shortest paths. From the results we observe that it agrees with the entropic centralities for the first 3 airports, however it fails to distinguish the others.

In Table 3.2, we show the top 10 airports (out of the entire network of 55 airports) as per the non-uniform entropic centrality $C_{H,p}$ with their ranking (several in fact have a joint second position). We include their rankings as per other centralities, including three eigenvector based centralities: the alpha centrality ([61] computed using the iGraph implementation ([160]), Katz centrality ([62]) and PageRank ([56], both Katz and PageRank are implemented in NetworkX ([161])), for all of which the graph with reversed edges was considered. This is to allow a meaningful comparison, since the importance of vertices for these 3 centralities are determined by the incoming edges, in contrast to the flow based centralities, for which importance depends on the spread extent of the flow, thus the outgoing edges.

The uniform entropic centrality $C_{H,unif}$, the alpha centrality, Katz centrality and PageRank centrality (UW) all consider the graph to be directed and

TABLE 3.3: Kendall rank correlation coefficient τ_κ across centralities for the 55 vertex airport network.

	$C_{H,unif}$	$C_{H,p}$	alpha/Katz ($\alpha = 0.1$)	PageRank (UW)	PageRank (W)
$C_{H,unif}$	0	0.1730639	0.0242424	0.2087542	0.2484848
$C_{H,p}$	0.1730639	0	0.1555555	0.2228956	0.2962962
alpha/Katz	0.0242424	0.1555555	0	0.2114478	0.239057239
PageRank(UW)	0.2087542	0.2228956	0.2114478	0	0.2713804
PageRank(W)	0.2484848	0.2962962	0.2390572	0.2713804	0

unweighted, in which case its largest eigenvalue λ_1 is $\lambda_1 \simeq 6.51111$, and $\frac{1}{\lambda_1} \simeq 0.153583428$. Since we need $\alpha < \frac{1}{\lambda_1}$ for both alpha and Katz centralities, we used $\alpha = 0.1$, while for PageRank, we adopt the NetworkX default damping parameter $\alpha = 0.85$. Repeating the arguments by [64], eigenvector based centralities are not the best suited to capture transfer type of behaviours. However it is still interesting to notice that the rankings of $C_{H,unif}$, alpha and Katz agree (PageRank also gives the same ranking to all nodes ranked 2). It is not a surprise that the alpha and Katz centralities agree (their formula differ by a constant). The fact that $C_{H,unif}$ also agrees could be explained by the structure of the network, which forces arbitrary walks to in fact be close to paths. Betweenness is not presented here since the network we consider has a star graph like shape - more precisely it looks like 3 stars conjoined together, and only the nodes whose in/out-degrees are at least three have non-zero betweenness centrality score.

$C_{H,p}$ and PageRank (W) are computed on the weighted graph. But for PWM, which is outstanding by all metrics, the rankings are uncorrelated. One explanation for this distinctiveness is the use of a stochastic matrix in the PageRank algorithm, which takes into account the weights of the edges and tends to give a higher rank to vertices with high weight edges, while weights are influencing $C_{H,p}$ in an indirect manner.

To understand quantitatively how distinct the results are across the algorithms, in Table 3.3 we synopsize the Kendall's tau coefficient τ_κ ([4]) which indicates the rank correlation among a pair of ranked lists. The lower the value of this coefficient, the closer (similar) two ranked lists are. We see that $C_{H,unif}$ produces results which are very similar to alpha and Katz centralities (which had mutually

identical results for this specific graph[2]), but $C_{H,p}$ yields a reasonably distinct result instead. Furthermore, results from PageRank applied to both weighted and unweighted graphs are most distinct both with respect to entropic centralities, as well as the other existing centralities explored in our experiments.

3.3.2 Organizational Cross-shareholding in Tehran Stock Market

We next consider 131 companies from the Tehran Stock Market, based on the dataset from [20][3]. They form the vertex set of the next graph, a cross-shareholding network of companies which have shares of other companies. There is an edge between i and j if company i belongs to the investment portfolio of company j, i.e., j owns some share of i. Therefore the in-degree of node j is the number of companies that belong to the investment portfolio of company j. Conversely, the out-degree of node j is the number of companies that are shareholders of j. Edges are weighted, edge (i, j) has for weight the percentage of shares that company j has from company i. We will consider both this graph, shown in Figure 3.6 and the graph obtained by changing the directionality of the edges.

Nodes highlighted in green in Figure 3.6 have one edge with weight more than 0.5, meaning that more than 50% of their shares are owned by another company, otherwise they are in grey. Nodes highlighted in red, superseding the other coloring, have the highest in-degrees, which means that they own shares of many other companies. They are nodes 121 (Adashare), 85 (NIKX1), 123 (Oilcopen), 42 (SA3A1), 119 (tamin org), and 118 (government). Nodes highlighted in blue have the highest out-degrees, which means that their shares are owned by many other companies (but possibly in small amounts). They are nodes 110 (BMEX1), 2 (CH12), 21 (FO041), 41 (GD021), 3 (GO02) with degree 7 and 69 (PFAX1), 1 (MADN), 20 (MS022) and 62 (PK061) with degree 8.

Now that we have a graph structure to analyze, given by the afore described nodes and edges, we need to assign probabilities to edges. To do so, we use the edge weight, and fix the edge probability to be inversely proportional to its weight. As for the self-loops and their probability, unlike in the case of Maine airports data set, there is a natural interpretation here. Since the outgoing edges of node j indicate

[2]The normalized absolute centrality scores were however distinct across alpha and Katz centralities.

[3]We thank the authors of [20] for sharing their data with us.

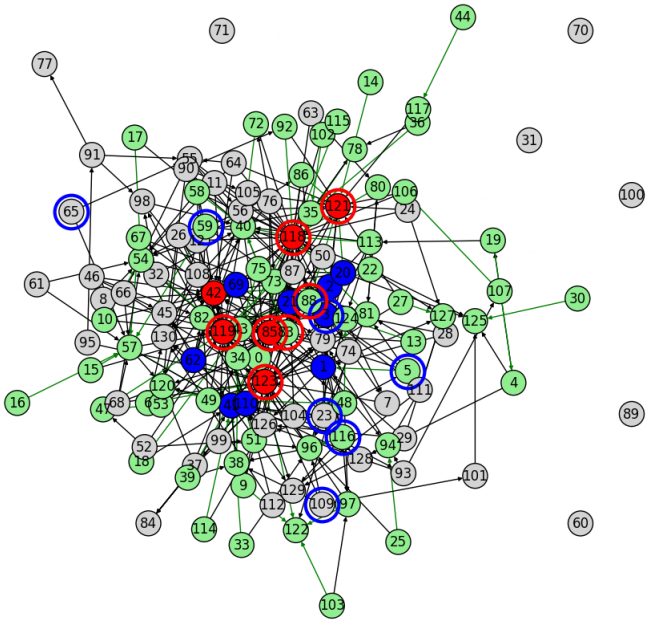


FIGURE 3.6: Cross-shareholding network of Teheran Stock Market companies: Red and blue nodes have resp. the highest in- and out-degrees. Nodes circled in blue resp. red have the highest entropic centrality under the current and reverse edge directionality.

no	ID	$C_H(u)$	market (f_u)	$f_u C_H(u)$	out	neighbours
23	ARFZ1	3.1990	0.7153	2.2882_3	6	(**1**, 0.1811), (**2**, 0.3536), (**41**, 0.05) (5, 0.0453), (43, 0.0453), (85, 0.04)
3	GO02	3.0205	0.9635	2.9103_1	7	(**1**, 0.2241),(**21**, 0.0994) (82, 0.0112),(7, 0.0377) , (42, 0.0282) (121, 0.1854), (43, 0.3775)
109	IPAR1	2.9680	0.6874	2.0402_6	5	(96, 0.2), (97, 0.0789), (38, 0.139) (123, 0.1732), (101, 0.0963)
59	PRDS1	2.9541	0.7951	2.3488_4	5	(88, 0.0111), (57, 0.6685), (82, 0.0113) (118, 0.054), (55, 0.0502)
65	PK3A1	2.8857	0.6267	1.8084_7	2	(57, 0.42), (55, 0.2067)
5	KNRX	2.8817	0.9415	2.7131_2	3	(**1**, 0.8876), (83, 0.033), (**3**, 0.0209)
116	PRSX1	2.8273	0.8086	2.2861_5	6	(96, 0.648), (97, 0.0494), (129, 0.0103) (88, 0.0717), (**41**, 0.0113), (79, 0.0179)

TABLE 3.4: Nodes with the highest entropic centrality, their scaled entropic centrality, where f_u is the market size in percentage (their relative ranking is marked with subscript), as well as the out-degrees and neighbours with the weight of the connecting edges. Bold faced nodes have one of the highest out-degrees.

the companies that are shareholders of j, the self-loop refers to j still owing some of its own shares, and the amount is 100% minus what the other companies own (note that share ownerships with negligible amounts were not taken into account in the data set we used, so the self-loop will absorb these portions).

Table 3.4 lists the seven nodes with the highest entropic centrality. The interpretation of entropic centrality is that we are looking at the nodes whose shares are

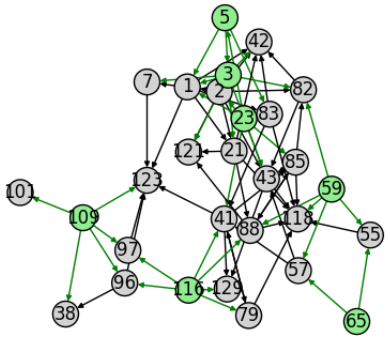


FIGURE 3.7: A subnetwork comprising only highly central nodes (in green, with their outgoing edges also in green) as listed in Table 3.4 and their neighbours.

"most diversely owned" in terms of their shares being owned by different companies, whose shares are in turn themselves owned by others. The economic fortunes of the company whose centrality is looked at also affect those of the other companies, and the entropic centrality thus indicates the impact a particular company's economic performance would have on the rest.

We immediately see that this centrality measure is different from out-degree centrality. We can however look at how they relate, by considering the role of out-degrees not only on the nodes but also on their neighbours. We observe that only node 3 has one of the highest out-degrees, however, nodes 23 and 116 still have high out-degrees, but also are connected to neighbours which have high out-degrees, in fact, node 23 which has the highest entropic centrality has three high out-degree neighbours. For nodes 109 and 59, they still have a relatively high out-degree. For 5, it has a fairly low out-degree, but out of the 3 neighbours, two have high degrees themselves. The case of node 65 is particularly interesting, since it has only two neighbours (refer to Figure 3.7), namely 55 and 57. Neither of 55 nor 57 has a high centrality individually, but they together provide node 65 a conduit to a larger subgraph than the individual transit nodes themselves do, illustrating the secondary effects of flow propagation.

The cross-shareholding network of Teheran Stock Market was analyzed in [20], where a closeness centrality ranking is shown to be almost identical to the degree based centrality one. The entropic centrality ranking in contrast manages to capture a different dynamics, by involving the spread of influence via flow propagation, together with a quantitative edge differentiation.

The above approaches furthermore ignore any other additional information regarding the nodes' importance, such as the market size share of the organizations.

Arguably, between two organizations with identical position in the graph, the one with larger market size may be deemed to have larger influence on the other nodes. This is modelled by the scaled entropic centrality $C_H(u)f_u$, where f_u is the market size in percentage. We notice that this indeed creates a distinct relative ranking (indicated by subscript in Table 3.4, for example, ARFZ1 is ranked 3rd as per weighted entropic centrality). Particularly, among the top seven companies as per $C_H(u)$, we see that only PRDS1 continues to be in the same (fourth) rank. KNRX has the largest change in ranking, up from sixth to second. While the scaled entropic centrality ranking of the top two nodes are congruent with the ranking based solely on the scale factor (market size), we see that ARFZ1, which would be ranked 5th by market size, and first by solely network effect, is ranked 3rd when both factors are taken into account.

We again compare the entropic centrality $C_{H,p}$ with respect to the alpha, Katz and PageRank centralities (using the reverse edge direction). Note that the unweighted graph has for largest eigenvalue $\lambda_1 \simeq 2.99715780$ (so $\frac{1}{\lambda_1} \simeq 0.3336494$). The ranking results for the most central entities from the Tehran stock exchange, and the overall Kendall tau rank correlation coefficients are reported in Tables 3.5 and 3.6 respectively. The entity 23 is outstandingly central with respect to all considered metrics but weighted betweenness. The alpha and Katz centralities yield, as expected, very similar results, but they rank the entity 3 as third instead of second. They rank second the entity 20. A likely explanation could be that that 20 actually has a higher out-degree (and thus a higher in-degree in the reversed edge network) than entity 3. The top 7 most central entities have mostly a 0 betweenness (ranked 39), and are mostly ranked pretty low with respect to both versions of PageRank, weighted and unweighted. The most central entity for the weighted betweenness is 85, which is one of the most central with respect to in-degree (it has an in-degree of 18). Then 111 and 76 are second respectively for the unweighted and weighted PageRank. Entity 111 has out-neighbours 88, 81, 127, 94 and 112 which become in-neighbours in the reversed edge graph. Neither 111 itself nor its neighbours stand out by their degrees, however 76 has for in-neighbours in the reversed edge graph 72, 73, 130, 85, 118 and 76, where both 118 and 85 are very central with respect to in-degree, which is why it is ranked high.

The Kendall rank correlation confirms that the entropic centrality differs from the other metrics not only in determining the most central vertices but also overall. The Kendall coefficient for unweighted betweenness has not been reported since

TABLE 3.5: Relative ranking for different centralities for the top 7 entities from Table 3.4 with highest $C_{H,p}$. The first column indicates the vertex identifiers.

no ($C_{H,p}$)	alpha ($\alpha = 0.1$)	Katz ($\alpha = 0.1$)	PageRank (UW) $\alpha = 0.85$	PageRank (W) $\alpha = 0.85$	Betweenness (weighted)
23 (1)	1	1	1	1	39
3 (2)	3	3	69	69	3
109 (3)	22	22	7	4	39
59 (4)	14	14	18	26	39
65 (5)	71	71	30	12	39
5 (6)	26	23	67	37	31
116 (7)	8	7	12	9	39

TABLE 3.6: Kendall rank correlation coefficient τ_κ across the centralities for the Tehran stock exchange.

	$C_{H,p}$	alpha ($\alpha = 0.1$)	PageRank (UW) $\alpha = 0.85$	PageRank (W) $\alpha = 0.85$	Katz ($\alpha = 0.1$)
$C_{H,p}$	0	0.225935	0.318585	0.332645	0.225290
alpha	0.225935	0	0.31716	0.378709	0.004774
PageRank(UW)	0.318580	0.31716	0	0.131741	0.316
PageRank(W)	0.332645	0.378709	0.131741	0	0.377806452
Katz	0.225290	0.004774	0.316	0.377806	0

only 38 vertices have a non-zero betweenness. This shows (and is because) that the graph considered is far from being strongly connected.

Both comparison tables reinforce that the entropic centrality $C_{H,p}$ provides a new perspective not captured by the other algorithms.

We next consider the same graph but where edge directionality is reversed. In this setup a node is central if it owns diverse companies, which themselves may in turn own various companies. Since owning shares could be used to influence an organization's management, the entropic centrality based on reverse edges is thus a proxy indicator of how much control a specific entity has over the other entities in the market. Organizations with very high entropic centralities using either sense of edge direction could then be seen as probable candidates causing structural risks - be it by being 'too big to fail', or having too much control over significant portions of the market for it to be fairly competitive.

The nodes with highest entropic centrality are shown in Table 3.7. The most important one is the government, which makes sense: we expect it to be one of the most important players in Iran when it comes to owning shares in other companies (and yet not to appear in the list when the original edge direction is considered). We see a higher correlation between entropic centrality and in-degree than there

no	ID	$C_H(u)$	market	in	neighbours
118	gov	5.343483526	9.8	40	(**85**,0.17370)
119	tamin org	4.698914925	5.7647	31	(**42**, 0.0305)
121	Adashare	4.477764513	6.7827	17	
88	BTEJ1	4.291673061	0.7143	9	(**85**, 0.474)
123	Oilcopen	3.915417631	3.5699	22	
83	TMEL1	3.809900739	0.4849	9	
85	NIKX1	3.769308035	0.9722	17	

TABLE 3.7: Most central nodes with respect to entropic centrality, when edge directionality is reversed. The column market refers to the market size in percentage. Then in-degree and neighbours whose in-degree is one of the highest are given, with the connecting edge weight.

was between entropic centrality and out-degree in Table 3.4. Among the seven most central nodes, five of them are having one of the highest in-degrees, the two most central nodes have themselves one high degree neighbour. Then node 88 has a fairly low in-degree, but it is connected to node 85.

In summary, the case study of the Tehran stock exchange network illustrates three important intertwined aspects of our model. Foremost, it shows that the model is flexible, and can seamlessly capture the effect of relationships, considered either in a binary fashion (just the *structure*), or quantified with relative strengths (the *skew* in strength of the relationships), while it can also accommodate information which is intrinsic to the node but somewhat disentangled from the network (used as a *scale* factor). Second, reversing the edge direction gives a dual perspective. As an aside, the model can also be applied to undirected graphs by considering directed edges for both directions. Finally, we see that we obtain different results and corresponding insights, based on which variation of the model is applied for a specific study. Naturally, the model does not in itself suggest a best way to apply it, and instead one would need to apply the variations case-by-case to best glean the relevant information.

3.3.3 A Bitcoin Subgraph

Our final case study is a subgraph of the Bitcoin wallet address network derived from Bitcoin transaction logs. Bitcoin is a cryptocurrency [16], and transactions (buy and sell) among users of this currency are stored and publicly available in a distributed ledger called blockchain. User identities are unknown, but each user has one or many wallet addresses, that are identifiers in every transaction. Then

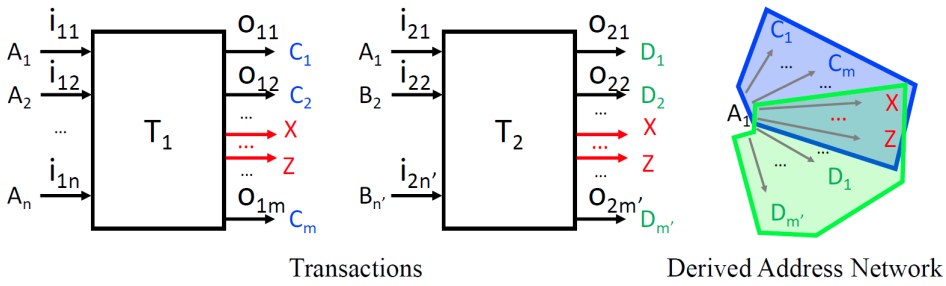


FIGURE 3.8: Two multi-input multi-output Bitcoin transactions and the address network derived for one address from these transactions.

one transaction record amalgamates the wallet addresses of possibly several payers and payment receivers, together with the transaction amounts.

To be more precise, consider an example of two Bitcoin transactions T_1 and T_2, as shown in Figure 3.8. The transaction T_1 has n inputs, from wallet addresses $A_1, \ldots, A_n$, of amounts $i_{11}, \ldots, i_{1n}$ respectively. The outputs, of amounts $o_{11}, \ldots, o_{1m}$ go to wallet addresses $C_1, \ldots, C_m$ respectively. The sum of inputs equals the sum of outputs and any transaction fees (say τ_1), i.e., $|T_1| = \sum_{k=1}^{n} i_{1k} = \tau_1 + \sum_{l=1}^{m} o_{1l}$. For the sake of simplicity, and without loss of generality, we will ignore the transaction fees (i.e., consider $\tau_1 = 0$). A similar setting holds for transaction T_2, where the same wallet address A_1 appears again as part of the inputs, together with some wallet addresses $B_2, \ldots, B_{n'}$ which may or not intersect with $A_2, \ldots, A_n$. By design, Bitcoin transactions do not retain an association as to which specific inputs within a transaction are used to create specific outputs.

Suppose one would like to create a derived address network from some extract of the Bitcoin logs of transactions, that is a graph whose nodes are Bitcoin wallet addresses, and edges are directed and weighted. There should be an edge from address u to address v if there is at least one transaction where some amount of Bitcoin is going from u to v. However as explained above, it is not always possible to disambiguate the input-output pairs. If the input amounts are particularly mutually distinct, and so are the output amounts, and there are input-output amounts that match closely, one might be able to make reasonable guess about matching a specific input to a specific output. In general, in absence of such particular information, one heuristic is to model the input-output association probabilistically. A common heuristic [119] is to consider that based on transaction T_1 there is an edge from A_1 to each of $C_1, \ldots, C_m$. This is shown as the upper subset of edges (in the blue region) in the derived address network shown in Figure 3.8. The same

holds for transaction T_2. Thus in the derived address network, there will be an edge from A_1 to each of the $C_1, \ldots, C_m, D_1, \ldots, D_{m'}$. If some outputs $X, \ldots, Z$ are in common to both transaction outputs, there is a single edge between A_1 and each of the addresses $X, \ldots, Z$.

The derived address network gives us the graph to be analyzed, whose nodes are wallet addresses and edges are built as above. Originally, a given wallet address is sending Bitcoins to possibly different output wallet addresses within one transaction, and the same wallet address may be involved in different transactions, with possible re-occurrences of the same output addresses (this is the case of A_1 which is an input to both transactions T_1 and T_2 and $X, \ldots, Z$ appear as output transactions in both). In the split-and-transfer flow model, we can incorporate this information into the derived address network by assigning the probabilities $q(\{C_1, ..., C_m\}) = \frac{i_{11}}{i_{11}+i_{21}}$ and $q(\{D_1, ..., D_{m'}\}) = \frac{i_{21}}{i_{21}+i_{22}}$ with which the respective subsets $\{C_1, ..., C_m\}$ and $\{D_1, ..., D_{m'}\}$ of the set $\{C_1, ..., C_m, D_1, ..., D_{m'}\}$ of neighbours of A_1 are used. Other choices for $q(x)$ are possible, the rationale for this specific choice is to use a probability that is proportional to the amount of Bitcoin injected by A_1 in each of the transactions.

Edge weights in the derived address network are computed as follows. Let $|T_1| = \sum_{k=1}^{m} o_{1k}$ and $|T_2| = \sum_{k=1}^{m'} o_{2k}$ denote the total amounts involved in each of the transactions. For an edge between A_1 and C_l, which happens in T_1, it is $\omega_{C_1,...,C_m}(A_1, C_l) = \frac{o_{1l}}{|T_1|}$, while for an edge between A_1 and D_l, which happens in T_2, it is $\omega_{D_1,...,D_{m'}}(A_1, D_l) = \frac{o_{2l}}{|T_2|}$. We thus have

$$\sum_{l=1}^{m} \omega_{C_1,...,C_m}(A_1, C_l) = \sum_{l=1}^{m} \frac{o_{1l}}{|T_1|} = 1, \quad \sum_{l=1}^{m'} \omega_{D_1,...,D_{m'}}(A_1, D_l) = \sum_{l=1}^{m'} \frac{o_{2l}}{|T_2|} = 1.$$

If some node pairs, and thus edges, repeat across transactions (for example, A_1 to $X, \ldots, Z$ in our example), these edge weights should cumulate in the overall derived address network. This is captured by the formula (3.2) which is here instantiated as

$$\begin{aligned} f_{A_1,X} &= q(\{C_1, \ldots, C_m\})\omega_{C_1,...,C_m}(A_1, X) + q(\{D_1, \ldots, D_{m'}\})\omega_{D_1,...,D_{m'}}(A_1, X) \\ &= \frac{i_{11}}{i_{11}+i_{21}}\frac{o_{1x}}{|T_1|} + \frac{i_{21}}{i_{11}+i_{21}}\frac{o_{2x}}{|T_2|} \end{aligned}$$

where in transaction T_1 the output to address X is o_{1x}, while it is o_{2x} in transaction T_2.

address	C_H	f_u	out
3CD1QW6fjgTwKq3Pj97nty28WZAVkziNom	8.663332385	0.047377226	2807
38PjB1ghFrD9UQs7HV5n15Wt1i3mZP8Wke	5.721441871	0.19610282	382
3Eab4nDg6WJ5WR1uvWQirtMzWaA34RQk9s	5.433971682	0.177882439	568
3MYqQJ5LbDe9U3drsaDprKxWobVZA3UgAw	5.331666192	0.927034097	2
38mMQxz4knqfmecjLW3atdygfWxvvnJfg7	5.331666192	0.926888565	2
33XZf8Ys9sbqnAKynA4yBckyzwN3SEZaU7	5.331666192	0.925407445	2
3P4C7jpF1oxHgxqt4VgMRcCBEV3YEpaDUm	5.331666192	0.92248333	2
3Fp5ejYY8FsJ6Y3kb377VRjJunTeUVYsuq	5.331666192	0.896621459	2
3Q9SPyCN95szQUoQYgAHKgdhC3YnRsrFrW	5.331666192	0.892827473	2
38A6nGSMR59WHVnj9gaJ2Cm62y9kFE318i	5.331666192	0.890820737	2
3Ce7jUQn2RH5Ysdb4VvShoYymZLpkcqaAA	5.331666192	0.88775841	2
364qbSJFhwkBgZnMuhmUHdczpaZNS2PmE6	5.331666192	0.883268612	2
3KDgKr3qov4Ws5WPnaA2RHjcE1N2UeVYs3	5.331666192	0.861964026	2
1NxaBCFQwejSZbQfWcYNwgqML5wWoE3rK4	5.331666192	0.117536609	2

TABLE 3.8: Nodes with highest entropic centrality.

In a departure from previous works that derive the address network in a manner explained above [119], our graph model is able to retain the information that subsets of edges co-occur, or not, as displayed above. For that reason, the Bitcoin address network is a natural candidate (and in fact, part of the inspiration) for the general flow model with arbitrary splits and transfers as considered in this section, where individual flows may go through a subset of outgoing edges.

For our experiments, we choose a known Bitcoin subgraph, studied in an (unrelated) recent investigation of wallet addresses involved in extorting victims of Ashley-Madison data breach [47] (see [22] for accessing the data used). In that work, we had identified a shortlist of suspect nodes, and studied subgraphs of various radii centred around a given node. We choose one such subgraph comprising 4571 vertices and 4762 edges, obtained by extracting a subgraph of radius 4 (if the graph were undirected) around the wallet address *1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*. Figure 3.9 shows this Bitcoin subgraph, but includes only the nodes which have non-zero entropic centrality score, and only the edges connecting these nodes. Nodes can have zero centrality score because they may have no outgoing edges, and this may further happen because we consider only a subgraph of a given radius around a specific address which in turn imposes a truncated boundary and ignores outgoing edges for the nodes at the 'periphery' of this subgraph. Tables 3.8 and 3.9 show the nodes with the highest (unscaled) entropic centrality, for the address network graph, and for the graph obtained by reversing the directionality of edges in the address network graph, respectively. The highest entropic centrality

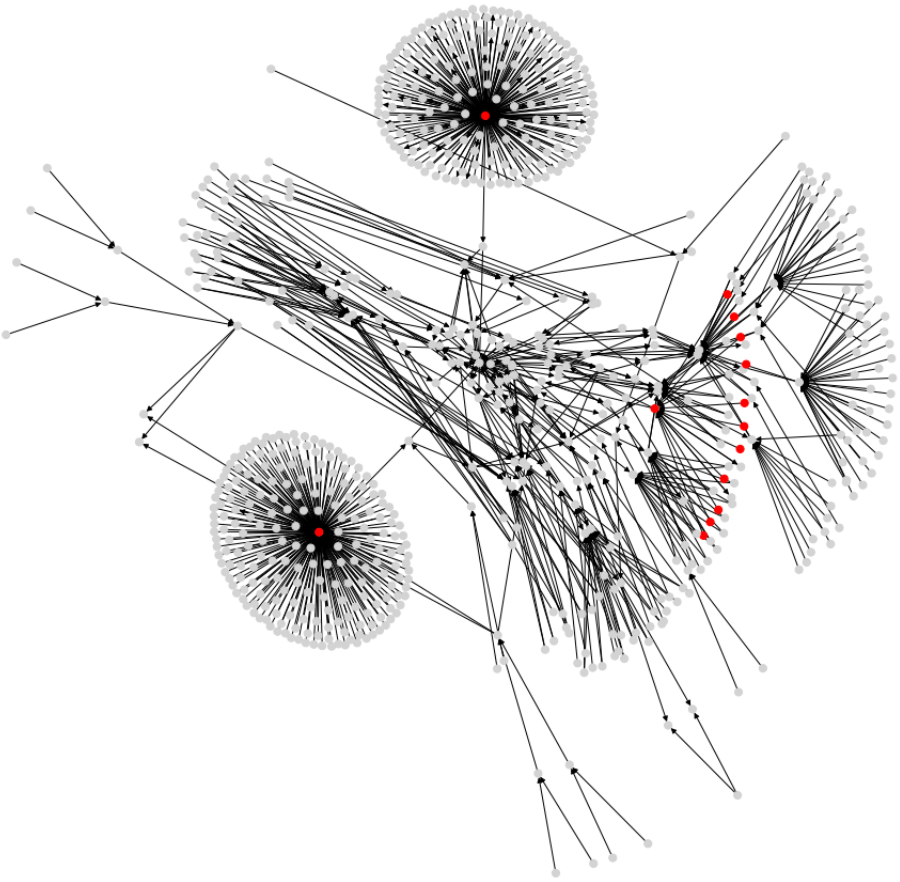


FIGURE 3.9: A subgraph of the Bitcoin subgraph, which comprises only the nodes that have non-zero entropic centrality. The nodes in red are those listed in Table 3.8, with the highest entropic centrality.

nodes from the address network graph are indicated as red nodes in Figure 3.9. Three nodes have very high out-degrees (2807, 382, 568) and they appear at the top of the list for entropic centrality also. They can also be seen to be at the center of hubs on Figure 3.9. However the other identified nodes with high entropic centrality have only degree 2. This is likely because they inherit the influence of one of the highly central nodes, as can be guessed from Figure 3.9. The column f_u in Tables 3.8 and 3.9 contains the normalized input transaction amounts, for example, for f_{A_1}, we have that the input amount is $i_{11} + i_{21}$, and the normalization is done by dividing by the sum of all the input amounts. We observe if $f_u C_H(u)$ is used instead of $C_H(u)$, then the ranking of the first two most central nodes in Table 3.8 would be swapped, suggesting that while one has the largest spread, the other actually deals with more Bitcoin.

While we would like to emphasize that the current discussion is decoupled

address	C_H	f_u	in
38PjB1ghFrD9UQs7HV5n15Wt1i3mZP8Wke	7.647798247	171.3596917	218
3Eab4nDg6WJ5WR1uvWQirtMzWaA34RQk9s	7.558346728	175.0224087	196
15hWpb3m5VXdn9KVsS4rSMnrQQJLhXVyN4	5.264957659	7.504156172	17
1C7PDYzjRDqomyywDHEqx9huYoYQoGYgdV	4.987628604	3.949210381	31
1zksVRSDUuX2E5mMNvvbA9C4esfnvVdfA	4.417642538	0.494446692	2

TABLE 3.9: Nodes with highest centrality when edges have reverse directionality.

from the previous study, and we use here this Bitcoin graph to explore the entropic centrality model, it may still be worth mentioning that one of those suspect nodes (*15hWpb3m5VXdn9KVsS4rSMnrQQJLhXVyN4*) from previous study [47] has high enough entropic centrality to be listed (see Table 3.9) as a top entropic centrality node in this work. Thus, the entropic centrality analysis can be used as a tool to identify nodes of interest, and to create a shortlist of nodes to be investigated further in detail, in the context of Bitcoin forensics.

Tables 3.10 and 3.11 compare the entropic centrality $C_{H,p}$ with other centralities. With respect to scaled entropic centrality, there is a large variation in the weightages associated with the edges, which has a significant impact on the relative rankings between scaled/unscaled entropic centralities. With respect to weighted betweenness, only three addresses are relevant, they are, with their respective in- and out-degree, *3Eab4nDg6WJ5WR1uvWQirtMzWaA34RQk9s* (ranked 1, in-degree: 196, out-degree: 568), *38PjB1ghFrD9UQs7HV5n15Wt1i3mZP8Wke* (ranked 2, in-degree: 218, out-degree: 382), and *3CD1QW6fjgTwKq3Pj97nty2-8WZAVkziNom* (in-degree: 14, out-degree: 2807). The other addresses are ranked 69 (corresponding to a betweenness of 0). The graph has as largest eigenvalue $\lambda_1 \simeq 7.1644140$ and $\frac{1}{\lambda_1} \simeq 0.139578$. As with the previous cases, alpha and Katz centralities are very close to each other, they also agree more closely with $C_{H,p}$ on the most central addresses, but Table 3.11 shows that this is not the case in general. The trends shown by the Kendall rank correlation coefficient is similar to previous cases: there are more dissimilarities between PageRank and entropic centralities than between alpha/Katz and entropic centralities, but overall, entropic centralities give a different view point, as is expected by extrapolating Borgatti's view point.

TABLE 3.10: The top 14 addresses with respect to $C_{H,p}$ in the first column and their respective relative ranks as per other centralities.

wallet address ($C_{H,p}(u)$)	$f_u C_H$	alpha/Katz ($\alpha = 0.1$)	PageRank (UW)	PageRank (W)
3CD1QW6fjgTwKq3Pj97nty28WZAVkziNom (1)	652	1	1	1
38PjB1ghFrD9UQs7HV5n15Wt1i3mZP8Wke (2)	481	3	14	14
3Eab4nDg6WJ5WR1uvWQirtMzWaA34RQk9s (3)	514	2	13	15
3MYqQJ5LbDe9U3drsaDprKxWobVZA3UgAw (4)	609	4	2	455
38mMQxz4knqfmecjLW3atdygfWxvvnJfg7 (4)	3	4	2	3
33XZf8Ys9sbqnAKynA4yBckyzwN3SEZaU7 (4)	9	4	2	10
3P4C7jpF1oxHgxqt4VgMRcCBEV3YEpaDUm (4)	7	4	2	8
3Fp5ejYY8FsJ6Y3kb377VRjJunTeUVYsuq (4)	2	4	2	3
3Q9SPyCN95szQUoQYgAHKgdhC3YnRsrFrW (4)	8	4	2	8
38A6nGSMR59WHVnj9gaJ2Cm62y9kFE318i (4)	5	4	2	6
3Ce7jUQn2RH5Ysdb4VvShoYymZLpkcqaAA (4)	10	4	2	11
364qbSJFhwkBgZnMuhmUHdczpaZNS2PmE6 (4)	1	4	2	2
3KDgKr3qov4Ws5WPnaA2RHjcE1N2UeVYs3 (4)	4	4	2	5
1NxaBCFQwejSZbQfWcYNwgqML5wWoE3rK4 (4)	6	4	2	7

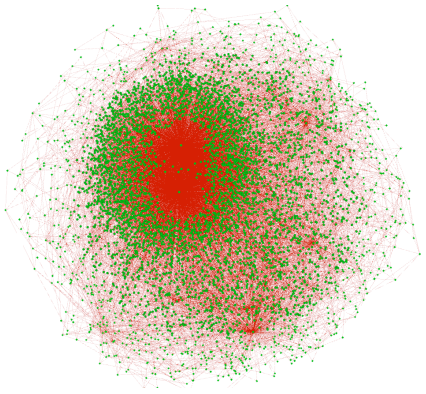
TABLE 3.11: Kendall rank correlation coefficient τ_κ across the centrality algorithms for the Bitcoin subgraph dataset (excluding 3926 nodes which all had an entropic centrality score of zero).

	$C_{H,p}$	$f_u C_{H,p}$	alpha ($\alpha = 0.1$)	PageRank (UW)	PageRank (W)	Katz ($\alpha = 0.1$)
$C_{H,p}$	0	0.233172	0.235947	0.342135	0.395660574	0.234265
$f_u C_{H,p}$	0.233172	0	0.213537	0.315283	0.43371	0.211848
alpha	0.235947	0.213537	0	0.265286	0.49497	0.003209
PageRank(UW)	0.342135	0.315283	0.265286	0	0.429998	0.265253
PageRank(W)	0.395660	0.43371	0.49497	0.429998	0	0.496515
Katz	0.234265	0.211848	0.003209	0.265253	0.496515	0

3.3.3.1 The Uniform Case

Consider a simplified network model, where we only consider the connections among users[4], where we deem two distinct users u and v to be connected by a directed edge $(u, v) \in E$ if there exists a transaction where a previous assignment of Bitcoins to u is used as an input to the transaction, and v is assigned some Bitcoins as an output of the transaction. So to say, if there are several such inputs between the same users, they are aggregated and shown only once. In our simplified model, we discard the actual amount aggregated over the transactions, and focus on the existence of edges, to reflect connectivity. For each user, we also assign a self-loop in the graph to indicate the money that is actually not spent - this could either be because the user sent some change back to itself, creating a Bitcoin transaction output, or just because the money was indeed never spent at all (so there is no transaction to itself in the actual list of Bitcoin transactions).

[4]For this section, we abuse the terminology and refer the wallet addresses interchangeably as users, even though these are different in practice, and a single user can use multiple wallet addresses.



C	$\tau(C, C_H)$
C_B	0.5185
C_C	0.3794
$C_{D,in}$	0.1709
$C_{D,out}$	0.0310

FIGURE 3.10: A component of the Bitcoin network with 5206 vertices. The table shows the Kendall-τ correlation coefficient [4] between the different centrality measures C and the entropic centrality C_H.

Note that, for large networks, computing the entropic centrality can be costly. Thus, when probabilities become very small, it is meaningful to apply a threshold and force the search for paths to stop, and consider that nodes from then on are reached with a probability 0. We used a threshold of 10^{-6} in our experiments reported next.

We took the Bitcoin transaction history for the month of December 2016, and derived the transaction graph among wallet addresses in a manner described above. From the graph obtained, we chose two connected components comprising 5206 (Fig 3.10 and see [23] for accessing data used) and 5251 (Fig 3.12 and see [24] for accessing data used) vertices to study. We chose these graphs because they are of similar size, a size which is computationally tractable with our current implementation, and also because they have very different structures, as apparent from the figures, where we also report the Kendall-τ coefficient [4] for the ranking of vertices determined by the entropy based centrality with respect to the rankings derived from other (above mentioned) centrality measures. Kendall-τ coefficient is a measure of rank correlation. Higher the measure, the higher the pairwise mismatch of ranks across two lists. A non-negligible value of this distance indicates that the rankings obtained from entropy centrality provide distinct information, with respect to the other centrality measures.

Not surprisingly, the out-degree based centrality has the least mismatch, because typically, a large out-degree would help a vertex to spread the flow widely, leading to a high entropy centrality. Yet, it is not always true, since a vertex

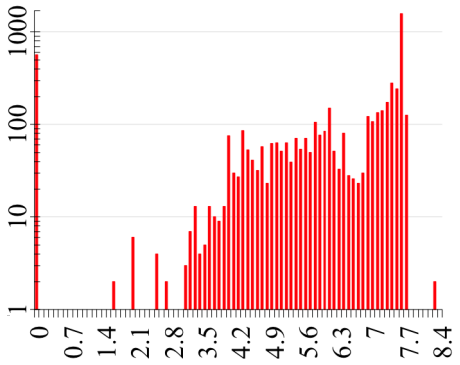


FIGURE 3.11: Entropy centrality frequency (network with 5206 vertices).

with small out-degree connected to very few or even one vertex with high centrality would gain high centrality by transitivity. We observe this in our network with 5026 nodes, where the top-5 entropy centralities are approximately 8.26, 8.23, 8.12, 7.68 and 7.62 and belong to vertices with out-degrees 1495, 1494, 12, 265 and 3 respectively. Notice both the fact that the out-degrees are not in a strict order, and hugely different for the vertices even with close entropy centrality. We provide a frequency (log-scaled y-axis) histogram of the entropy centrality (x-axis) in Fig 3.11. There were 1586 vertices with a entropy centrality approximately 7.6, so we had to use log-scale, and values that occurred only once are not visible and are thus indistinguishable from those which did not occur at all. In this graph, essentially most nodes are well connected to a hub with high spread (there are two such nodes with out-degrees of 1495 and 1494), and thus most of them have pretty high entropy centrality - while 566 leaf vertices have zero entropy centrality. In contrast, for the graph with 5251 nodes shown in Fig 3.12 (we embed the entropy centrality (bins of size 0.1) frequency of vertices in the figure itself) we see that there are very few vertices with high centrality values (only six of them have centrality of more than 6.6), most vertices have moderate centrality values (they are close to the leaves in the directed graph) and finally there is a large number of leaves which receive money from several users, without transferring anything forward. In fact, in this graph, six vertices had out-degrees between 429 and 898, and only four more vertices had out-degree more than 1. Every other vertex had out-degree of one or zero. Unsurprisingly, $\tau(C_{D,out}, C_H)$ is very low for this graph, and the out-degree versus entropy centrality based importance of nodes predominantly does not vary.

$C_H(v)$	frequency
8.4	1
7	1
6.7	4
2.6	1
2.2	4781
2.1	3
1.5	1
1	1
0	458

C	$\tau(C, C_H)$
C_B	0.3236
C_C	0.4039
$C_{D,in}$	0.2606
$C_{D,out}$	0.0010

FIGURE 3.12: A component of the Bitcoin network with 5251 vertices. The tables show the entropy centrality stats, and the Kendall-τ correlation coefficient between the different centrality measures C and the entropic centrality C_H.

3.4 Summary

In summary, the experiments in general and the two variants of Bitcoin network particular illustrate several salient aspects of our entropic centrality model. First, regardless of whether we consider weighted or unweighted networks, we see that a node with low-degree can gain high centrality if it connects to a hub with high spread - a transitive aspect of influence or spread not captured in most centrality measures, which however may be relevant in certain contexts. Second, in the case study of the Bitcoin network which reflects the weights and co-occurrence of specific subset of edges, our entropic centrality score accounts for the actual amount of Bitcoin and hitches centrality accordingly to emphasize on nodes which distribute significant amount of Bitcoins over a portion of the network. In the particular context of Bitcoin analysis, this can be used to identify shortlist of nodes or hubs of interest for further investigations (for example, using techniques we present later in Chapter 6).

In order to compute entropic centrality in a large scale network, for very long paths, one can apply a threshold and force the search for paths to stop when probabilities become very small and consider that nodes from then on are reached with a probability 0. Doing so, we can still get a close estimate of the entropic centrality while speeding up overall computations.

The current model of entropic centrality computation based on paths without cycles has equivalent computational cost as the original work we extend [18]. In [5], it has been argued that relaxing the 'no cycles' constraint, and studying

random walks without constraint (and thus, accepting possibility of cycles) helps reduce the computational costs, since the process can be modeled as a Markov process. Accordingly, in the next chapter we first study entropic centrality with this alternative approach, but again generalizing the ideas so that the model can capture weighted and directed graphs. We then design a clustering algorithm that leverages on the probability distributions induced by the random walkers.

Chapter 4

Centrality and Clustering for Weighted Directed Graphs

In the previous chapter, we discussed how entropic centrality based on flows in the network can capture the characteristics of a wide range of real world systems including airport network, shareholding network and Bitcoin network. With the cycle-free path based restriction of the flows, entropic centrality computation can be complicated. A constraint free random walk approach proposed in [5] for undirected graphs was shown to simplify the computations, since a Markov model can be applied for capturing the process. Furthermore, a link deletion based iterative approach built over the entropic centrality measure was used in [5] as a means to realize clustering. The work presented in this chapter builds upon these ideas.

We consider the generalized setting of directed weighted graphs, such that any undirected graph and/or unweighted graph can be treated as special cases. We adhere to the definition of entropic centrality, which treats a node as central if there is a high uncertainty about the destination of a random walker starting its walk at the given node, where however a suitable probability to choose a given edge by a random walker is introduced in order to capture the weights of outgoing edges, with the further possibility to also assign an intrinsic initial weight (importance) to nodes based on its weight cumulated out-degree. We analyze the behaviour of random walks, particularly investigating the probability distribution that a random walker stops at any given node as a function of time, i.e., the number of hops of random walk.

As noted in our literature review (see Section 2.1.2), in the context of flow

network clustering, InfoMap [7, 42] uses PageRank [56] matrix to determine network partitions by minimizing average description length. Community detection based on personalized PageRank diffusion [9, 89] utilizes PageRank vector and a random walk. Clustering based on label propagation [87, 88] labels nodes based on the label of the majority of their neighbors. These are best suited for scenarios that can be viewed as parallel duplication.

In the studies [38, 162], local clusters are detected using multi-agent random walks, using the intuition that multiple walkers are more likely to stay confined within local communities is such structures exist in a network, than escape the local communities simultaneously. These studies inspire us to explore whether and how the local communities are inferred from the gradient observed in the evolution of the entropic centralities but with the use of a single random walker, by initiating the random walks at nodes with low entropic centrality since nodes at the interface of multiple local communities are expected to have higher centrality.

We design a 2-stage base clustering algorithm that leverages entropic centrality to detect local communities, and then perform a bottom-up iterative agglomeration. Based on our experiments with networks with ground truth, we noticed that this base mechanism may lead to clusters which may be perceived as an amalgamation of several actual clusters. Reapplication of the base technique over such subgraphs led to significantly 'better' fit with ground truth, and the quality of results so obtained were comparable with other prominent community detection algorithms (we experimented with InfoMap [7], label propagation and Louvain clustering [8]) demonstrating the strength of the fundamental approach, but also exposing some limitations of the current realization of the approach. Determining automatically whether and how often subgraphs may need to be reclustered is something we defer for future study, and would need to be addressed to better meet practical requirements.

To put the last issue in proper context, we will like to emphasize that many other clustering approaches in the literature are also known to exhibit similar behaviour, and the issue is not singular to our approach, nor is it a fundamental bottleneck. For instance, quoting from Neo4j [163]'s documentation of the Louvain algorithm[1] "For larger networks, the Louvain method doesn't stop with the "intuitive" communities. Instead, theres a second pass through the community

[1] `https://neo4j.com/docs/graph-algorithms/current/algorithms/louvain/` accessed on 14 June 2019

modification and coarse-graining stages, in which several of the intuitive communities are merged together. This is a general problem with modularity optimization algorithms; they have trouble detecting small communities in large networks."

Overall, we have thus designed and demonstrated the efficacy of a new flexible entropic centrality guided approach for graph clustering, applicable to weighted and directed graphs.

4.1 System Model

As explained in the previous chapter, the notion of entropic centrality $C_H(u)$ for a node u in an undirected graph was originally defined in [18] as

$$C_H(u) = -\sum_{v \in V} q_{uv} \log_2(q_{uv}) \tag{4.1}$$

where q_{uv} denotes the probability of reaching v from u following any path. This formalizes the idea that a node is central if there is a high uncertainty about the destination of a random walker starting at it, assuming that at each hop, the random walker is equally likely to continue to any unvisited node, or to stop there. This is because the Shannon entropy $H(X) = -\sum_x p(X = x) \log_2 p(X = x)$ of a random variable X measures the amount of uncertainty contained in X.

Consider now a connected directed graph $G = (V, E)$ with $n = |V|$ nodes labelled from 1 to n and $|E|$ directed edges, and a random walker on G, which starts a random walk at any given node according to an initial probability distribution. Note that in assuming a random walker, we relax the assumption (in [18]) that only paths are traces, and the walker cannot revisit previously traversed nodes. This is in lines with [5], which demonstrated for the case of unweighted undirected graphs, that, this relaxation simplifies the analysis while still giving meaningful entropic centrality measure, since such an unconstrained random walk can be modeled as a Markov process. In essence, we extend and explore the model to encapsulate directed and weighted graphs.

4.2 The Notion of Entropic Centrality for Unweighted Graphs

A *right stochastic matrix* P is a square matrix, where every coefficient is a non-negative real number and each row sums to 1. A right $n \times n$ stochastic matrix describes a Markov chain X_t, $t \geq 1$, over n states. The coefficient $P_{u,v}$ of P is the probability of moving from state u to state v, and since the total $\sum_{u=1}^{n} P_{u,v}$ of transition probabilities from state u to all other states must be 1 by definition of probability, the matrix P is indeed a right stochastic matrix, called a transition matrix. To show that P^k is again a right stochastic matrix for any positive integer k, take the all 1 column vector $\mathbf{1}$ and notice that $P\mathbf{1} = \mathbf{1}$ for a right stochastic matrix. Then $P^k\mathbf{1} = P\mathbf{1} = \mathbf{1}$. Let $(P_{u,l})_l$ be the uth row of P, and $(P_{l,v})_l$ be its vth column. Their product $(P_{u,l})_l(P_{l,v})_l = \sum_l (P_{u,l})_l(P_{l,v})_l$ determines the probability of going from the state i to the state j in two steps, where the intermediate state is l, for any possible state l. In general, P^k gives the probability transition of going from any state to any another state in k steps.

In unweighted directed graph $G = (V, E)$, if the walker decides to which node v to walk to, based only on the current node u at which it currently is, the random walk is a Markov chain, which can be modelled by a stochastic matrix P where P_{uv} is the probability to go to node v from node u. Let $d_{out}(u)$ denote the out-degree of u. A typical choice that we consider in this section is $P_{uv} = \frac{1}{d_{out}(u)}$ for every node v belonging to the set $\mathcal{N}_u$ of out-neighbours of u (this is saying that each neighbour is chosen uniformly at random). Other choices are possible, and indeed, subsequently in this chapter, we shall consider the case of weighted graphs where the transition probabilities depend on the edge weights.

4.2.1 A Markov Entropic Centrality

To define a notion of entropic centrality similar to (4.1), the random walker actually needs to have the choice of stopping at any given node. This is not well captured by the transition matrix P described above, even if there is a self-loop at every node. Thus for any node v in G, an auxiliary node v' is added [5], together with a single directed edge (v, v'), and a probability $p_{vv'}$ to be chosen (the original probabilities are adjusted so that the overall matrix remains stochastic). Once the flow reaches an auxiliary node, it is absorbed. This gives a notion of Markov entropic centrality as defined in [5].

Definition 4.1. The *Markov entropic centrality* of a node u at time t is defined to be

$$C_H^t(u) = -\sum_{v \in V} (p_{uv}^{(t)} + p_{uv'}^{(t)}) \log_2(p_{uv}^{(t)} + p_{uv'}^{(t)}) \tag{4.2}$$

where $p_{uw}^{(t)}$ is the probability to reach w at time t from u, for w a node in V or an auxiliary node.

A node u is central if $C_H^{(t)}(u)$ is high: if $C_H^{(t)}(u)$ is high, using the underlying entropy interpretation, this means that for a random walker starting at node u, the uncertainty about its destination v after t steps is high, thus u is well connected.

The time parameter t can also be interpreted as a notion of locality. It describes a diameter around the node u, of length t steps. Thus the entropy centrality at $t = 1$ emphasizes a node's degree, t being the graph diameter implies that we are considering a period of time by when the whole graph can be first reached, and $t \to \infty$ describes the asymptotic behaviour over time. In other words, $C_H^{(t)}(u)$ can be regarded as a measure of influence of u over its close neighbourhood for small values of t, and over the whole graph for t asymptotic.

Given an $n \times n$ stochastic matrix P describing the moves of a random walker on a directed graph where every node has a self-loop, let us then introduce $n = |V|$ auxiliary nodes, one for each node v, $v \in V$, with corresponding probabilities $p_{vv'} = D_{vv}$ to walk from node v to node v'. This creates the following right stochastic matrix

$$P_{new} = \begin{bmatrix} \tilde{P} & D \\ \mathbf{0}_n & \mathbf{I}_n \end{bmatrix}$$

where $D = D_{uu}$ is a diagonal matrix, and $\tilde{P}$ is such that $[\tilde{P}, D]\mathbf{1} = \mathbf{1}$. The identity matrix $\mathbf{I}_n$ represents the absorption of the flow at the auxiliary nodes. To determine the centrality of a specific node u, we assume an initial distribution that gives a probability of 1 to start at u, and 0 elsewhere.

The definition of Markov entropic centrality was used as part of a clustering algorithm in [5], and the probabilities $p_{vv'}$ were explored numerically to optimize the clustering results. Our first contribution is the following closed-form expression for the asymptotic behaviour of the transition matrix.

Lemma 4.1. *For an integer $k \geq 1$ and $D \neq \mathbf{0}$, we have*

$$P_{new}^k = \begin{bmatrix} \tilde{P}^k & (\sum_{j=0}^{k-1} \tilde{P}^j)D \\ \mathbf{0}_n & \mathbf{I}_n \end{bmatrix}. \tag{4.3}$$

In particular

$$P_{new}^{k} = \begin{bmatrix} \mathbf{0}_n & (\mathbf{I}_n - \tilde{P})^{-1}D \\ \mathbf{0}_n & \mathbf{I}_n \end{bmatrix} \tag{4.4}$$

when $k \to \infty$.

Proof. Formula (5.1) follows from an immediate computation. Since $\tilde{P}, D$ have non-negative real coefficients, so has $\tilde{P}^j D$, thus $(\sum_{j=0}^{l} \tilde{P}^j)D \geq (\sum_{j=0}^{l-1} \tilde{P}^j)D$ for any $l \geq 1$, and equality holds if and only if $\tilde{P}^N = 0$ for some N. But this N must exist when $k \to \infty$, because P_{new} is a stochastic matrix, meaning that the sum of every row must remain 1, while the coefficients of $(\sum_{j=0}^{k-1} \tilde{P}^j)D$ increases at every increment of $k \leq N$. Then

$$P_{new}^{N} = \begin{bmatrix} \mathbf{0}_n & (\sum_{j=0}^{N-1} \tilde{P}^j)D \\ \mathbf{0}_n & \mathbf{I}_n \end{bmatrix}$$

but since $(\tilde{P} - \mathbf{I}_n)(\sum_{j=0}^{N-1} \tilde{P}^j) = \tilde{P}^N - \mathbf{I}_n = -\mathbf{I}_n$, the matrix $(\tilde{P} - \mathbf{I}_n)$ is invertible, $\sum_{j=0}^{N-1} \tilde{P}^j = -(\tilde{P} - \mathbf{I}_n)^{-1}$, yielding (4.4). □

The probabilities $p_{uv}^{(t)} + p_{uv'}^{(t)}$ in (4.2) are obtained from the uth row of the matrix P_{new}^{t}, by summing the coefficients in the columns v and v'. When $t \to \infty$, the term in column j becomes 0, and we are left with the term in column j'. Since every node u has a self-loop, the out-degree $d_{out}(u)$ counts the self-loop (however the auxiliary node is not counted in $d_{out}(u)$, node and graph properties refer to the original graph).

4.3 The Notion of Entropic Centrality for Weighted Graphs

Consider now a weighted directed graph $G_w = (V, E)$, where the weight function $w : E \to \mathbb{R}_{\geq 0}$ attaches a non-negative weight $w(u, v)$ to every edge $(u, v) \in E$. For a node $u \in V$, let $\mathcal{N}_u = \{v \in V,\ (u, v) \in E\}$ be the set of out-neighbours of u, $d_{out}(u) = |\mathcal{N}_u|$ be the out-degree of u, and $d_{w,out}(u) = \sum_{v \in \mathcal{N}_u} w(u, v)$ be the weighted out-degree of u.

A natural way to define a transition matrix P_w to describe a random walk over G_w taking into account the weight function w would be to set $P_{w,uv} = \frac{w(u,w)}{d_{w,out}(u)}$. Now at $t = 1$, a node u_1 whose out-degree is 2 with outgoing edges of weight 1 is considered less important than a node u_2 with out-degree 6 and all 6 outgoing edges

of weight 1, because u_1 contributes a probability $\frac{1}{2}$ instead of $\frac{1}{6}$ for u_2, therefore reducing the uncertainty from $\log_2 6 \approx 2.585$ to $\log_2 2 = 1$. But if a node u_3 has two outgoing edges, one with weight 2 and one with weight 3, then its entropic centrality is $-\frac{2}{5}\log_2 \frac{2}{5} - \frac{3}{5}\log_2 \frac{3}{5} \approx 0.97 < 1$. Thus the node u_3 with weighted outgoing edges has lower entropic centrality than the node u_1 with unweighted outgoing edges. This is meaningful from an entropy view point, since it captures uncertainty, and the uniform distribution over the outgoing edges gives the most uncertainty about the random walk. However this may not be desired if the given scenario requires nodes with high weighted out-degree to be more central than others, e.g., when using the centrality measure as a proxy for determining influence.

4.3.1 A Weighted Markov Entropic Centrality

In order to capture the weights of the outgoing edges in the current framework of Markov entropic centrality, we introduce two tuning parameters, a conversion function $\alpha : w(e) \to \alpha(w(e))$ and a node weight function $\mu : v \to \mu(v)$.

The conversion function $\alpha : w(e) \to \alpha(w(e))$ adjusts the transition matrix $P_{\alpha(w)}$ such that $P_{\alpha(w),uv} = \frac{\alpha(w(u,w))}{d_{\alpha(w),out}(u)}$ (with $d_{\alpha(w),out}(u) = \sum_{v \in \mathcal{N}_u} \alpha(w(u,v))$) depending on the importance that weights are supposed to play compared to edges. The two obvious choices for α are $\alpha(w(e)) = 1$ for all edges e, which treats the weighted graph as unweighted, and $\alpha(w(e)) = w(e)$ which considers the graph with its given weights, more generally one could define $\alpha(w(e)) = w(e)^\beta$ for some parameter β. This formulation has the flexibility to give more or less importance to weights with respect to edges, but for the up-coming analyses, it is enough to consider the case where weighted edges are more important than unweighted ones. We normalize the weights with respect to the lowest non-trivial weight, to indicate the relative importance of edges.

The conversion function α allows the weights to play a role in the transition matrix $P_{\alpha(w)}$, based on the scenario needs, but it does not address the issue of defining entropic centrality for weighted graphs. To do so, we use the node weight function $\mu : v \to \mu(v)$ and propose the notion of weighted Markov entropic centrality, which is inspired by the concept of weighted entropy [164].

Definition 4.2. The *weighted Markov entropic centrality* $C^t_{\alpha(w),H}(u)$ of a node u at time t is defined to be

$$-\sum_{v\in V}\mu(v)(p^{(t)}_{\alpha(w),uv}+p^{(t)}_{uv'})\log_2(p^{(t)}_{\alpha(w),uv}+p^{(t)}_{uv'}), \tag{4.5}$$

where $p^{(t)}_{\alpha(w),uv}$ is the probability to reach v at time t from u, for v in V, taking into account the weights $\alpha(w(e))$ for every edge in E. Auxiliary nodes defined for the unweighted case are still present and $p^{(t)}_{uv'}$ is the probability to reach an auxiliary node v' from u at time t, which depends on the choice of the absorption probability matrix D, as in the unweighted case. The probabilities $p^{(t)}_{\alpha(w),uv}+p^{(t)}_{uv'}$ are obtained from the matrix $P^t_{\alpha(w),new}$ (use $P_{\alpha(w)}$ instead of P in Lemma 4.1).

The computation of $C^t_{\alpha(w),H}(u)$ involves a sum over all nodes $v\in V$, including $v=u$, and the probability to go from u to every one v. The weight $\mu(v)$ captures the influence of the reached node v: if we reach influential nodes v of weight $\mu(v)$ in t time, then u should be more central than if the only nodes we reach from it have no influence themselves.

We define $\mu(v)$ to be $\mu(v)=\left(\frac{d_{w,out}(v)}{d_{out}(v)}\right)^\gamma$ if $d_{out}(v)\neq 0$. When $d_{out}(v)=0$, then $d_{w,out}(v)=0$ and we set $\mu(v)=1$. The weighted Markov entropic centrality (4.5) then gives an influence which is proportional to the ratio $\left(\frac{d_{w,out}(v)}{d_{out}(v)}\right)^\gamma$, while γ gives a way to amplify or reduce this influence. If all weights are 1, then the ratio simplifies to 1, and we are back to the non-weighted definition of Markov entropic centrality (as also with $\gamma=0$). Other possible choices for $\mu(v)$ could be explored, such as $\mu_(v)=\frac{\log_2 d_{w,out}(v)}{\log_2 d_{out}(v)}$. Indeed, $\log_2 d_{w,out}(v)$ would be the influence that v would have had over its neighbours in terms of entropic centrality if it had $d_{w,out}(v)$ neighbours of weight 1, while $\log_2 d_{out}(v)$ is the entropic centrality over the neighbours of v, ignoring the weights.

As a consequence of the weight normalization that makes every weight at least 1, the absolute entropic centrality values obtained when using a non-trivial function for $\mu(v)$ will necessarily be at least as much as obtained with $\mu(v)=1$.

4.4 Entropic Centrality Based Clustering

We next explore whether and how the local communities inferred from the gradients observed in the evolution of the entropic centralities as a function of t (see Section 4.2.1) can be exploited to realize effective clustering.

The formulas (4.2) and (4.5) for the (weighted) Markov entropic centralities involve the sum of probabilities $p_{uv}^{(t)} + p_{uv'}^{(t)}$ and $p_{\alpha(w),uv}^{(t)} + p_{\alpha(w),uv'}^{(t)}$ respectively. We define $\hat{P}$ to be the matrix whose uth row contains the coefficient $p_{uv}^{(t)} + p_{uv'}^{(t)}$ in the unweighted case, and $p_{\alpha(w),uv}^{(t)} + p_{\alpha(w),uv'}^{(t)}$ in the weighted case, in column v. Since the algorithm that we describe next uses $\hat{P}$ in the same manner, irrespectively of t and of whether there is a weight function, we keep the same notation $\hat{P}$.

The proposed algorithm works in two stages: first, we create local clusters around 'query nodes', and then, we carry out a hierarchical agglomeration.

In the initial step, we look for a cluster around and inclusive of a specific query node, similarly to [38]. In [38] though, it was not possible a priori to determine suitable query nodes (e.g., a node could be at the boundary of two clusters, and thus yield the union of subsets of these clusters as a resultant cluster), and hence multiple constrained random walkers were deployed to increase the confinement probability of the random walks within a single cluster. In contrast, our choice of query nodes is informed by their Markov entropic centrality: nodes tend to have a relatively high entropic centrality when they either are embedded within a large cluster, or when they are at the boundary of two clusters (and could arguably belong to either). Accordingly, we choose to start identifying local communities by choosing the low entropic centrality nodes as query nodes. Our experiments support this intuition, and the low entropic centrality nodes as query nodes yield meaningful local clusters, which can then be further coalesced into larger (and fewer) clusters, via a bottom-up hierarchical clustering. The clustering's ***first stage*** is given in Algorithms 1 and 2.

In Algorithm 1, we maintain $S_{cluster}$ as a current global view of clusters. We start from the lowest entropic centrality node as a query node, and repeat the process as long as there are nodes that do not already belong to some existing cluster (listed in $S_{cluster}$)[2]. We consider the transition probabilities from a query node v_q to all the other nodes as per $\hat{P}$, and carry out a clustering of these (one-dimensional, scalar) probability values[3]. This identifies an initial set of clusters S_{ini,v_q}. Among these clusters, we choose the cluster with the highest average transition probability from v_q. We define S_{v_q} to be this cluster along with v_q itself since, (i) the comprising nodes have similar probabilities for random walks to end up there when originating

[2]Experiments indicate that as long as the high (from the top 10% to 40%) entropic centrality nodes are not used as query nodes until the end, clusters obtained using either random or lowest entropic query nodes first, are all mutually consistent (high F-score [102], see Subsection 4.5.5).

[3]Experiments use the Python scikit-learn agglomerative clustering [165].

Algorithm 1 Probability distribution based graph clustering

procedure ProbDistClustering($G = (V, E), N, t$)
▷ $N << |V|$ stands for the top-N most central nodes
  Compute $\hat{P}$ and the entropic centrality of all $v \in V$.
  Assign $S_{HE} = \{$the top-N entropic centrality nodes$\}$.
  ▷ Initialization
  Set Q: nodes in ascending order of entropic centrality.
  Set $S_{cluster} = \varnothing$. ▷ Current clusters
  ▷ Global clustering
  while Q is not empty **do**
    Take the query node v_q from Q's head (remove it).
    ▷ Obtain query node centric local cluster
    Apply a(ny) clustering algorithm[2] on the row $(\hat{P}_{v_q,v})_{v \in V}$ of $\hat{P}$ to form the set S_{ini,v_q} of clusters.
    Choose S_{v_q} from S_{ini,v_q} with the highest average transition probability (include v_q). ▷ v_q's raw cluster
    ▷ Prune the raw cluster S_{v_q} using Algorithm 2.
    ProcessRawCluster($S_{v_q}, S_{HE}, S_{cluster}$)
    if S_{v_q} is empty (after ProcessRawCluster) **then**
      Put v_q in the tail of Q.
    else (clustered nodes cannot be query nodes)
      $\forall v \in S_{v_q}$, remove v from Q.
    ▷ Integrate the local result with the global view.
    if S_{v_q} intersects any cluster(s) in $S_{cluster}$ **then**
      Merge them.
    else Add S_{v_q} to $S_{cluster}$.
  return $S_{cluster}$

from v_q (this follows from the clustering of the probability values), and crucially, these nodes are considered to comprise the immediate local community because (ii) this is the largest (in expectation) such probability. Then S_{v_q} is pruned, according to Algorithm 2, as follows. If the query node v_q belongs to a pre-designated group of high entropic centrality nodes S_{HE}, then there is a risk that it may inadvertently merge multiple clusters which one may want to be separate. Accordingly, we check the intersection of S_{v_q} against existing clusters from $S_{cluster}$, and in case there are non-trivial intersections, and yet there is no unique cluster with which there is a largest intersection, we discard S_{v_q} since it otherwise risks merging clusters which ought to be distinct. Otherwise, we retain in S_{v_q} the nodes from the largest intersection, as well as nodes that were in no intersection, but discard the rest. Furthermore, if there are multiple pre-designated high entropic centrality nodes in S_{v_q} (other than v_q), we retain among these only the one to which the transition

Algorithm 2 Pruning of the raw cluster S_{v_q}.

1: **procedure** ProcessRawCluster($S_{v_q}, S_{HE}, S_{cluster}$)
▷ most central node list S_{HE}, current cluster list $S_{cluster}$
2: **if** $v_q \in S_{HE}$ **then**
3: Set $\{C_1, \ldots, C_r\} = \arg\max_{C \in S_{cluster}} |C \cap S_{v_q}|$.
4: **if** $r > 1$ **then return** $S_{v_q} = \varnothing$
5: **else** $S_{v_q} = S_{v_q} \backslash (\cup_{C \in S_{cluster}} (C \cap S_{v_q}) \backslash \cup_{i=1}^{r} C_i)$.
6: **if** $|S_{v_q} \cap (S_{HE} \backslash v_q)| > 1$ **then**
▷ S_{v_q} contains multiple high entropy nodes beside v_q
7: Among nodes in $S_{HE} \backslash v_q$, keep only the node(s) which have the highest transition probability from v_q
▷ nodes not in S_{HE} are not affected
8: **return** S_{v_q}

probability from v_q is highest. This pruned list is given to Algorithm 1, where it is checked against existing clusters in $S_{cluster}$, and if there is any intersection, then they are merged, otherwise, S_{v_q} is added as a new cluster in $S_{cluster}$.

Having identified localized query-node centric community structures, in the ***second stage***, we agglomerate these to identify clusters at different degrees of granularity. A single stage of agglomeration is almost identical to the initial clustering process described above, with the following subtle changes. The cluster results from the previous step are considered as the new nodes. We still only use the matrix $\hat{P}$, and hence we do not (need to) explicitly define edges connecting the clustered nodes. The new coalesced nodes are assigned an entropic centrality value corresponding to the average of the entropic centrality of their constituent nodes. For transition probabilities across clustered nodes (say C and C'), we considered the minimum, mean and maximum transition probabilities amongst all node pairs $u \in C, v \in C'$ as per $\hat{P}$. Finally, we discard a specific agglomeration in case the resulting agglomerated cluster would not result in a weakly connected graph. Our experiments indicate that the best clustering results are obtained using the minimum transition probabilities, we only report the corresponding results next.

4.5 Experimental Results

The applicability of the proposed framework is to explore with all combinations of graph (un/directed and un/weighted graph). We apply our clustering algorithm on the Karate club network [13], a cocaine dealing network [6] and a subgraph

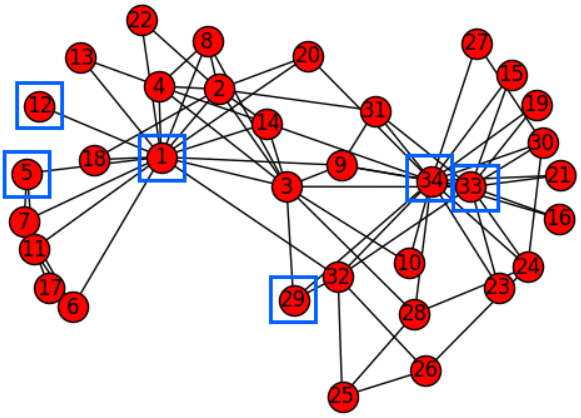


FIGURE 4.1: The karate club network: we show in Fig. 4.2 the path and Markov entropic centralities (for different values of D and t) for the nodes 1, 5, 12, 29, 33 and 34.

extracted from Bitcoin transaction log [21, 22]. Depending on various factors, such as connectivity, directionality, and weight, node centralities and communities are shown to be different.

4.5.1 A Markov Entropic Centrality Analysis of the Karate Club

Consider the karate club network [13] (used as an example in [5]) shown in Fig. 4.1. We use this small social network comprising 34 members (nodes) as a toy example to illustrate and validate some of the ideas explored in this chapter. The 78 edges represent the interactions between members outside the club, which eventually led to a split of the club into two, and are used to predict which members will join which group. This is an undirected unweighted graph, which is treated as a particular case of directed graph ($d_{out}(u)$ is the degree of u). Let P denote the transition matrix such that $P_{uv} = \frac{1}{d_{out}(u)}$ for every $u \in V$ and every neighbour v of u. The work [5] explores the choice of D (and t) in the context of clustering. Here, we start by investigating the role of D in terms of the resulting Markov entropic centrality, for t finite ($t = 1, \ldots, 6$ since the karate club network has a diameter of 5) and asymptotic (using Lemma 4.1).

Influence of D Fig. 4.2 illustrates the Markov entropic centrality $C_H^t(u)$ of the nodes $u \in \{1, 34, 33, 29, 12, 5\}$ for different values of D: for $D = w\mathbf{I}_{34}$, with

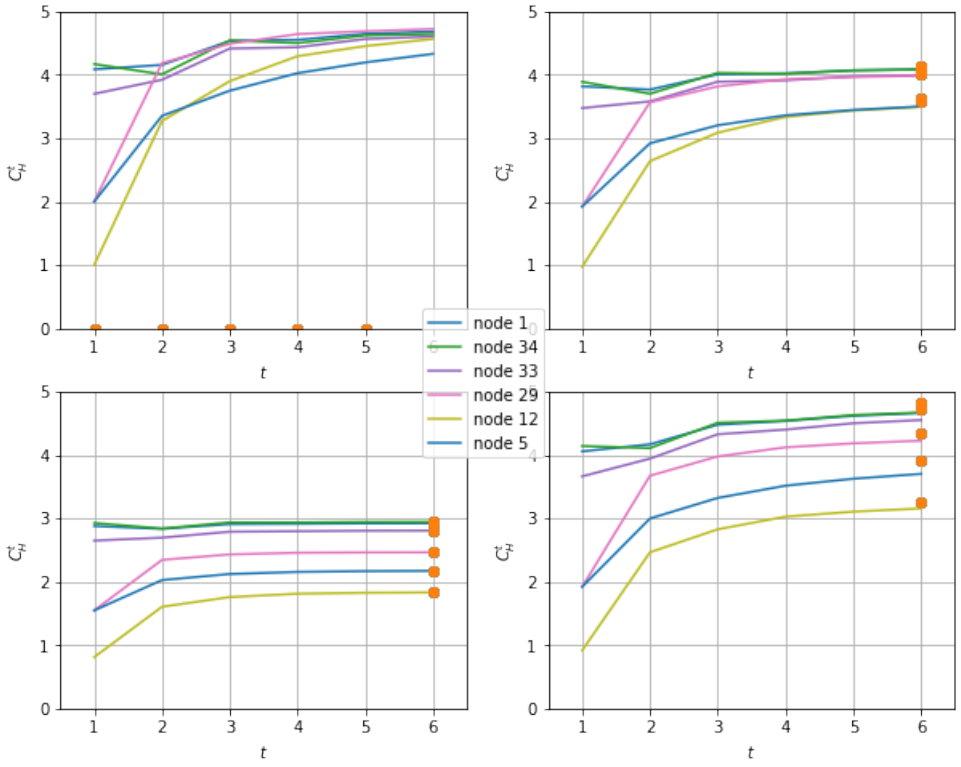


FIGURE 4.2: The Markov entropic centrality $C_H^t(u)$ for $u = 1, 34, 33, 29, 12, 5$ (from Fig. 4.1) for $t = 1, 2, 3, 4, 5, 6$: on the upper left, $D = 0.001\mathbf{I}_{34}$, on the upper right, $D = \frac{1}{5}\mathbf{I}_{34}$, on the lower left, $D = \frac{1}{2}\mathbf{I}_{34}$ and for D such that $D_{uu} = \frac{1}{d_{out}(u)+1}$ on the lower right. The dots show the asymptotic values $C_H^{\inf}(u)$.

$w = 0.001$[4], $0.2, 0.5$, we observe that $C_H^t(u)$ is decreasing and flattening, for all nodes and for every value of t. This is expected, since when the probabilities at the auxiliary nodes are increasing as a constant, the overall uncertainty about the random walk is reducing. Thus the higher the absorption probability, the greater the attenuation of the entropic centralities for all nodes. For D such that $D_{uu} = \frac{1}{d_{out}(u)+1}$ (shown on the lower right subfigure), the Markov entropic centralities are more separated than previously: indeed, a node with small degree then has a high absorption probability, which induces a large attenuation on its entropic centrality. The net effect is a wide gulf in the centrality scores between nodes with low and high degrees.

[4]We cannot use $D = \mathbf{0}$ since this means no absorption probability. Also the matrix $(\mathbf{I}_n - \tilde{P})$ in Lemma 4.1 would have no reason to be invertible.

Influence of t We notice several subtleties regarding the time evolution of the Markov entropy centrality values[5].

The most striking one is how the centrality of a node can be influenced over time by its neighbors. This is seen for example on the upper left corner for nodes 12 and 5. Node 12 starts (at $t = 1$) with an entropic centrality significantly lower than node 5 - indeed node 5 has more neighbors than 12, yet node 12 reaches a "hub" (a node of itself high centrality), node 1, at $t = 2$, and thus for $t \geq 2$, its entropic centrality grows, and eventually ends up being almost as high as that of node 1 itself. In contrast, even though node 5 reaches the same hub, it also belongs to a local community within the graph, inside which a significant amount of its influence stays confined. This explains why node 5 ends up having a lower entropic centrality than node 12 in particular. In the other three plots, we have assigned a significant volume of the flow to be absorbed at the auxiliary nodes, which has a net effect of attenuating the absolute values of entropic centrality for all nodes, i.e., a downward shift and flattening. Taken together with the initial gap among the centralities of nodes 5 and 12, we thus do not observe the overtake in these experiments, unlike in the case where the absorption probability was negligible.

Another less prominent behaviour that we notice for node 34, and to a certain extent for node 1, is that there is a dip in their entropic centralities at the beginning, before they rise up to their long term values. We hypothesize that this is because some of the immediate neighbors of these nodes form local groupings (for instance, 34's neighbor node 15 has only 33 and 34 as neighbors, node 27 has only nodes 30 and 34 as neighbors), and thus after the initial dispersal of influence among a relatively large number of these neighbors (hence the initial high centrality), their influence gets concentrated within the local cluster with the initiating node itself for a while (thus the dip), before reaching out more evenly to the rest of the network, leading to another rise in entropic centrality.

Table 4.1 compares different centralities. Both the path and the asymptotic Markov centralities give the same ranking, it is similar to the ranking given by the degree centrality C_D (with some difference regarding node 5 and 12, which are explained by the above discussion). The betweenness and load centrality agree on their ranking, which is different from the other ones. Table 4.1 demonstrates that asymptotic Markov centrality captures a unique perspective from path-based

[5]We have $C_H^1(29) = C_H^1(5) = 2 = 4\log_2(4)$ because both nodes have degree 4, this differs from [5] where they assume that the degree of 29 is 3.

node	C_H	C_H^∞	C_D	C_B	C_L
node 34	4.83992	4.82504	0.5151	0.3040	0.2984
node 1	4.83041	4.81999	0.4848	0.4376	0.4346
node 33	4.76892	4.72539	0.3636	0.1452	0.1476
node 29	4.50092	4.34323	0.0909	0.0017	0.0017
node 5	4.07244	3.90674	0.0909	0.0006	0.0006
node 12	3.39469	3.26763	0.0303	0	0

TABLE 4.1: Different centrality measures: C_H and C_H^∞ are the path (4.1) and asymptotic Markov entropic centralities (4.2), C_D, C_B, C_L are resp. the degree, betweenness and load centralities.

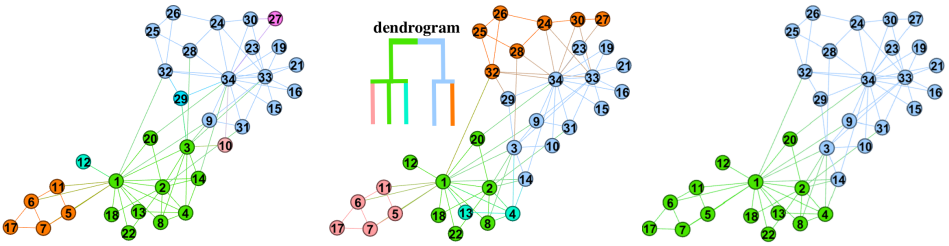
entropy centrality where a node gains high centrality (influence) from neighbors by transitivity.

We hypothesize that changes in the Markov entropic centralities over time are indicative of local communities in the graph, with changes in gradient indicating traversal of boundaries from one community to another. We will explore and exploit this observation to design a clustering algorithm in Section 4.4.

4.5.2 Clustering of the Karate Club Network

We apply our clustering algorithm to the karate club network [13], as illustrated in Figure 4.3, to illustrate and analyse the workings of the clustering algorithm with a toy example with known baseline, before studying larger graphs. In subfigure 4.3b, the initial set S_{ini,v_q} of clusters is shown along with the dendrogram for agglomeration, and subfigure 4.3c shows the final clusters. Clustering obtained using the edge removal technique (20 iterations) from [5] is shown in subfigure 4.3a for comparison. This top-down clustering approach follows the idea of edge removal from [12], but using the reduction of average Markov entropic centrality to determine which edge to remove. The implementation and exploration of clustering in [5] was restricted to undirected and unweighted graphs (since their entropy centrality model itself did not handle weighted or directed graphs), and the largest graph studied comprised 115 nodes and 613 edges.

For a point of comparison, we provide the computation time for asymptotic $\hat{P}$ and entropy centralities for the karate club graph: the edge clustering [5] took 14.154 seconds, while our final 2-clustering results were computed in 0.026 seconds. All our experiments were run on a 64-bit PC with x64-based Intel(R) Xeon(R) CPU E5-1650 0 @3.20GHz processor and 16 GB RAM. This is easily explained: in our



(A) Edge removal (20 iterations). (B) Initial clusters (stage 1) (C) The final 2-cluster result (stage 2)

FIGURE 4.3: Clustering of the karate club network: using the edge removal clustering of [5] on the left, and the proposed algorithms in the middle (stage 1) and on the right (stage 2, with agglomeration).

algorithm, the transition probability matrix $\hat{P}$ needs to be computed only once. In contrast, even within a single iteration, the edge removal algorithm [5] needs to recompute the said matrix for every graph instance created by removal of each possible edge, to determine which edge to remove, and this exercise is then repeated in every iteration. That accounts for the huge discrepancy in the computation time, and demonstrates the computational efficacy of our approach. Furthermore, given the bottom-up agglomerative approach, where we explore local communities in the graph first, we argue that our algorithm can more naturally be executed in a distributed manner (though this is outside the scope of our current presentation, and we are yet to explore the aspect of distribution for further scalability). Finally, the results obtained for the karate club network indicate that our algorithm produced good clusters, based on the established ground truth [13]. Note that, though our proposed clustering technique is based on flow-confinement in network, that is an abstraction and its efficacy is not limited to networks which actually have some sort of underlying flow. More experiments are discussed in Section 4.6.

4.5.3 A Weighted Markov Entropic Centrality Analysis of a Cocaine Dealing Network

We consider the cocaine dealing network [6], a small directed weighted graph coming from an investigation into a large cocaine traffic in New York City. It involves 28 persons (nodes), and 40 directed weighted edges indicate communication exchanges obtained from police wiretapping.

The (weighted) Markov entropic centrality depends on the choice of the absorption matrix D, we thus start by looking at how a change in D influences the

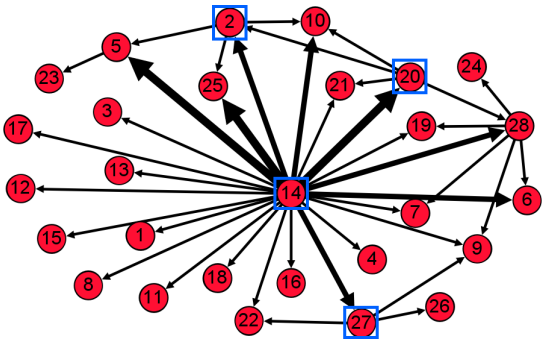


FIGURE 4.4: The cocaine dealing network [6]: weighted edges are drawn with quantized girth. Fig. 4.5 shows the (weighted) Markov entropic centrality of the nodes 2, 27, 20 and 14.

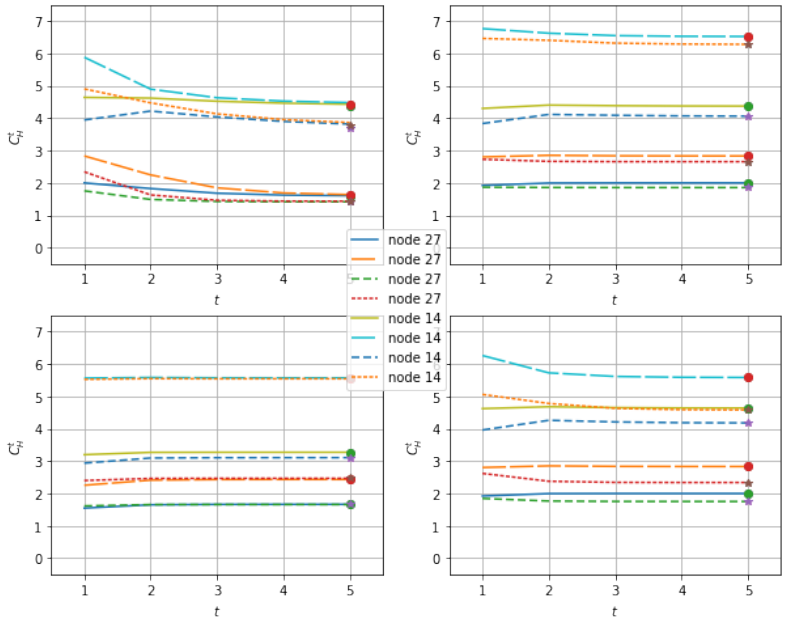


FIGURE 4.5: The Markov entropic centrality $C^t_{\alpha(w),H}(u)$ for $u = 27, 14$ (from Fig. 4.4) for $t = 1, \ldots, 5$: on the upper left, $D = 0.001\mathbf{I}_{28}$, on the upper right, $D = 0.2\mathbf{I}_{28}$, on the lower left, $D = 0.5\mathbf{I}$ and D with $D_{uu} = \frac{1}{d_{w,out}(u)+1}$ on the lower right.

entropic centralities. We keep the same choices for D as for the unweighted case, namely $D = 0.001\mathbf{I}_{28}$, $D = 0.2\mathbf{I}_{28}$, $D = 0.5\mathbf{I}$ and D such that $D_{uu} = \frac{1}{d_{w,out}(u)+1}$, since when $w(e) = 1$ for all edges, then $d_{w,out}(u) = d_{out}(u)$ for all nodes.

In Fig. 4.5, we plot the (weighted) Markov entropic centrality $C^t_{\alpha(w),H}(u)$ for $u = 27, 14$ for $t = 1, 2, 3, 4, 5, 6$, for the 4 choices of absorption probabilities D. For $u = 27, 14$, 4 variations of Markov entropic centralities are considered: $\alpha(w) = 1$ and $\mu(v) = 1$ (straight lines), corresponding to the unweighted case, $\alpha(w) = 1$ and $\mu(v) = \left(\frac{d_{w,out}(v)}{d_{out}(v)}\right)$ (long dash lines) for the case where the transition matrix

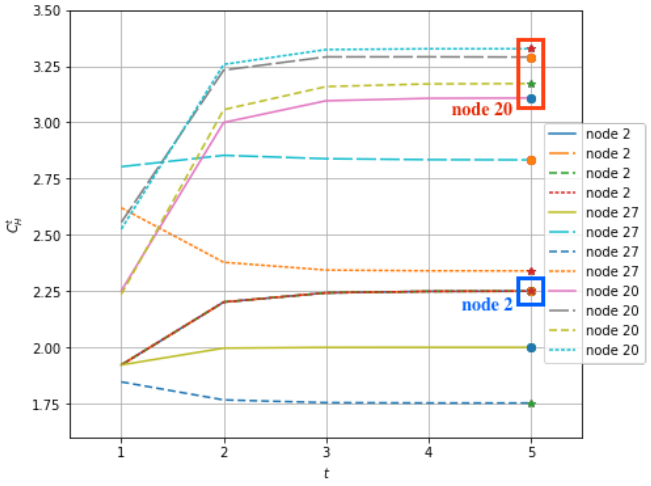


FIGURE 4.6: The Markov entropic centrality $C^t_{\alpha(w),H}(u)$ with $D_{uu} = \frac{1}{d_{w,out}(u)+1}$: for $u = 2, 27, 20$ (from Fig. 4.4), for $t = 1, \ldots, 5$ and for $(\alpha(w), \mu(v)) = (1, 1)$ (straight lines), $(1, \frac{d_{w,out}(u)}{d_{out}(u)})$ (long dash lines), $(w, 1)$ (short dash lines) and $(w, \frac{d_{w,out}(u)}{d_{out}(u)})$ (dotted lines).

P is used, $\alpha(w) = w$ and $\mu(w) = 1$ (short dash lines) to show how centrality is computed based purely on P_w, and finally $\alpha(w) = w$ with $\mu(v) = \left(\frac{d_{w,out}(v)}{d_{out}(v)}\right)$ (dotted lines), for which P_w is used for the transition matrix, together with the weighted entropic centrality. Nodes 27 and 14 are different in nature, in that node 14 is a hub, with high degree compared to the rest of the other nodes, while node 27 has only one incoming edge and two outgoing edges. They are chosen as representatives of nodes of respectively high and low degree (even though the degree is not the only contributing factor in the node centrality, it still plays an important role for small values of t). For each of the 4 plots, the 4 upper lines characterize the behaviour of node 14, and the 4 lower lines the one of node 27. We observe that the behaviour of the entropic centralities when $\mu(v) = 1$ are similar to what was already observed for the karate club network: for $D = 0.2\mathbf{I}_{28}$ and $D = 0.5\mathbf{I}$, the centralities are flattened because the path uncertainty is reduced when the absorption probabilities are increasing, while for D such that $D_{uu} = \frac{1}{d_{w,out}(u)+1}$, the gap among the centralities is (slightly) increasing. When $\mu(v) = v$, both centralities are amplified. Since node 14 has many outgoing edges, with weights including 10, 11, 14 18, 19, the introduction of $\mu(v)$ creates a higher amplification for node 14 than for node 27, whose edge weights are all less than 5.

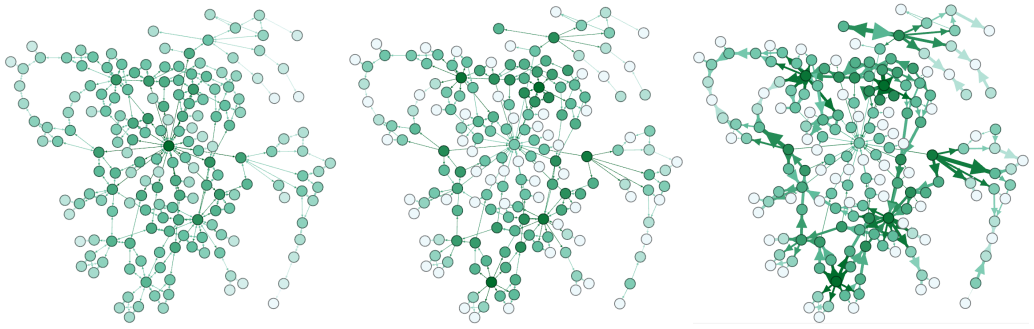
We then focus on the case where $D_{uu} = \frac{1}{d_{w,out}(u)+1}$, which depends only on the network setting, instead of other choices of D which are parameters whose range can be explored one by one. With this choice of D, we consider the (weighted)

Markov entropic centrality for $u = 2, 27, 20$, as displayed in Fig. 4.6. These nodes have outgoing edges with weights $1, 1, 1$, for $u = 2$, weights $4, 3, 1$ for $u = 27$, and weights $2, 1, 1, 1$ for $u = 20$. These nodes are chosen because they have similar out-degrees, but different weighted out-degrees. The same 4 scenarios as above are considered. For node 2, its edge to 10 stops at 10, its edge to 5 has only one connection to node 23 which has weight 1, and its last edge goes to 25, which has no connection either, therefore the 4 centralities are actually of the same quantity, and there all the 4 lines are coincident. For node 27, since it has the same out-degree as node 2, in the unweighted case, it starts at the same centrality as node 2. However it is even less influential since none of its own neighbours have neighbours. In the 3rd scenario, its entropic centrality is even lower than in the unweighted case, which is normal, since in this case, the walk distribution is not uniform anymore. In the two cases where $\mu(v) = v$, we see a jump in the centrality, explained by the weights of the outgoing edges which are higher than for node 2. However, the centrality of node 27 remains below that of 20, whose weighted out-degree is less than that of 27, but its out-degree is actually more.

4.5.4 A Markov Entropic Centrality Analysis of a Bitcoin Subgraph

From section 4.5.1 and 4.5.3, we observe that the proposed Markov entropic centrality measure is suitable for undirected, directed and weighted graphs. We next consider a small subgraph, extracted from the Bitcoin network. We may want to apply entropic centrality on small subgraph with three different perspectives, specifically subgraph as undirected graph, as directed unweighted graph and as directed weighted graph.

We want to investigate the performance of Markov entropic centrality mechanism in the 'wild', for an arbitrary flow-based graph which may or not even have very well defined hub/clusters to begin with. Accordingly, we next consider a small subgraph extracted from the Bitcoin network [21] comprising 178 nodes and 250 edges. Nodes are Bitcoin addresses, and there is an edge between one node to another if a Bitcoin payment has been made. Since the proposed Markov entropic centrality measure is suitable for undirected, directed and weighted graphs, we look at the chosen Bitcoin subgraph as an undirected unweighted graph by



(A) (B) (C)

FIGURE 4.7: Asymptotic Markov entropic centralities for a 178 node subgraph of the Bitcoin network: undirected and unweighted on the left, directed and unweighted in the middle, directed and weighted on the right. The top 10%, next 20%, 30% and remaining 40% nodes in terms of their entropic centrality are shown in progressively lighter colours.

ignoring the direction of transactions, as a directed unweighted graph by just considering whether any transactions exists, and as a directed weighted graph (with $\alpha(w(e)) = w(e)$, $\mu(v) = \frac{d_{w,out}(v)}{d_{out}(v)}$) to capture the effect of transaction amount.

In Fig. 4.7 (drawn using Gephi [10]) we show the three variations with nodes coloured according to their Markov entropic centrality at $t = \infty$ (asymptotic). The darker the node colour, the higher the Markov entropic centrality. In subfigure 4.7a, one hub stands apart, with the highest entropic centrality, which is a node which is highly connected to other nodes. If we look at the same node in subfigure 4.7b, we see that it is not central anymore: this is because in the directed graph, this node is actually seen to receive Bitcoin amounts from many other nodes, so as far as sending money is concerned, it has very little actual influence (in the same graph but with edge directions reversed, this node would have had the highest centrality). On the other hand, nodes that were not so prominent in the undirected graph appear now as important. The graph in subfigure 4.7b highlights nodes which are effectuating many Bitcoin transactions. Now one node may do several transactions with little Bitcoin amount, we may want a node that does few but high amount transactions to be more visible, which is illustrated in subfigure 4.7c. It turns out that for the subgraph we studied, high centrality nodes in the unweighted case remain high centrality nodes in the weighted case, meaning that nodes performing many Bitcoin transactions are also those sending high amount transactions.

For the sake of completeness, we provide Kendall-τ rank correlation coefficients among different variations of (weighted) Markov entropic centralities for the directed graph, for $t = \infty$: for $(\alpha(w(e)) = 1, \mu(v) = 1)$ (shown on subfigure

α,μ	1,1	d_{out}	w,1	$1,\frac{d_{w,out}(v)}{d_{out}(v)}$	$w,\frac{d_{w,out}(v)}{d_{out}(v)}$
1,1	0	0.159	0.075	0.084	0.086
d_{out}	0.419	0	0.193	0.214	0.218
w,1	0.315	0.60	0	0.120	0.050
$1,\frac{d_{w,out}(v)}{d_{out}(v)}$	0.186	0.538	0.305	0	0.089
$w,\frac{d_{w,out}(v)}{d_{out}(v)}$	0.346	0.671	0.097	0.283	0

TABLE 4.2: Kendall-τ coefficients of 5 centrality measures: 4 of them are the Markov entropic centralities for different values of $\alpha(w(e))$ and $\mu(v)$; d_{out} is the out-degree centrality.

graph	UU	$D_{1,1}$	$D_{w,1}$	$D_{1,M}$	$D_{w,M}$	UU_{er}
UU	1	0.416	0.506	0.372	0.271	0.125
$D_{1,1}$	0.416	1	0.473	0.806	0.364	0.146
$D_{w,1}$	0.506	0.473	1	0.408	0.416	0.122
$D_{1,M}$	0.372	0.806	0.408	1	0.413	0.145
$D_{w,M}$	0.271	0.346	0.416	0.413	1	0.131
UU_{er}	0.125	0.146	0.122	0.145	0.131	1

TABLE 4.3: Pairwise F-score among clusterings achieved for different graph variants. Legend — UU: undirected & unweighted; $D_{x,y}$: Directed with $\alpha = x$, and $\mu = 1$ if $y = 1$, else $\mu = \frac{d_{w,out}(v)}{d_{out}(v)}$ if $y = M$; UU_{er}: edge removal algo [5].

4.7b), for $(\alpha(w(e)) = w(e), \mu(v) = \frac{d_{w,out}(v)}{d_{out}(v)})$ (shown on subfigure 4.7c), but also for $(\alpha(w(e)) = w(e), \mu(v) = 1)$, and for $(\alpha(w(e)) = 1, \mu(v) = \frac{d_{w,out}(v)}{d_{out}(v)})$. The out-degree centrality is also tested. A Kendall-τ coefficient is a measure of rank correlation: the higher the measure, the higher the pairwise mismatch of ranks across two lists. Results are shown in Table 4.2. Above the diagonal, coefficients are obtained using all 178 nodes. Below the diagonal, only 67 nodes are used, which are in the union of the top 20% nodes for each of the 5 aforementioned centrality measures. In the given graph, many (peripheral) nodes have the same (low) entropic centrality, so when all nodes are considered for ranking correlation, the average is misleadingly low. Since high centrality nodes are often of interest, ranking variations among the top nodes is pertinent, and we observe that (i) out-degree is not a good proxy for most entropic centrality variants, and (ii) the different variations yield significantly distinct sets of high centrality nodes, corroborating the need for our flexible entropic centrality framework.

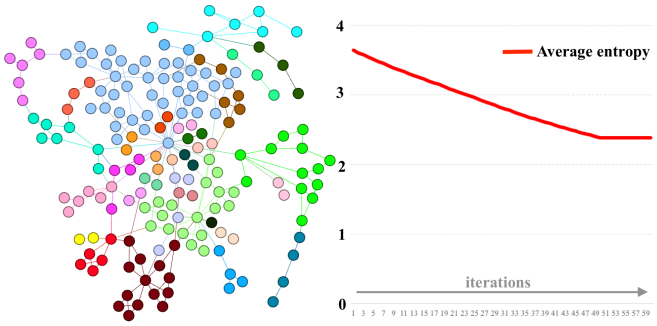


FIGURE 4.8: Clustering by iterative edge removal [5].

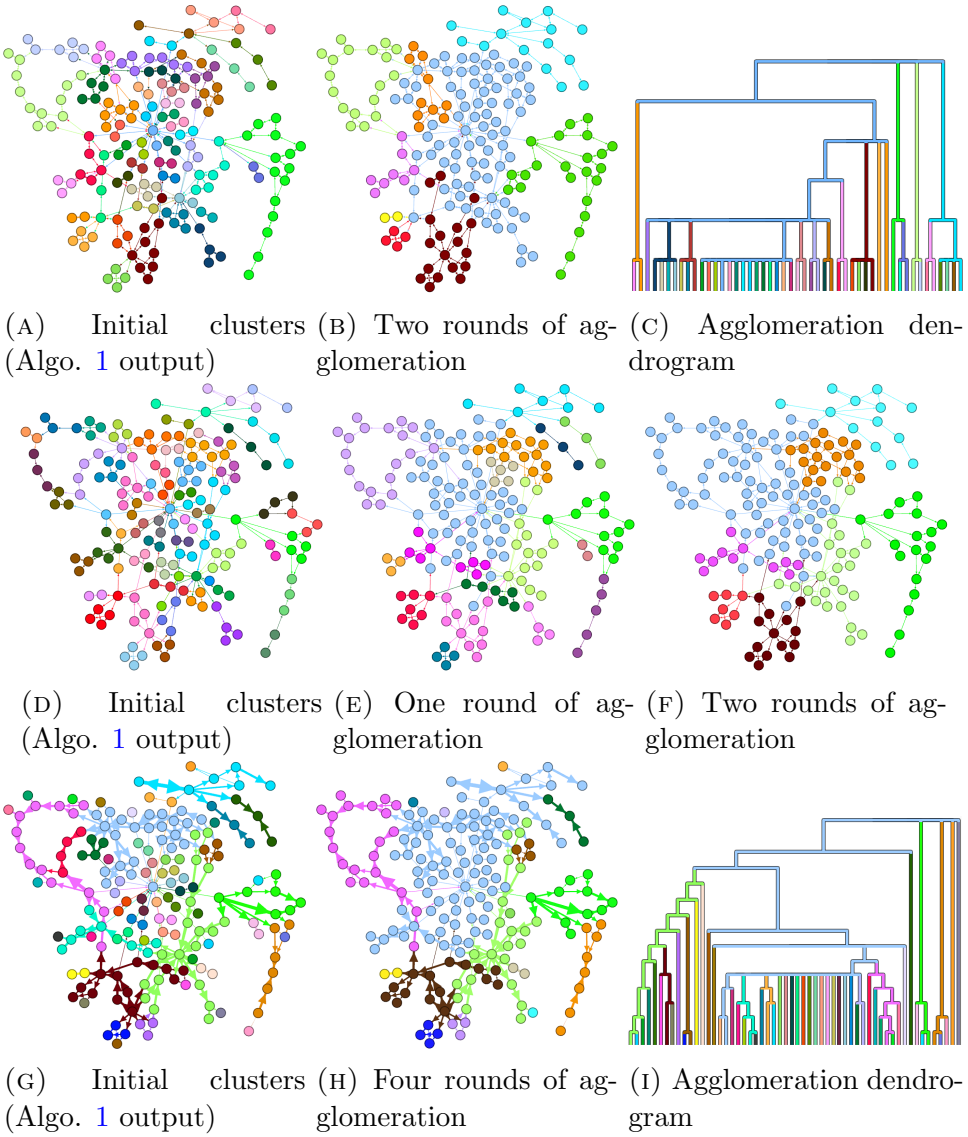
4.5.5 Clustering of a Bitcoin Subgraph

We apply our proposed clustering algorithm to variations (un/directed, un/weighted) of the 178 node Bitcoin network subgraph [21] and report the results in Fig. 4.9. The results obtained using the edge removal algorithm [5] on the undirected unweighted graph variant is shown on Fig. 4.8. Unless otherwise stated, we use as a parameter $N = 53$, essentially considering S_{HE} to comprise the top 30% entropic centrality nodes (see the sensitivity analysis below).

While it is visually clear from Fig. 4.9 that we obtain different clusters depending on the scenario considered, Table 4.3 confirms this by reporting F-scores [102] across the graph variants. Also the effect of the parameter μ (without the transition probabilities being altered by edge weights) is rather low. This reinforces an underlying motivation of our work, namely that which graph variant (un/directed and/or weighted) to study is application dependent, and hence having one graph clustering algorithm that works across all variants is beneficial.

The clusterings of the undirected unweighted graph found by our algorithm in 1.072 seconds (Figure 4.9) and by the edge removal algorithm in 3196.07 seconds (Figure 4.8) are very different (F-score of 0.125). Our algorithm visually yields (more) meaningful results, though in the absence of a ground truth, this assertion remains subjective. The agglomeration process in our algorithm could have been continued beyond the final result shown here, as indicated by the associated dendrogram. The edge removal based clustering was stopped at the 50th iteration, after which no further average entropy reduction was observed with any single edge removal.

When to stop agglomerating is typically purpose dependent. The second row of Fig. 4.9 shows various stages of agglomeration, the dendrogram for agglomeration is given for the 1st and 3rd row. There are many community structures that repeat



(A) Initial clusters (Algo. 1 output) (B) Two rounds of agglomeration (C) Agglomeration dendrogram

(D) Initial clusters (Algo. 1 output) (E) One round of agglomeration (F) Two rounds of agglomeration

(G) Initial clusters (Algo. 1 output) (H) Four rounds of agglomeration (I) Agglomeration dendrogram

FIGURE 4.9: Clustering of the 178 node Bitcoin subgraph (results obtained in 1.072, 1.077 and 1.123 seconds respectively): **1st row** - undirected unweighted graph, **2nd row** - directed unweighted graph, **3rd row** - directed weighted graph ($\alpha = 1,\ \mu = \frac{d_{w,out}(v)}{d_{out}(v)}$).

across the considered scenarios, and most of the cluster boundaries can be traced back to the high entropy nodes (Figure 4.7), yet there are also subtle differences, e.g., in the weighted directed graph, there are instances of single isolated nodes, which stay isolated for several interactions of agglomeration because of weak (low weight) connections.

Sensitivity analysis. For each graph variant, the size of S_{HE} was varied to comprise 10%, 20%, 30% and 40% of the top entropic centrality nodes. For the unweighted and weighted directed graph, the F-score between clusterings obtained

with 10% and the others are 0.824 and 0.989 respectively, while all the other pairs have F-score of 1. This suggests very consistent results in these cases, irrespectively of the choice of $|S_{HE}|$. However for the unweighted undirected graph, $|S_{HE}|$ has a significant impact, with the F-score between 20% and 40% being the lowest at 0.626, while the best score of 0.912 is obtained between 30% and 40%. This justifies the use the top 30% entropic centrality nodes for the reported results.

We next want to explore whether some other (simpler to compute, e.g., clustering coefficient) ways of choosing the query nodes as a variation of our choice of entropy centrality ranked list for query nodes, and applying the rest of the algorithm as is, would also work well. This idea is inspired from [166] where authors used clustering coefficient list for initial seed for their community detection techniques. Additionally, we also compare our approach with other widely used community detection techniques, specifically, InfoMap [7], Louvain [8] and label propagation [9].

We first consider the 178 nodes Bitcoin subgraph with unweighted directed edges. The results obtained by the above mentioned algorithms and variations are shown in Figure 4.10. For meaningful comparison, communities detected by our proposed clustering and two variants are obtained from two iterations of agglomeration. By using clustering coefficient (ascending order) list, we can see a community with blue color can be split into two smaller communities by single link. While changing to clustering coefficient node list with descending order, the result obtained is improved visually, and we can see that a blue cluster comprises more than half of the subgraph. Meanwhile, results by InfoMap and Louvain splits subgraph into ten communities, and a cluster with cyan color is clearly seen as two smaller clusters without any connected link for both techniques. Furthermore, a result obtained from label propagation splits the graph into four big communities and outliers located around centre of subgraph.

A visual inspection of the resulting clusters shown in Figure 4.10 suggests that our original clustering algorithm yields more meaningful results than the others (followed by the variant using query nodes as nodes with lowest clustering coefficients), albeit in the absence of a ground truth, this assertion remains subjective. In order to determine how distinct the different results are, we compute F-scores [102] across above mentioned results, and report this comparison in Table 4.4. As expected based on the prior visual inspection, with F-score of 0.515, using clustering coefficient nodes (descending order) produces closest result to our original

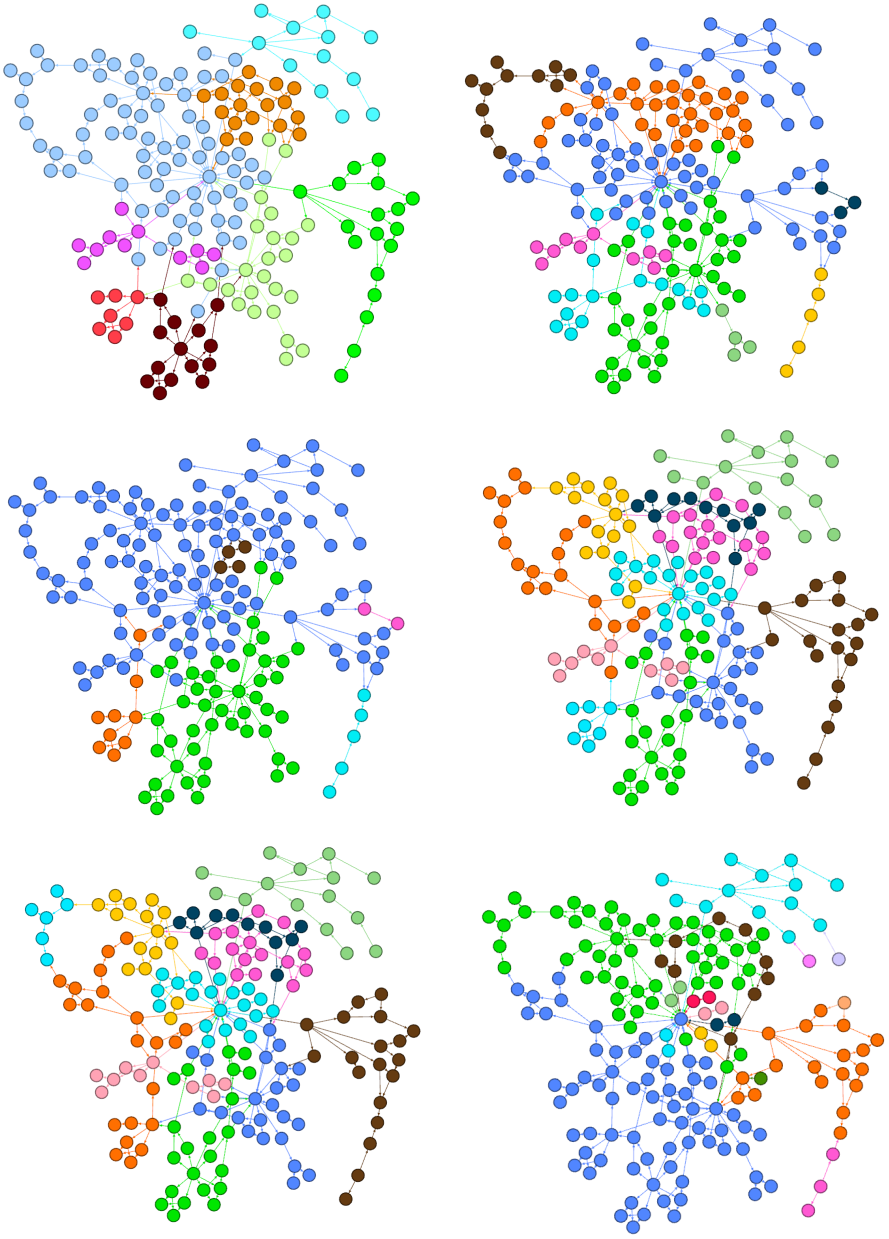


FIGURE 4.10: We show the clusterings of a Bitcoin subgraph with 178 nodes. The plots are organized as follows. From **left to right** and **top to bottom**: our proposed technique (two rounds of agglomeration), variation of our techniques (two rounds of agglomeration - query nodes chosen from a list of nodes ordered based on clustering coefficients in the ascending order), variation of our proposed technique (two rounds of agglomeration - query nodes chosen from a list of nodes ordered based on clustering coefficients in the descending order), InfoMap [7], Louvain [8] and label propagation [9]. The plots are drawn using Gephi [10].

graph	$D_{1,1}$	$D_{1,1,C,A}$	$D_{1,1,C,D}$	$D_{InfoMap}$	$D_{Louvain}$	D_{Label}
$D_{1,1}$	1	0.425	0.515	0.186	0.482	0.367
$D_{1,1,C,A}$	0.425	1	0.558	0.192	0.421	0.310
$D_{1,1,C,D}$	0.515	0.558	1	0.093	0.268	0.370
$D_{InfoMap}$	0.186	0.192	0.093	1	0.376	0.178
$D_{Louvain}$	0.482	0.421	0.268	0.376	1	0.355
D_{Label}	0.367	0.310	0.370	0.178	0.355	1

TABLE 4.4: Pairwise F-score among clusterings achieved for different community detections. Legend — $D_{1,1}$: Directed with $\alpha = 1$, and $\mu = 1$, $D_{1,1,C,A}$: Directed with $\alpha = 1$, $\mu = 1$, query nodes from clustering coefficient list with ascending order, $D_{1,1,C,D}$: Directed with $\alpha = 1$, $\mu = 1$, query nodes from clustering coefficient list with descending order, $D_{infoMap}$: Directed with InfoMap algo [7], $D_{Louvain}$: Directed with Louvain algo [8], D_{Label}: Directed with label propagation algo [9].

clustering (but it is still significantly different). In addition, InfoMap has its own unique way to partition the 178 nodes subgraph, its result is most inconsistent to what is obtained from our clustering result with F-score of 0.186. On the other hand, Louvain and label propagation techniques produce closer clusters visually and their results are likewise relatively more consistent with our proposed clustering with F-score of 0.482 and 0.367 respectively. Further, we notice that the results from these algorithms are not particularly close among each other either. For instance, the F-score between results from Louvain and InfoMap is 0.376, between Louvain and label propagation, it is 0.355, and between label propagation and InfoMap it is 0.178. Each row in the Table highlights the closest results (excluding self). The two closest algorithms are the two variants of our algorithm both of which use clustering coefficients (lowest and largest respectively) as query nodes. When not the closest (as is the case with Louvain) the result from our algorithm is the next closest to these other algorithms, for this data set.

We next consider another Bitcoin subgraph, but this time, we use a specific subgraph [22] of 4571 nodes where we have some further information, because it is constructed around Bitcoin addresses suspected to be involved in the Ashley-Madison extortion scam [32]. The result of our clustering algorithm is shown in Fig. 4.11: (1) the graph was considered as unweighted, since we are interested only in node connection here, (2) the asymptotic Markov entropic centrality was used (we have no specific diameter of interest), (3) the top 30% nodes with the highest centrality are treated as central (and thus unsuitable) when determining the choice of starting nodes for random walkers, (4) five agglomeration iterations

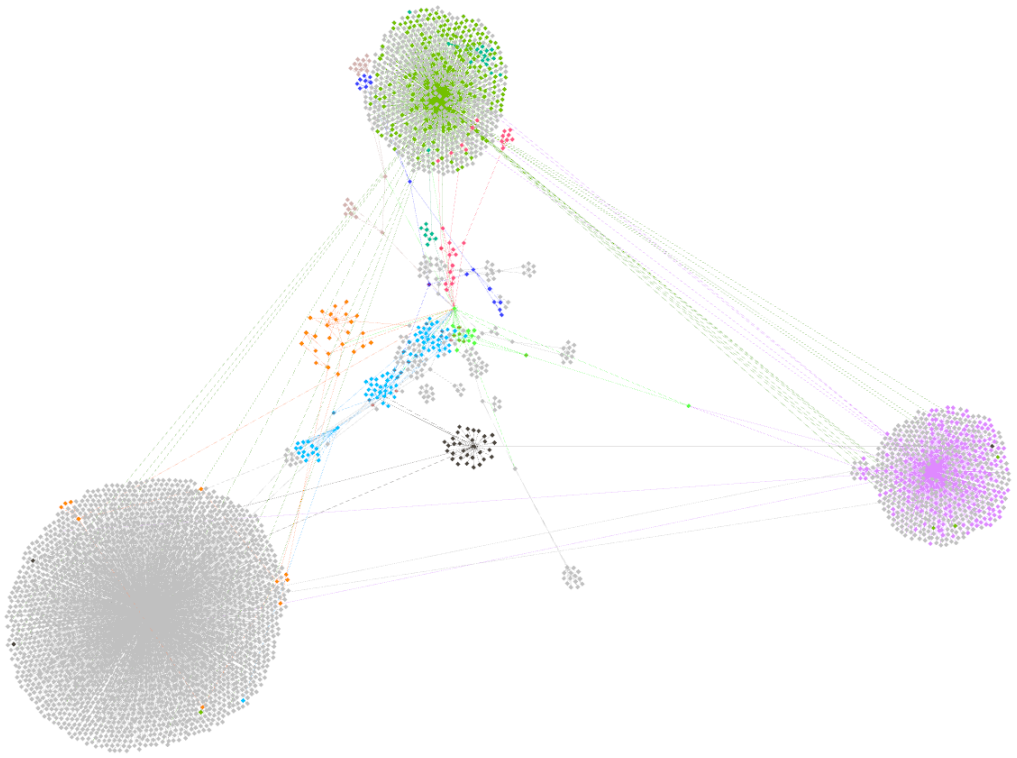


FIGURE 4.11: Clustering of a Bitcoin subgraph involved in the Ashley-Madison extortion scam.

were performed for the clustering algorithm.

The graph is drawn using Gephi, and some of the node positions were manually adjusted so that they can be easily inspected visually (same color is used to indicate nodes which have been identified to be in a given cluster by our algorithm) next to each other. There are three visually obvious main clusters: the upper green cluster, the purple cluster on the right, and the grey cluster on the left. The first observation is that the grey color here only represents nodes whose cluster size is too small to be significant (only 5 iterations were performed), they are thus kept in grey so as to make the other clusters more visible. The green and purple clusters are easily interpreted: each contains one Bitcoin address that is a hub for all its neighbours.

We then zoom into the central clusters, shown on Fig. 4.12. We observe an orange cluster on the left. Why and whether the involved addresses form a meaningful cluster can be determined by checking the relationship among addresses (e.g. by using `blockchain.com/explorer`). The Bitcoin addresses *1EaQSXEnaLk6XnBCsHngDU3PGXYv7Ky3RE* and *1HXLhwF2rHaY7XARQuq7p9P8RQqir5BZCp* appear as inputs/outputs in three Bitcoin transactions and thus tie up the

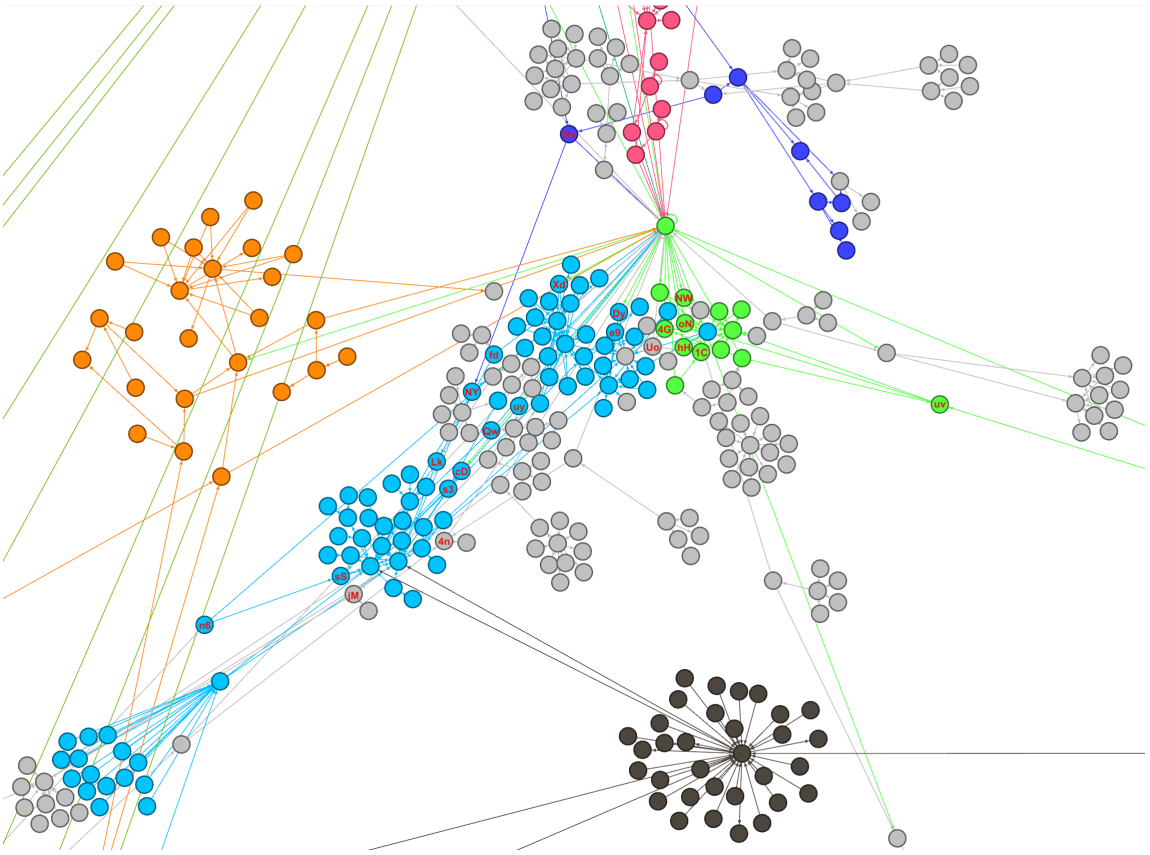


FIGURE 4.12: Zoom into the above clustering, labelled nodes are suspect in the Ashley-Madison extortion scam.

other addresses involved in these transactions. The blue and green clusters are particularly interesting because they contain nodes that are suspected of extortion in the Ashley-Madison scam (they are labelled in the figure). Being able to locate local clusters around them points at nodes among which the scam money could be circulating, and thus identifies further new suspect nodes. We can correlate this result with our analysis discussed later in Chapter 6 (see Subsection 6.5.2.3), where new suspect nodes are identified using a Bitcoin flow confluence algorithm. This example thus helps us in showcasing that the clusters obtained by the proposed techniques are meaningful, and are cross-validated with another approach. It also exposes one potential application (Bitcoin forensics) of our clustering algorithms: given a large graph, identify small subgraphs where a substantial flow of money (Bitcoin) is confined, to be studied in more detail individually.

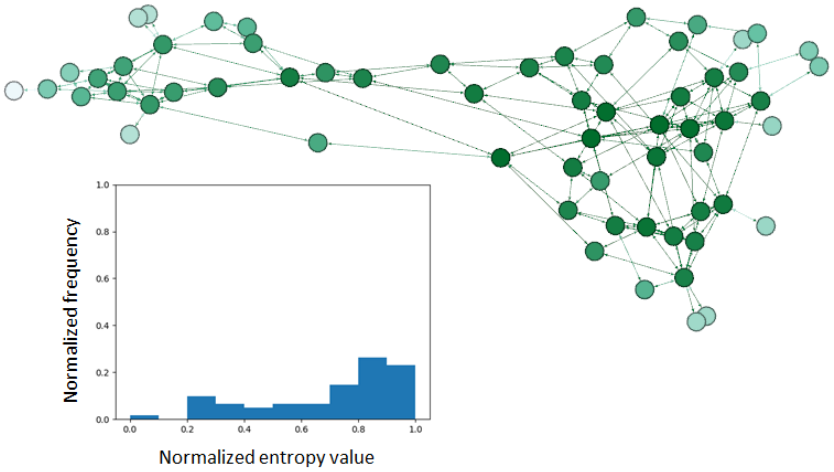


FIGURE 4.13: The asymptotic Markov entropic centralities for the dolphin network [11] are shown (darker the color, higher the relative entropic centrality score). The histogram of the normalized (by the maximum) centralities distribution for the same network is also shown.

4.6 Clustering of Networks with Known Ground Truth

In the previous section we studied our algorithm with graphs which are induced by flows, and hence there is a natural correspondence to how our algorithm works. However, we do not have any ground truth regarding those networks. In order to assess the applicability of our propose algorithms to other graphs even when those graphs are not induced by a flow, and also because we could not locate any flow induced graph with known ground truth, we run experiments on two networks with known ground-truth, namely, the dolphin network [11] and the America college football network [12] which were previously used in the work [5] that we extend. Also, we provide comparison with other popular community detection algorithms, InfoMap [7], label propagation [9] and Louvain clustering [8].

4.6.1 Clustering of the Dolphin Network

We consider the "Dolphin Network" [11] for our next experiment. It is an undirected social network where bottlenose dolphins are represented as nodes and association between dolphin pairs are represented as links. The network comprises 62 nodes and 159 links, and it was noticed that the dolphin community split into two communities [11] comprising 20 nodes and 42 nodes. We use this as the ground truth.

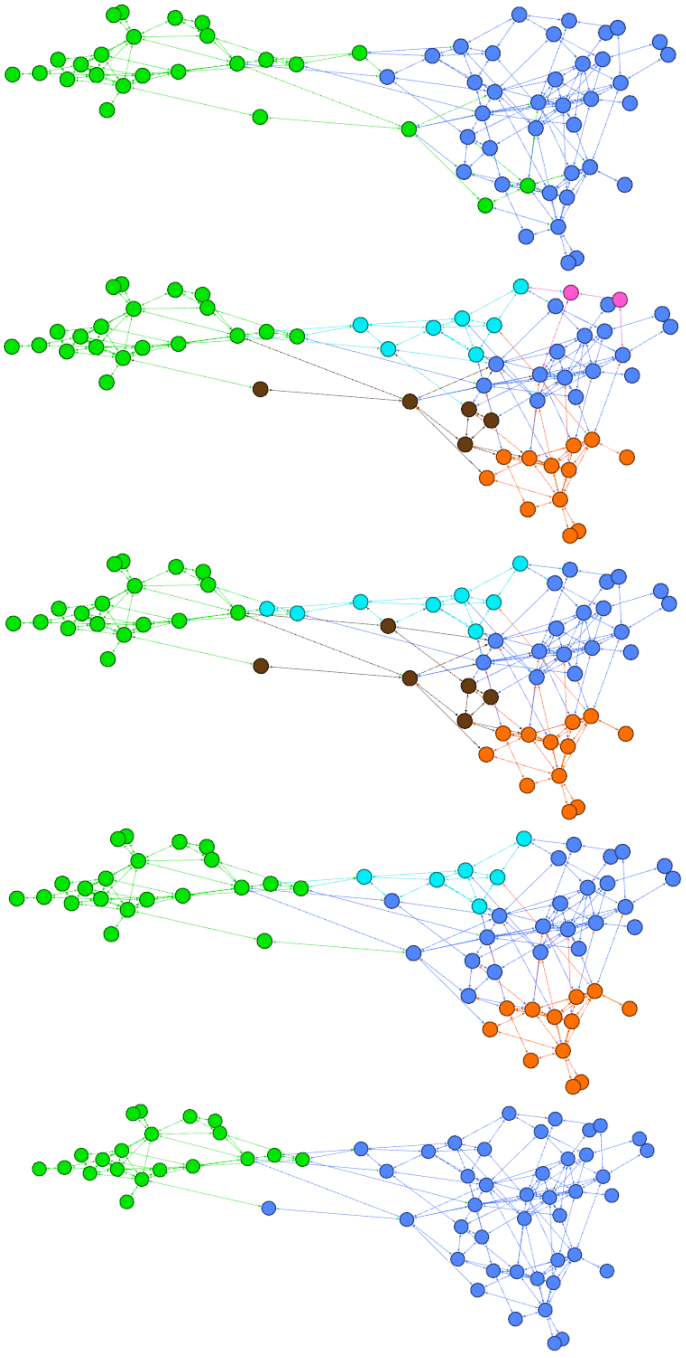


FIGURE 4.14: We show the clusterings of the dolphin network [11]. The plots are organized as follows. From **top to bottom**: our proposed techniques (second iteration and top 60%), InfoMap [7], Louvain [8], label propagation [9] and the ground truth [11]. These plots are drawn using Gephi [10].

We start by analyzing the dolphin network with our Markov entropy (unweighted undirected variation) computation. We show the network structure in Figure 4.13. The darker the node color, the higher the Markov entropic centrality. In the same figure 4.13 we show the entropy centrality (fractional, bucketed) distribution, where entropic centrality value is normalized by maximum entropic centrality value. We observe that more than 60% of the nodes have normalized entropy value above 0.7. This indicates that it would be more meaningful to choose clustering parameter S_{HE} to exclude top 50%-70% (instead 30% as previously used in Section 4.5.4) top entropic centrality nodes from being query nodes in our clustering algorithm.

We applied our proposed clustering (we tried several choices of S_{HE} from the range indicated above, and show the result obtained with the choice of 60%) and other clustering techniques (with their default parameters) including InfoMap [7], Louvain [8] and label propagation [9] on the dolphin network. The results are shown in Figure 4.14. We observe visually that our proposed clustering produces a better result compared to other clustering. To confirm this rigorously, we calculate F-score [102] for each clustering result against the ground-truth. Our proposed clustering result obtained F-score of 0.858. Even though, our clustering technique fails to capture the same communities as ground truth, it produces a reasonably good result. This shows applicability of our clustering technique even on networks which are not flow-based. In contrast a result with InfoMap obtained F-score of 0.545, a result with Louvain obtained F-score of 0.565 and a result with label propagation obtained F-score of 0.657. Recall that an ideal result would yield a F-score of 1. We note that other results obtained from other community detections have lower F-score, because their results contain more than two community. From Figure 4.14, we also visually infer that the other clustering results could be improved if agglomeration techniques were applied to the smaller communities located on right hand side of the dolphin network. This indicates several further fundamental issues in designing and validating clustering algorithms. The first is the nature and interpretation of ground truth. Even though one considers a third party's interpretation of ground truth, that is not necessarily an absolute or unique way to look at the data. Likewise, instead of F-score against ground truth, if one uses a different metric (such as maximization of modularity), as is done in the Louvain algorithm, then the notion of 'best performance' will again change. Thus, use of ground truth and/or use of any particular metric, while a standard practice, needs

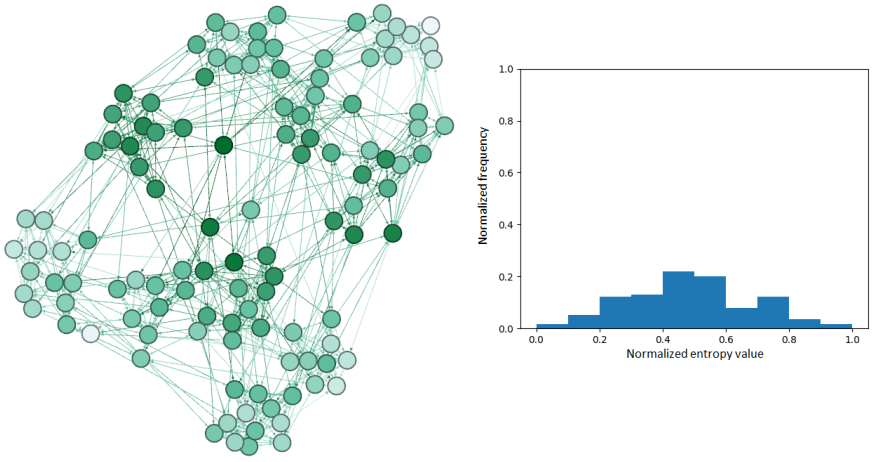


FIGURE 4.15: The American College football network [12] is shown (darker the color, higher the relative entropic centrality score). The histogram of the normalized (by the maximum) centralities distribution for the same network is also shown.

to be interpreted with caution, and a case-by-case review to qualitatively determine the quality of clusters obtained is also essential. Related to this issue, we also notice that a fully automated way of using community detection algorithms as a black-box may not necessarily yield the best results (whichever way best is determined), and instead the results obtained from the algorithms can (and may need) be refined with manual interpretation and intervention. For the dolphin network, this would mean agglomerating some of the smaller communities determined by several of the algorithms.

4.6.2 Clustering of the America College Football Network

We carry out a similar experiment based on another network with known ground truth, namely, the "American College football network" [12]. This is an undirected network representing the Division-I football games from Fall 2000. A team is represented as a node, and a game between two teams is represented as a link between two nodes. There were in total 115 teams and 613 games. During regular football season, teams were divided into 12 conferences and authors of [12] label teams to which conferences they belong. If teams were in the same conference, they frequently had games against other teams which are members of the same conference. Therefore, we treat the 12 conferences as the network's ground truth, which consists of 12 clusters.

We first analyze the (unweighted, undirected) network to determine the (asymptotic $t = \infty$) Markov entropic centrality scores. Figure 4.15 (drawn using Gephi [10]) shows the network. Darker the color, higher the relative entropic centrality score. The Figure 4.15 also shows the normalized Markov entropic centrality distribution, where entropic centrality values are normalized by the maximum entropic centrality value. We observe that a majority of nodes have normalized entropic centrality between 0.2 to 0.4. This helps us to identify our clustering parameter S_{HE} which are set of high entropy nodes deemed as centre/border of a cluster (and unsuitable as query nodes). Accordingly, we choose S_{HE} to comprise the top 50/60/70/80% entropic centrality nodes. We applied our proposed clustering algorithm (undirected variation) and results obtained had F-scores against ground truth as 0.273, 0.406, 0.409, and 0.517 respectively.

Considering that the American College football network is not induced by any flows, our proposed clustering still produces reasonable clusters (with F-score 0.517) as shown in Figure 4.16 for parameter S_{HE} comprising the top 80% entropic centrality nodes. We stop at first iteration since our clustering techniques are bottom-up approach which means that second iteration will produce fewer number of clusters. Based on the ground truth (bottom right graph in Figure 4.16), we notice that the reason the F-score is not very good with our algorithm is that, in fact, in an automated manner, our algorithm coalesced several of the ground truth communities to create larger communities. Recall the discussion in the previous subsection (in which, incidentally, our algorithm performed well on its own, but the other popular algorithms had in fact created too many clusters, which is related to the problem we encounter here (namely, what is a meaningful size of cluster, and when to stop further partitioning or agglomerating them?), though in this instance, it is the reverse issue of too few clusters). By extracting (the largest three) subgraphs of these larger communities and re-applying our clustering techniques on each of these subgraphs (again with parameter S_{HE} consisting of the top 80% entropic centrality nodes), we obtained a new group of clusters, shown on the top right of Figure 4.16, with a significantly improved F-score of 0.811. The overall computation time including all the steps (excluding the time for manual intervention) was a total of 0.221 seconds.

We compare this with results obtained with InfoMap [8] (F-score of 0.904 and total time of 0.013 seconds), Louvain [7] (F-score of 0.823 and total time of 0.002 seconds), and label propagation [9] (F-score of 0.796 and total time of 0.001

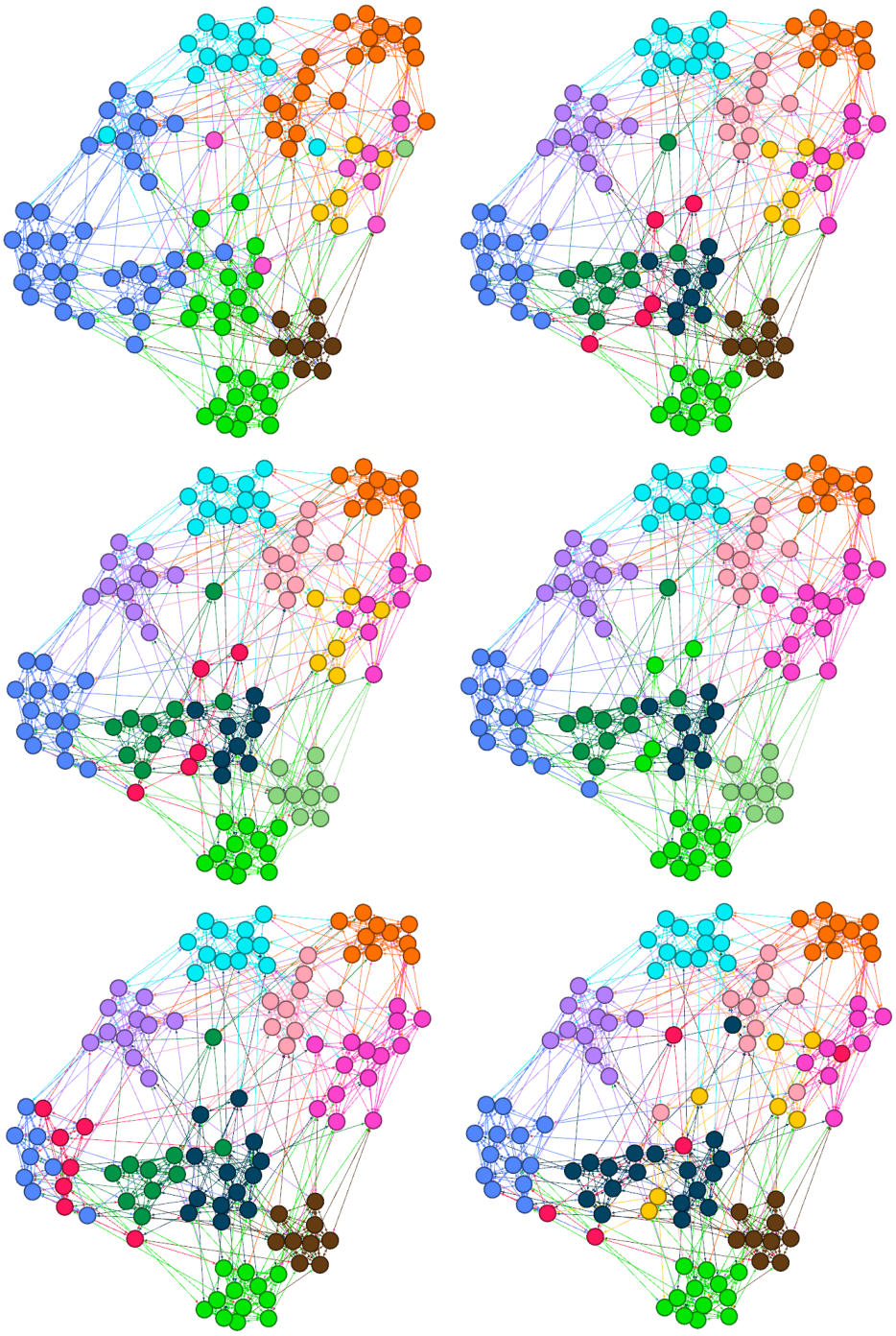


FIGURE 4.16: We show the clusterings of the American College football network [12]. The plots are organized as follows. From **left to right** and **top to bottom**: our proposed technique (one iteration), our proposed technique (reapplied on manually chosen (3 largest) subgraphs obtained from the first iteration), InfoMap [7], Louvain [8], label propagation [9] and the ground truth [12]. These plots are drawn using Gephi [10].

seconds) as shown in Figure 4.16 along with ground truth. In the ground truth [12], a cluster with yellow color and another cluster with crimson color spread their members out over the network. Considering this anomalous 'ground truth', ours as well as other clustering techniques produce very good results. InfoMap has the highest F-score, Louvain has similar F-score to our clustering technique (reapplied manually on suitable subgraphs), and label propagation has the lowest F-score. This was again a network which was not induced by any flow. Our experiments indicate that nevertheless, our algorithm is able to produce meaningful clusters.

Improving the quality of clusters obtained, including by automation of the termination criteria of our base algorithm, as well as for choosing subgraphs to be reclustered, is something we want to address in the immediate future. We also will like to emphasize that the objective of our implementation was to translate the ideas into a working system, however, our implementation is not optimized. In contrast, the libraries we used to evaluate the other algorithms are all mature industry ready implementations, resulting from efforts of many years and persons, optimized and refined both in terms of their computational performance as well as to deal with several of the nuances that we do not address (but which are mainly implementation issues, and can be achieved by following industry best practices). Improving our implementation accordingly, and also exploring how to distribute and parallelize some of the computations to achieve better scalability are also part of our future plan.

4.7 Application to Bitcoin Network Analytics

Having validated our clustering algorithm, we apply it on another dataset, specifically a Bitcoin network constructed around the fund movement from Bitcoin wallet addresses used for WannaCry ransomware based extortion. We make this choice because some information about this event is already known, which can be used to further validate our approach, and it also helps us illustrate how a flow based clustering algorithm may be used for Bitcoin network analysis. After 12 weeks since the WannaCry ransomware attack, on 3rd August 2017, it was reported[6] that the perpetrators moved out $140,000 from the three initial Bitcoin (wallet) addresses which were tied to WannaCry ransomware, listed as in-addresses in Table 4.5. The WannaCry ransomware attack was a May 2017 cyber-attack

[6]https://qz.com/1045270/wannacry-update-the-hackers-behind-ransomware-attack-finally-cashed-out-about-140000-in-bitcoin/

in-address	out-address	BTC
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn	14Y8rfeRAcZkGqG451UGk1epq5zw3dVQif	7.05953352
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn	1Q8maVpVNAZbPiavySQz9JaiwsfhbT9vBz 18gsrbQsTY7HzYVZEbvtVBfhywpQk6No2Q	7.32042682 0.01511375
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw	1FQQ86tMuvhQ4Ruyggbb8j7iaNfUZ69gpY 1JC41YHmjKEcW1rLH6pmMWEFHkoNwSmhnC	8.71529348 0.01227173
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw	16dfTuSx4f78eQ81PzTgBtBDyZ7QhNZ8Vy	9.02796322
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94	1H68h8qsVkMUgY8khcdFpbHV22cCnC74dk 1ARirZgU4q61sSjVK2iB8BEYC5w2B8ZnE9	9.56285137 0.10287428
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94	1M1CfXLynR6vqbjwTqSiiLRVDQZEXHHJbb	10.05800019

TABLE 4.5: List of transactions which perpetrators moved Bitcoins out from known WannaCry's wallet addresses (as input addresses), on 3rd August 2017.

which was caused by a malicious software, specifically WannaCry ransomware cryptoworm, infecting computers worldwide. The infected computers' data was encrypted and owners were asked to pay $300 to $600 for the decryption key. The perpetrators used Bitcoin (specific addresses were provided in the ransom demand) to collect the ransom payments from victims.

Our objective is to use our clustering algorithm to identify list of suspect (wallet) addresses that might be further used by perpetrators to obscure the trail of ransom money. In Table 4.5, we report 9 addresses as immediate outputs which received fund from WannaCrys (wallet) address. We start by choosing *1M1CfXLynR6vqbjwTqSiiLRVDQZEXHHJbb* as an initial suspect address. Note that, one may use any/all of these 9 addresses as an initial suspect address(es). We report our clustering result on a directed address subgraph containing nodes within a distance of 4 (in an undirected variant of the address graph around the node of interest, as obtained from the transactions for the period of 10 July- 24 August 2017). This graph is illustrated in Figure 4.17. This subgraph has 19658 nodes, and 34211 directed edges.

We show the immediate clustering results from third iteration of agglomeration of our clustering algorithm on the unweighted rendition of the subgraph of interest. Curiously, Figure 4.17 shows two clearly separated clusters comprising a red cluster and a blue cluster as highlighted with a blue circle in the figure. We choose to study the clustering result from third iteration, since the two (red and blue) clusters merge after that.

In Figure 4.17, the red cluster covers most of the subgraph, and is centred around (1) a wallet address *1ETWkyQUY9nRpVMyGwha4vRhwKgMbomMQe* which was involved in 3415 transactions. Among said transactions, there are 1877 transactions which have more than 2 inputs (47 inputs on an average) but has less than 3 outputs; 253 transactions which have less than 3 inputs and more than

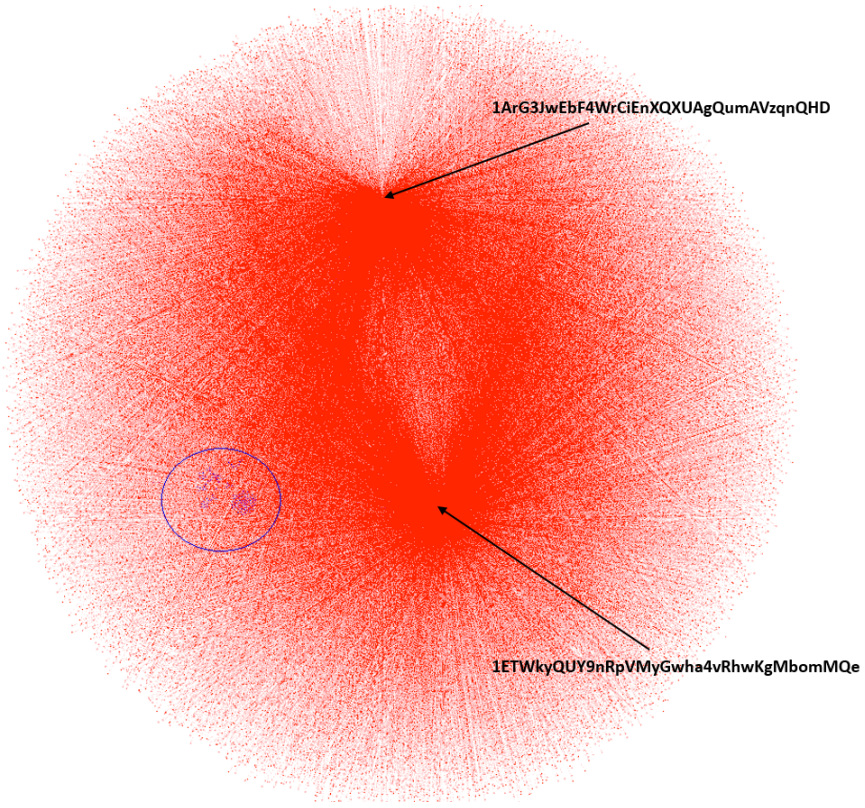


FIGURE 4.17: Clustering of a Bitcoin subgraph involved in the WannaCry ransom fund movement after incident 12 weeks.

2 outputs (34 outputs on average), and 92 transactions which have more than 2 inputs and 2 outputs. The average time to spend Bitcoins after receiving Bitcoins is around two hours. Moreover, its last transaction was on October 2017 and its first transaction was on September 2016. Based on these information (and particularly given that it has been active long before the extortion campaign), it is highly likely that this address was used as a mixing service; (2) a wallet address *1ArG3JwEbF4WrCiEnXQXUAgQumAVzqnQHD* which was involved in almost one hundred thousand transactions and is ranked among top 100 of popular addresses[7]. This popular address contains almost ninety thousand transactions which have less than 3 inputs and 3 outputs, almost 9200 transactions which comprise more than 2 inputs (23 inputs on average) and less than 3 outputs, around 1000 transactions with less than 3 inputs and more than 2 outputs (31 outputs on average). Also, it takes around 2 weeks to spend after receiving Bitcoin, on average. Furthermore, its last transaction was on September 2018 and its first transaction was on May 2016. Accordingly, we suspect that this popular address may be used by Money

[7]https://www.blockchain.com/btc/popular-addresses. Accessed on 02 December 2018.

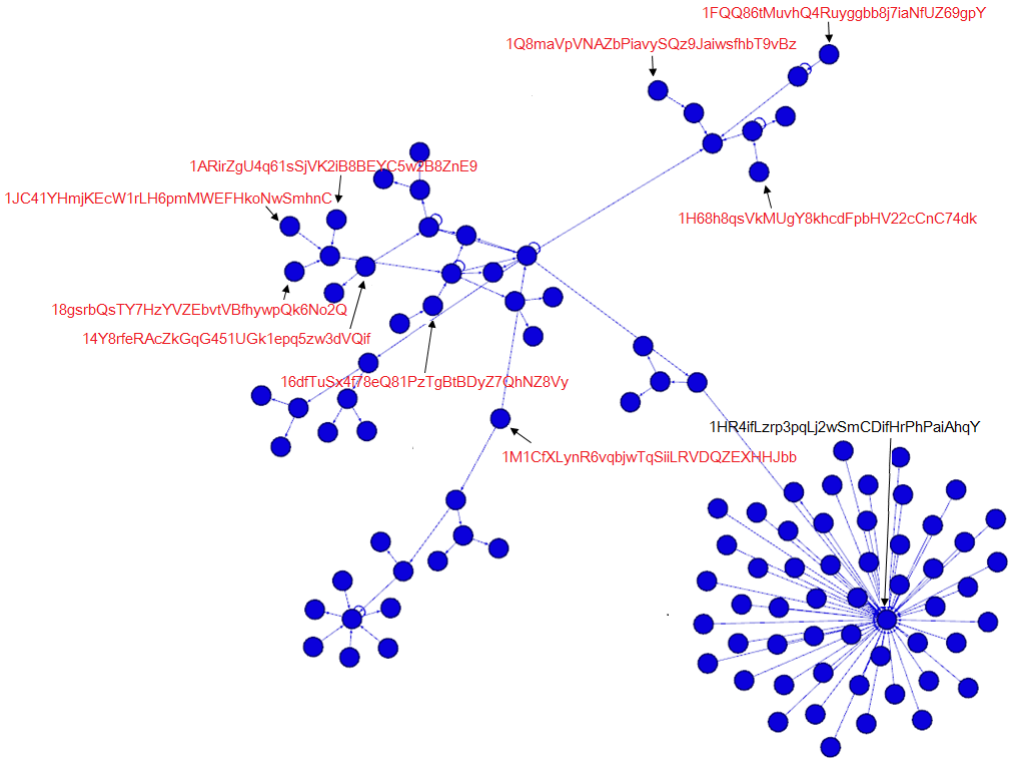


FIGURE 4.18: Zoom into the above clustering, red labelled nodes are immediate outputs of transactions, in Table 4.5, which contains addresses tied to WannaCry ransomware as inputs; a black labelled node is suspect address involved in mixing transactions.

exchange service, or a more sophisticated manner of mixing where money for one user is not paid back to other addresses of the same user, but instead payment is likely delinked across transactions. Unfortunately, we could not find any side information regarding either of these two addresses to validate these guesses.

We next zoom into the blue cluster as illustrated in Figure 4.18. The blue cluster comprises of 106 addresses which include our initial suspect address as well as all the immediate output addresses listed in Table 4.5. The latter are indicated in the figure. Their presence in our cluster, in absence of any other ground truth, still acts as a sanity check if not validation of the clustering algorithm. On the bottom-right, we observe that there is a small community centred around a wallet address *1HR4ifLzrp3pqLj2wSmCDifHrPhPaiAhqY* which is one of the addresses that flows are likely to go to, since it receives Bitcoins from several transactions, almost half of its transactions contain more than 2 inputs and 2 outputs. Its first transaction was on April 2017, and last transaction was on February 2018 and again this address is likely to be one from a mixing service which perpetrators may have used to obscure the money trail. Nevertheless, the community of 106 addresses,

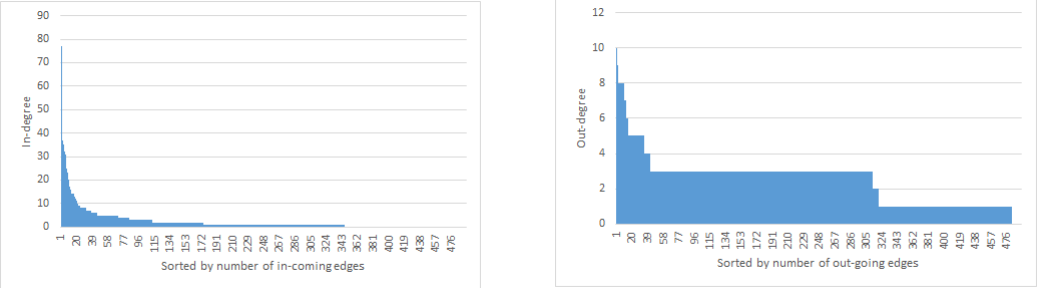


FIGURE 4.19: In-degree (left) and out-degree (right) distribution of the synthetic network.

which is detected from our clustering based on confined circulation of money, helps us discover a new shortlist of suspect addresses for further investigation, using a myriad of possible mechanisms [30, 31, 47]. Later in Chapter 6, we will elaborate in greater detail our own approach (as well as relevant related works) on how to investigate further based on such an initial list of suspect nodes of interest.

4.7.1 Validation with Synthetic Network

In order to explore the applicability of the clustering algorithm in the context of Bitcoin forensics, we may want to determine if it is robust in the presence of multiple Bitcoin mixing events. Due to lack of real world data with ground truth for this, we create a synthetic dataset for better control over the experiments. We consider three types of transactions: (1) transactions with single input and two outputs, where one of the outputs is for change (i.e., belonging to the same owner of the input); (2) two inputs and two outputs transaction, which has both inputs belonging to the same owner and one of the outputs is used for collecting change (i.e., belonging to the same owner of inputs); and (3) multi-input multi-output transactions, where the inputs and outputs belong to multiple parties, and indicate mixing operations.

We first generate a small set of transactions which consists of only the single input and two outputs transactions, and two inputs and two outputs transactions, such that the corresponding (wallet) address network has statistical properties which have been observed in real-world Bitcoin networks [47, 119], particularly, power law in/out degrees distribution. Specifically, we apply the Barabasi-Albert preferential attachment model [167] to obtain the power-law degree distributions. The network so formed comprises 469 nodes and 1136 links (without any mixing

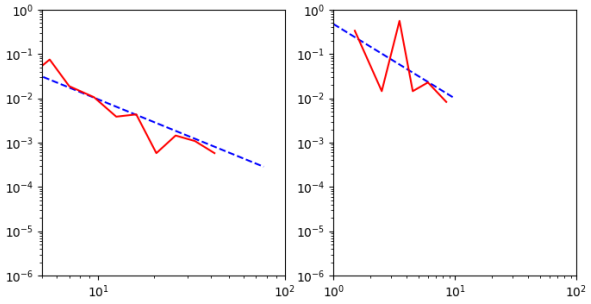
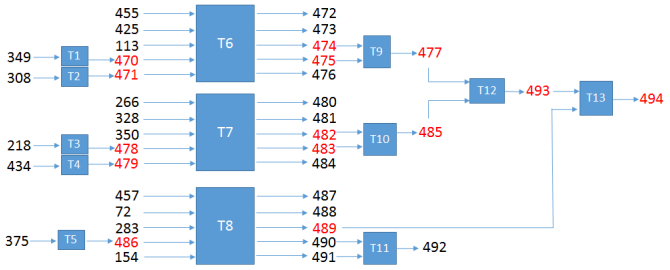


FIGURE 4.20: Fitting a power law for synthetic network: on the left, the in-degree distribution with $\alpha_{in} = 1.7033$, $x^{in}_{min} = 1$, and on the right, the out-degree distribution $\alpha_{out} = 1.6807$, $x^{out}_{min} = 1$.

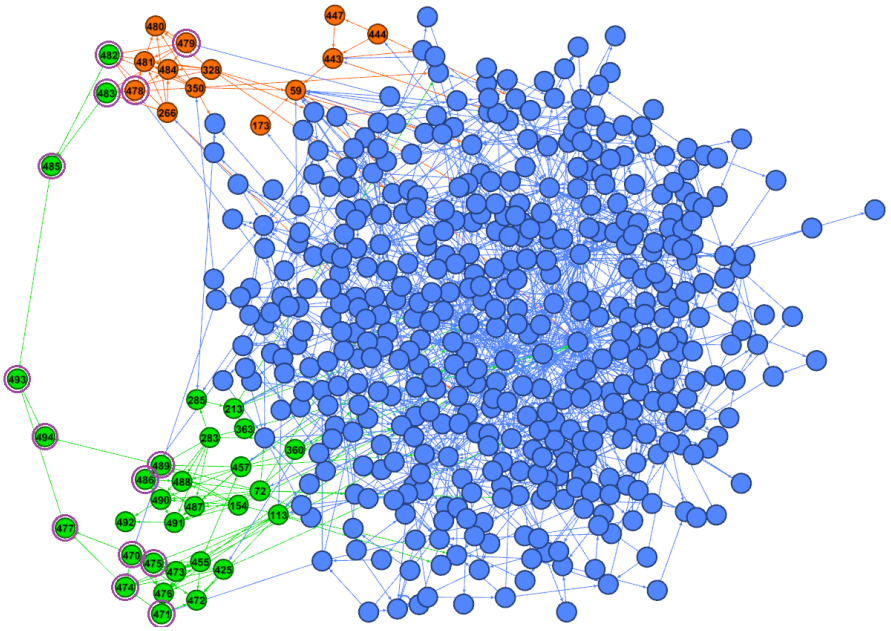
transactions). After that, the nodes and edges corresponding to the mixing transactions are added to the synthetic address graph, resulting in a graph with 494 nodes and 1226 edges, with in/out-degree distributions as shown in Figure 4.19. To verify whether our synthetic network still holds the power law in/out degree distribution, we report on how the distributions fit (using [168]) in Figure 4.20. The algorithm in [168] looks for either the best fit, or the best fit given a choice of x_{min}. We show the results obtained by fixing x_{min} to be 1, since we are more interested in the overall behaviour of the distributions rather than estimating the parameters for best fit.

Since the network is synthetically generated, we determine and keep a record of the relations between (wallet) addresses and users as ground truth, so that we may subsequently compare our cluster results to this ground truth. In constructing the multi-input multi-output transactions, we considered a perpetrator who collects money from five users uniformly at random in transactions $T1, T2, T3, T4$ and $T5$, then obfuscates the flow using mixing operations which are multi-input multi-output transactions $T6, T7$ and $T8$, and gathering back the money to a new wallet address with transactions $T9, T10, T12$ and $T13$ as illustrated in Figure 4.21a, where a red index represents addresses which belong to the perpetrator.

We next apply our proposed clustering algorithm on the synthetic network described above. Figure 4.21b shows our resulting clusters. We observe that our clustering captures specific addresses which acted in a coordinated manner, as shown in orange cluster and green clusters. Indices of labelled nodes in orange cluster match indices in $T7$ (see Figure 4.21a), similarly, indices of labelled nodes in blue clusters match indices in $T6$ and $T8$ (see Figure 4.21a). This partially verifies that our clustering approach captures probabilistic flows of Bitcoins despite of mixing in the directed address network. Based on visual inspection, we see



(A) Transactions which involve (wallet) addresses of interest as shown as red index and blue blocks represent Bitcoin transactions in our synthetic network.



(B) Clustering of our synthetic network: purple circles highlight nodes of interest.

FIGURE 4.21: Top: Transactions which involve (wallet) addresses of interest on the top; and Bottom: the clustered obtained with our algorithm after stage 2.

that the approach provides high recall but moderate precision in capturing nodes actually associated with the perpetrator, as highlighted with an outer purple circle in Figure 4.21b, i.e., even though the green and orange clusters comprise addresses from other normal users, all the addresses belonging to the perpetrator belong to these two clusters. Hence, one may want to use the orange and green clusters as shortlist of suspected addresses for further investigation. Essentially, this simple toy example with known ground truth helps validate the inferences (both the strengths and limitations of the clustering algorithm in the context of Bitcoin analysis) that we drew with the real world data.

4.8 Summary

The study of the Bitcoin network originally motivated our investigations of centrality and clustering for weighted directed graphs, capturing characteristics of Bitcoin flow circulation among nodes. Most of our study thus naturally used Bitcoin subgraphs to expose the concepts and validate the efficacy of our centrality computation and clustering algorithms, and we propose Bitcoin forensics as an immediate possible application. We expect our framework to apply to directed weighted graphs in general, and in particular to networks induced by financial activities.

We also apply our clustering algorithm to networks which are not flow based (we could not locate any graph data set fitting our flow model, but with ground truth). Specifically, we use the dolphin network[11] and the American College football network [12]. These are networks with some sort of ground truth, which we use to quantify the quality of clustering results obtained with our approach, and compare it against other popular community detection and clustering algorithms. Note that even though these networks are not based on flows, yet our algorithm yielded very good results. We furthermore validated our clustering with simple synthetic data where ground-truth is 'known'. We also demonstrated how the flow based clustering algorithm can be applied to Bitcoin networks to identify clusters of interest, which can then be further studied. In that context, the proposed clustering algorithm can be viewed as a macroscopic tool, which can help identify and isolate subgraphs of interest for further investigations.

Generating more complex dataset (comprising all varieties of Bitcoin transactions [153]) with ground truth for more rigorous validation will be interesting and desirable. However, there is not yet enough work in the literature on how to generate realistic but synthetic Bitcoin network traces. The main challenge there is the generation of synthetic networks that do preserve a multitude of real world characteristics at scale, and it is a research problem (orthogonal to the focus of this thesis in general, and this chapter in particular) in its own right.

Chapter 5

Renyi Entropy Based Graph Clustering

In the previous chapter, we explored techniques to cluster all combinations (un/directed and un/weighted) of graph with an entropy-based approach that looks at the distribution of flows over the network. We next want to explore another (Renyi [169]) entropy-based clustering approach, which does not consider flows, but instead defines a distance metric. A Renyi entropy based clustering approach was proposed in the context of image clustering [43], and, in this chapter we essentially investigate whether we can translate the ideas in the context of graph clustering.

The original problem formulation [43] we build upon can be summarized as follows. Given N d-dimensional points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in $\mathbb{R}^d$, the problem of data clustering consists of assigning a label $l \in \{1, \ldots, C\}$ to each $\mathbf{x}_i$. The points with the same label l then form a cluster, and the goal is to put in the same cluster points which are "similar", where similar means close with respect to a given measure which describes the considered problem. It is known [170] that one possible information theoretic formulation for the clustering problem is to suppose that each cluster corresponds to samples from a given probability distribution, and separating the clusters becomes maximizing the distance among distributions. There are standard information theoretic measures that could be considered for this. For example, the Kullback-Leibler divergence $D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx$ indeed characterizes how one probability distribution diverges from another expected probability distribution, and the Bhattacharya distance $D_B(p, q) = -\ln \int \sqrt{p(x)q(x)} dx$ measures the similarity of two probability distributions. One issue in using them for clustering though, is that they need to be estimated without making too restrictive

assumptions about the data distribution. The mutual information $I(X;Y)$ between two random variables X, Y quantifies the amount of information obtained about one random variable given the other, this is another information theoretic measure which has been well studied for clustering, see e.g. [171–175]. Accordingly, in [43], an information theoretic measure was proposed for clustering, based on Renyi quadratic entropy. The entropy-based clustering produces clusters that have in-between cluster entropy the least among all cluster combinations which can arguably be found by exhaustive search, which however is not scalable.

A main challenge in actuating the idea is the search or construction of a suitable solution - for the former (search of a solution) we explore a simulated annealing based approach, and for the latter (construction of a solution), we explore a hierarchical clustering approach. Another challenge in applying the Renyi-entropy based clustering pertains to how distances are to be defined and interpreted in a graph, and particularly in the context of directed graphs (where the symmetry property of a distance metric is broken if a typical measure like shortest path is applied). We propose a metric derived by modification of the Jaccard distance based on nodes' neighbours, and our experiments demonstrate its suitability. The overall contribution of this chapter is thus to explore and design several mechanisms to translate the idea of Renyi entropy for clustering in the context of graph clustering, for both undirected and directed graphs.

5.1 The Notion of Entropy-based Measure for Graph Clustering

In the context of image clustering, among other approaches, a Renyi entropy based clustering mechanism has been proposed [43], which is the inspiration of the work presented in this chapter. We thus first discuss the general idea of Renyi entropy, and how it is used for the purpose of clustering. Then we explore how the idea can be adapted in the context of graph clustering.

5.1.1 A Sample-based Estimator for Renyi Entropy

The quadratic Renyi entropy [169] of a vector $\mathbf{x} \in \mathbb{R}^d$ is

$$H_2(p) = -\ln \int p^2(\mathbf{x}) d\mathbf{x}$$

where $p(\mathbf{x})$ is the pdf (probability distribution function) of $\mathbf{x}$. When $p(\mathbf{x})$ is unknown, but samples are available, the pdf can be replaced by a sample-based estimator. The probability P that a vector with pdf $p(\mathbf{x})$ falls in a region $\mathcal{R}$ of $\mathbb{R}^d$ is given by

$$P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x}.$$

If $\mathcal{R}$ is small enough to ensure that $p(\mathbf{x})$ does not vary too much within it, we can approximate P by

$$P \approx p(\mathbf{x}) \int_{\mathcal{R}} d\mathbf{x} \tag{5.1}$$

where $\int_{\mathcal{R}} d\mathbf{x}$ is the volume of $\mathcal{R}$. Now if we have N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ which are independently drawn according to $p(\mathbf{x})$, and there are say k out of the N samples falling with $\mathcal{R}$, then

$$P = \tfrac{k}{N}. \tag{5.2}$$

Combining (5.1) and (5.2) yields

$$\hat{p}(\mathbf{x}) = \frac{k/N}{\int_{\mathcal{R}} d\mathbf{x}}$$

as an obvious estimate for $p(\mathbf{x})$. If now $\mathcal{R}$ is a hypercube in $\mathbb{R}^d$ with edge length h centered at $\mathbf{x}$, then $\int_{\mathcal{R}} d\mathbf{x} = h^d$, and we can introduce the following window function W which indicates whether $\mathbf{x}_i$ is inside $\mathcal{R}$:

$$W_h(\mathbf{x}_i - \mathbf{x}) = \begin{cases} 1 & \text{if } \frac{|x_j - x_{ij}|}{h} \leq \frac{1}{2},\ j = 1, \ldots, d \\ 0 & \text{else.} \end{cases}$$

The total number k of samples falling in $\mathcal{R}$ is thus given by

$$k = \sum_{i=1}^{N} W_h(\mathbf{x} - \mathbf{x}_i)$$

and the Parzen window estimator [176] is given by

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h^d} W_h(\mathbf{x} - \mathbf{x}_i).$$

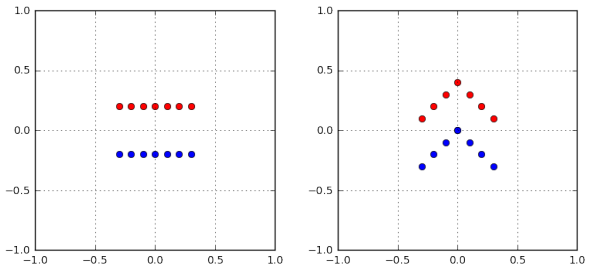


FIGURE 5.1: Two shapes in the plane being distinguished by the considered between-cluster evaluation function, with $\sigma^2 = 0.01$.

A typical choice for the window function W_h is the Gaussian function given by $W_\sigma(\mathbf{x} - \mathbf{x}_i) = \frac{1}{\sqrt{2\pi^d}} \exp\left(-\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{x}_i||^2\right)$, where the covariance matrix Σ is simplified to be $\Sigma = \sigma^2 \mathbf{I}_d$ and $h = \sigma$ is a scale parameter, for which

$$\begin{aligned} \hat{p}(\mathbf{x}) &= \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma^d} W_h(\mathbf{x} - \mathbf{x}_i) \\ &= \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma^d \sqrt{2\pi^d}} \exp\left(-\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{x}_i||^2\right). \end{aligned}$$

The product of two such Gaussian distributions is [177, 8.1.8]

$$\begin{aligned} & \tfrac{1}{\sigma^d\sqrt{2\pi^d}} \exp(-\tfrac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma^2}) \tfrac{1}{\sigma^d\sqrt{2\pi^d}} \exp(-\tfrac{||\mathbf{x}-\mathbf{x}_j||^2}{2\sigma^2}) \\ = & \tfrac{1}{\sqrt{(2\sigma^2)^d(2\pi)^d}} \exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2}) \mathcal{N}_{\mathbf{x}}(\tfrac{1}{2}(\mathbf{x}_i + \mathbf{x}_j), \tfrac{\sigma^2}{2}\mathbf{I}_d) \end{aligned}$$

where $\mathcal{N}_{\mathbf{x}}(\frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j), \frac{\sigma^2}{2}\mathbf{I}_d)$ denotes a multivariate Gaussian distribution with mean $\frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$ and covariance matrix $\frac{\sigma^2}{2}\mathbf{I}_d$. Thus taking the integral over the product simplifies to

$$\tfrac{1}{\sqrt{(2\sigma^2)^d(2\pi)^d}} \exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2})$$

and a Renyi entropy sample-based estimator $\hat{H}_2(p)$ is then

$$\begin{aligned} & -\ln \int \tfrac{1}{N} \sum_{i=1}^{N} \tfrac{1}{\sigma^d} W_\sigma(\mathbf{x} - \mathbf{x}_i) \tfrac{1}{N} \sum_{j=1}^{N} \tfrac{1}{\sigma^d} W_\sigma(\mathbf{x} - \mathbf{x}_j) d\mathbf{x} \\ = & -\ln \frac{1}{N^2} \sum_{i,j=1}^{N} \tfrac{1}{(2\sigma)^d\sqrt{(2\pi)^d}} \exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2}) = \hat{H}_2(p). \end{aligned} \tag{5.3}$$

5.1.2 Entropy Measure for Clustering

The quantity $\hat{H}_2(p)$ in (5.3) is interpreted as a *within-cluster entropy* [178], since if we consider a single cluster, associated to a single pdf $p(\mathbf{x})$, $\hat{H}_2(p)$ computes an estimate of its entropy.

However, if we use (5.3) but by summing over two different clusters (there is a probability distribution $p_1(\mathbf{x}), p_2(\mathbf{x})$ associated to each cluster), then as proposed in [43, 178], we get a *between-cluster entropy*

$$D(\hat{p}_1, \hat{p}_2) = -\ln \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \frac{1}{(2\sigma)^d \sqrt{(2\pi)^d}} \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{4\sigma^2})$$

as a cluster evaluation function that estimates the distance between two clusters, since the between-cluster entropy estimates the distance between sample-based estimation of their distributions. This also gives an estimator of the "cross Renyi entropy" $-\ln \int p_1(\mathbf{x}) p_2(\mathbf{x}) d\mathbf{x}$ (which is close to, but differs from the Bhattacharya distance [179]).

This between-cluster entropy function was proposed in [43] for image processing applications. Figure 5.1 illustrates how it helps separating shapes in the plane, where an exhaustive search is done to test all possible labellings for two clusters.

In the case of several clusters with respective pdfs $p_1, \ldots, p_C$, we have the generalized *between-cluster entropy*

$$D(\hat{p}_1, \ldots, \hat{p}_C) = -\ln \frac{1}{N^2} \sum_{i,j=1}^{N} \frac{\delta_{ij}}{(2\sigma)^d \sqrt{(2\pi)^d}} \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{4\sigma^2})$$

where $\delta_{ij} = 0$ if both $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster and 1 otherwise.

5.1.3 Entropy-based Measure for Graph Clustering

The between-cluster evaluation function depends on the distance $||\mathbf{x}_i - \mathbf{x}_j||^2$ between two data points. In the case of graph clustering, given an undirected graph $G = (V, E)$ with set V of nodes and set E of edges, data points become the nodes, and the distance between two nodes depends on E. Typically, the distance between two nodes in a graph is the number of edges in a shortest path, and we assume that the graph is connected (otherwise, we consider the connected components of the graph separately). If the graph is weighted, then the weight of the edges can be taken into account, but we will consider the case where the edges have no weight,

thus the distance $d(u, v)$ between u and v is the length of a shortest path between u and v.

5.2 Entropy-based Graph Clustering - A Simulated Annealing Approach

An exhaustive search across all possible combinations of clustering for a graph with n nodes, even for 2 clusters, would require the enumeration of $2^{n-1} - 1$ combinations, discounting for the equivalent clusters obtained by interchanging all the labels and the two single cluster cases, a solution obviously impractical. We consider a simulated annealing heuristic [180] instead. Simulated annealing is a widely used randomized algorithm technique, which has also been utilized in various graph algorithms, e.g., determine graph bisection [181], graph partitioning [182, 183], thickness [184], coloring [185] and link prediction by deanonymization [186], to name a few. Even though our work and the works on graph partitioning [182, 183] both use simulated annealing techniques and address related problems, the problems of partitioning and clustering are distinct, and so are the underlying mechanisms that have been applied: [182, 183] focus on minimum cut and define their own energy evaluation criteria, while our proposed technique focus on entropy-based distance evaluation criteria. The details of our approach are contained in Algorithm 3.

Algorithm 3 assumes a predefined number of clusters, that is, of distinct labels, and mutually disjoint clusters (not fuzzy ones), meaning that at any time point, a unique label is assigned to any node to indicate which cluster it belongs to. The guessed clustering at any step t of the algorithm is denoted by l^t. We initialize the labels uniformly at random (to obtain the labelling l^0). Then, at every step, we pick a node uniformly at random and alter its label to any of the other possible labels uniformly at random, to obtain a new candidate clustering corresponding to l^{new}. If the new clustering decreases the between-cluster evaluation function, then we found a better clustering, chosen for the next round. However, even if the new clustering increases the evaluation function, thus yielding a worse clustering, it may still be considered as the new candidate - but with a probability that decays exponentially with the difference $\Delta = D(l^{new}) - D(l^t)$ of the entropy measures. The rate of decay is parameterized (by β). The rationale behind allowing such "bad choices" occasionally is at the heart of the simulated annealing heuristic to

Algorithm 3 Simulated annealing based graph clustering

for every node in the graph **do** ▷ Initialization
  Choose a label uniformly at random ▷ Create l^0
  ▷ Number of label choices is predetermined by the number of clusters being sought
for chosen # of times **do** ▷ Annealing
  ▷ other termination criteria may be used instead
  ▷ At iteration $t \geq 0$, make a transition from the overall labelling l^t at t to the labelling l^{t+1} at $t+1$ as follows:
  Select a node $v \in V$ uniformly at random
  Assign v a different label chosen uniformly at random, to obtain new overall labelling l^{new}
  Assign $\Delta = D(l^{new}) - D(l^t)$
  ▷ $D(.)$ is the between-cluster entropy
  if $\Delta \leq 0$ **then**
    Set $l^{t+1} = l^{new}$ ▷ Accept the new labelling
  else ▷ The case when $\Delta > 0$
    With probability $e^{-\beta\Delta}$: Set $l^{t+1} = l^{new}$
    With probability $1 - e^{-\beta\Delta}$: Set $l^{t+1} = l^t$
    ▷ parameter β determines the rate of decay
for chosen # of times **do** ▷ Optional: post-processing
  $\forall$ node X s.t. X is in a cluster boundary:
  Move X to best neighbor cluster if it reduces D(.)
  $\forall$ node X s.t. X has multiple neighbors:
  Assign all neighbors the same cluster as X if doing so reduces D(.)

extricate the search from being trapped in local extrema. We will discuss the optional post-processing step afterwards, together with the results shown in Figure 5.6.

5.2.1 Numerical Results

We next apply entropy-based clustering on undirected unweighted networks. We start by analyzing effectiveness of clustering result on graphs. For small graphs, specifically, a 7 nodes lollipop graph and a 12 nodes ladder graph, we consider entropy-based clustering by using exhaustive search approach. We next validate the sanity of our simulated annealing algorithm 3 on some simple synthetic data. We then run the algorithm on real world graphs, particularly the Karate club network [13], and a 178 nodes Bitcoin network [21].

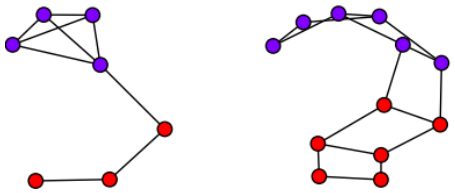


FIGURE 5.2: Two clusters found on a lollipop graph (on the left) and on a ladder graph (on the right), $\sigma^2 = 0.01$.

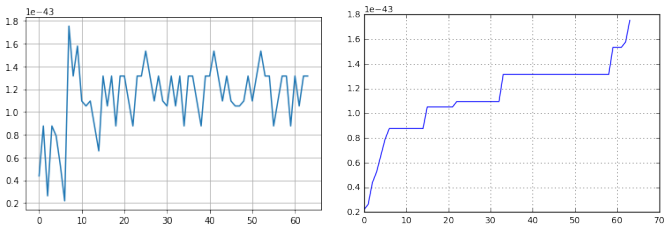


FIGURE 5.3: Fluctuation of the between-cluster evaluation function, for the lollipop graph of Figure 5.2: on the left, the values taken by the between-cluster evaluation function as a function of the labellings in an exhaustive search; on the right, the same values (but) sorted in increasing order.

5.2.1.1 An Exhaustive Search Approach

Figure 5.2 shows two small graphs on which the between-cluster evaluation function has been tried for two clusters. Numerical computations are done using Networkx [161], and an exhaustive search is done to test all possible node labellings for two clusters. Figure 5.3 illustrates the behavior of the between-cluster evaluation function during an exhaustive search for the lollipop graph with 7 nodes shown on the left of Figure 5.2.

5.2.1.2 A Simulated Annealing Approach

We first tested Algorithm 3 (solely the simulated annealing, without the (optional) post-processing step) with some synthetic graphs. In Figure 5.4 we demonstrate multiple (four) clusters obtained in a network which comprises four lollipop graphs interconnected with a star at the center. Since the simulated annealing approach is a randomized algorithm which does not carry out an exhaustive search, the results obtained from different runs of the algorithm can be different, and they may diverge from the optimal. We notice this in the left graph of Figure 5.4, where the obtained result is visibly sub-optimal. This is a known and inherent weakness of simulated annealing, and one usual remedy is to run the algorithm multiple times, with distinct initializations, and pick the best result from multiple runs of

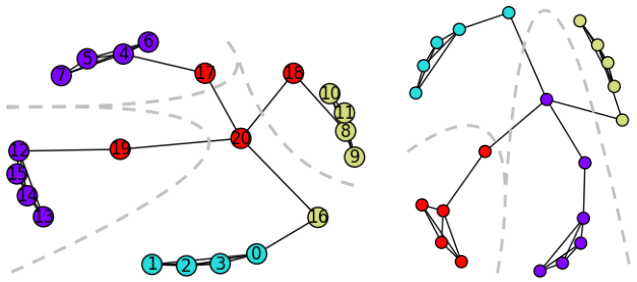


FIGURE 5.4: A clustering with 4 labels: on the left, an interesting (though not optimal) clustering found by the simulated annealing algorithm; on the right, the best clustering found (so far). Specific nodes (may) have different colorings in the two representations. Given the symmetry of the network, the clustering shown on the right is the expected one, given that the central node could have belonged to any of the four clusters. We indicate this expected clustering with dotted lines in the left figure.

the algorithm. The best result we thus obtained is shown on the right of Figure 5.4.

Having validated the proposed approach with some simple synthetic graphs, we next study it using a small real social network — the Karate club graph [13], because it is a canonically studied graph in the clustering literature [12], with a well understood ground truth. The clusters obtained with our algorithm (solely the simulated annealing, without the (optional) post-processing step) are shown in Figure 5.5 [1]. When compared with several existing clustering techniques (using implementations available in Python through *sklearn.cluster* [165]), including k-means, affinity propagation, spectral and agglomerative clustering – the clusters we obtain by our method agree with the results from these other techniques, but for the node 10. In fact, in different runs of our algorithm, the node 10 fluctuates across the two clusters. We show this particular instance because of three reasons: (i) to emphasize the non-determinism of the randomized approach, (ii) it so happens that as per our between-cluster entropy measure, the other possible clustering is in fact marginally (0.347%) worse, and (iii) most interestingly, node 10 had indeed no strong affiliation with either of the factions [13], though the person had eventually joined the faction represented by the left cluster (in yellow). But this sort of "error" is not surprising. For instance, node 9 had in fact a slight leaning towards the left cluster, and our clustering is "correct" (and even agrees with the results from all the above mentioned clustering algorithms), but node "9" still had finally joined the faction represented by the cluster on the right (green nodes). Quoting [13]:

[1]The code used can be found at `https://github.com/feog/Bitcoin-Analysis`

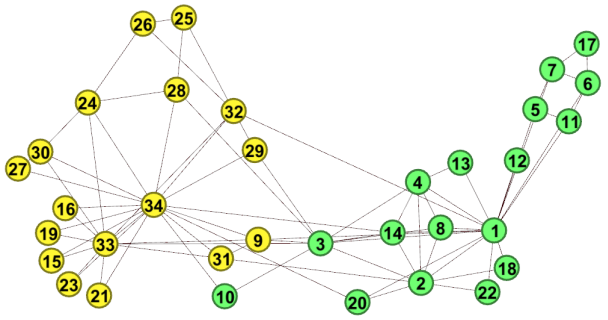


FIGURE 5.5: A clustering of the Karate club graph [13] with two labels.

'*This can be explained by noting that he was only three weeks away from a test for black belt (master status) when the split in the club occurred. Had he joined the officers' club he would have had to give up his rank and begin again in a new style of karate with a white (beginner's) belt, ...*'.

The next issue we wanted to investigate was how the algorithm performs in the 'wild', for an arbitrary graph which may or not even have very well defined clusters to begin with. We have experimented with subgraphs from the Bitcoin transactions network (undirected version), and report our results and insights using a subgraph comprising of 178 nodes [21]. The results for 2 and 5 clustering are shown in Figure 5.6. We also show k-means, Ward's agglomerative (with Euclidean affinity) and spectral clustering.

We ran our simulated annealing algorithm for 2000 generations, and repeated the experiments ten times with distinct random initializations. We report the best result obtained from the ten runs. That is 20000 samples as opposed to 2^{177} samples an exhaustive search would have involved. This is to elaborate on the earlier comment that one can execute the randomized search multiple times, and pick the best result, because an individual run is relatively inexpensive. We used the best result so obtained to further run the post-processing for ten rounds.

In the absence of an established and accepted ground truth (unlike for the Karate club graph case) for the given graph, we discuss the current results qualitatively, and based on the visualization of the clusters. Inspecting the clustering results with simulated annealing, we notice two interesting (and possibly intertwined) issues with our algorithm (we highlight some instances in our plots): There are islands of "sub-clusters" isolated from other nodes supposedly belonging to the same cluster, and embedded inside another cluster. In the extreme case there are isolated nodes. We see such artefacts also from the clustering results from other standard algorithms we tested. Secondly, there are certain regions of the graph,

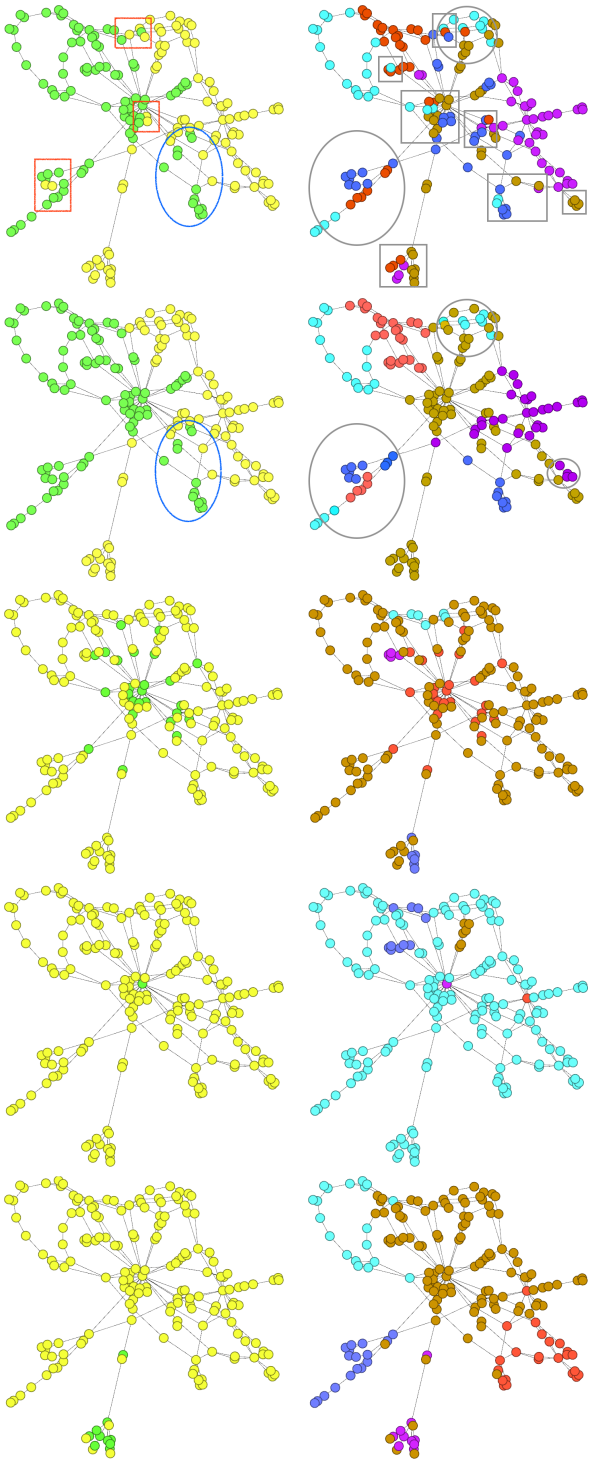


FIGURE 5.6: We show the clusterings of a Bitcoin transactions subgraph of 178 nodes. The plots are organized as follows. On the **left**: 2-clusters; on the **right**: 5-clusters. From **top to bottom**: entropy-based simulated annealing only, simulated annealing with post-processing, k-means, agglomerative (Ward's), spectral clustering algorithms. These plots are drawn using Gephi [10].

where we notice that if these regions were indeed studied in isolation, there would have been clusters. Yet, when studying the overall graph, we may not want to view these "local" artefacts, particularly if we seek a predefined number of clusters. For the former issue (isolated nodes or islands): given that our simulated annealing carries out a random walk over the search space, by changing the labelling of one node at a time, it may take a very long time to chance upon outliers. In fact, while they are very fast in reaching a reasonably good (approximate) solution, such randomized algorithms are in general known to get very slow as they get close to the optimal. For the second issue (localized clustering artefacts): We are at the moment unsure of the behavior of the between-cluster entropy function, but it certainly does not explicitly discern local versus global differences, and hence may indeed be treating such localized artefacts as separate clusters. Another way to look at this is that, there are just more clusters in the graph than what we are searching for with the parameter choice indicated in the algorithm, which can probably be used to dynamically adjust the parameter itself (something we have not explored yet). Accordingly, we propose some (optional stage in Algorithm 3) heuristic post-processing which make localized decisions based on individual nodes' neighborhood. The essential ideas of the heuristics applied are as follows: Explore whether reallocation of nodes with neighbors from other clusters to one of those neighboring clusters improves the entropy measure; for nodes with multiple neighbors, determine whether assigning all its neighbors to the same cluster as itself improves the measure. We do see the current post-processing heuristics improve the quality of results - but some of the previously mentioned artefacts linger (some instances are highlighted in our plots). Overall, for the two clustering case - only our approach (both variants) resulted in meaningful clusters. For the five cluster case, in additional to ours, the spectral clustering also produced reasonable clusters (and our post-processing heuristic would have further improved its results too). We speculate that for the given graph, our entropy measure favors, and accordingly the algorithm discerns more than five clusters. Further exploration of these issues - as well as quantifying the quality of obtained clusters by other metrics are relevant topics for further study in future.

Algorithm 4 Initial Clustering

1: Initialization: $G = (V, E)$, C a number of initial clusters
2: Compute a shortest path between any $u, v \in V$.
3: Choose C nodes uniformly at random, assign one label to each.
▷ Each node is a starting point for a cluster.
4: **while** some nodes are still unlabelled **do**
5: **for** every cluster **do**
6: Find neighbours of nodes in the given cluster.
7: **if** there are unlabelled neighbours (or labelled neighbours with a worse distance to another cluster) **then**
▷ Optional: Using only the first condition enhances speed, but loses the ability to relabel nodes.
8: Add the neighbour whose addition minimizes the within-cluster evaluation function.

5.3 Hierarchical Clustering

In the previous section, we explored whether a Renyi entropy based clustering mechanism, originally designed for clustering images, when applied to graphs, results in meaningful clusters. Having thus validated its applicability, we next investigate how the clustering may be realized in a bottom-up hierarchical manner. While we do not explicitly implement or benchmark our implementation for computational performance or distribution, the rationale for this design choice is its inherent scalability potential. This is because (i) an agglomerative bottom-up algorithm inherently uses local information, and thus breaks the overall problem into many smaller sized problems, and (ii) this is also thus naturally amenable to parallelism and distribution. The other issue we address here is how to handle directed graphs, where typical measures such as shortest distance loses the particular distance metric property of symmetry. We propose a Jaccard-based distance, which could also be used with the simulated annealing approach, but we focus on its use for hierarchical clustering, because of its scaling potential described above.

5.3.1 Undirected Graphs

Given an undirected graph $G = (V, E)$, we use the shortest path length as a distance matrix, and design a hierarchical algorithm that begins with initial clusterings as described in Algorithm 4.

In the initial clustering of G described by Algorithm 4, nodes that will serve as centers of their respective clusters are chosen uniformly at random, after which,

Algorithm 5 Agglomeration

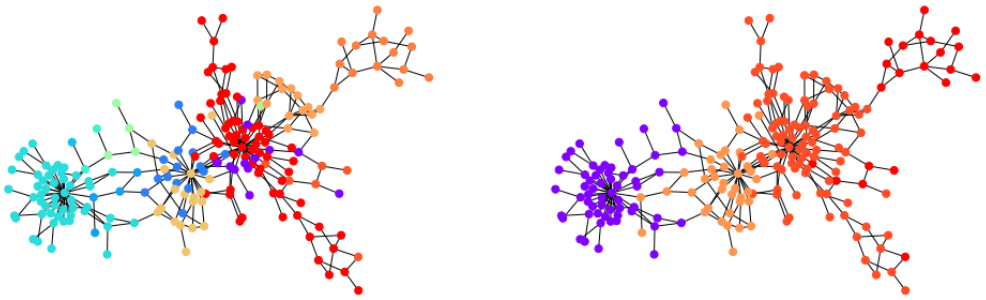
1: Given: $G = (V, E)$, an initial clustering
2: Compute the between-cluster evaluation function between any two initial clusters.
3: **for** for every cluster from the initial clustering in a random order **do**
4: Find the closest initial cluster using the average between-cluster evaluation function.
5: Merge both clusters.

neighbours are added around them, so as to keep the within-cluster evaluation function minimized. The search is performed locally, starting from the center and expanding in the neighbourhood.

The initial clustering algorithm processes until all labels are assigned. It is usually run with the condition "if there are unlabelled neighbours or labelled neighbours with a worse distance to another cluster", but it could also be used with the simplified condition "there are unlabelled neighbours". The latter makes the algorithm terminate fast. However, although more time-consuming, the condition that compares distances from one node to two clusters prevents one node to be assigned to a cluster without the possibility to be re-assigned to a better cluster. The algorithm terminates because at every iteration, either one non-assigned node gets a label, or an already assigned node changes label, but since the condition for changing label is a strict inequality, it is not possible for a node already assigned to infinitely move from one cluster to another. We note that this part of the algorithm contains a non-deterministic element: the initial cluster centers.

Once an initial clustering is done, the next phase consists of an agglomerative clustering, as described in Algorithm 5. A cluster is drawn uniformly at random, the closest cluster (with respect to the in-between evaluation cluster D) is found and both of them are merged. Note that we are here computing an averaging distance (one could choose a maximum or minimum instead). The process is iterated, yielding a bottom-up hierarchical clustering algorithm. There is again one element of randomness, the choice of a cluster to be agglomerated.

An illustration is provided in Figure 5.7. The subgraph comprises 209 nodes and was extracted as a subgraph of the Bitcoin network. There were 21 initial cluster centres (approximately 10% of the total nodes), instantiated with nodes 145, 80, 134, 90, 78, 160, 164, 128, 165, 137, 66, 176, 93, 39, 74, 109, 99, 34, 129, 158, 12. After the initial clustering, there were 13 clusters, after the first round of agglomeration there were 7 clusters, and after the second round of agglomeration

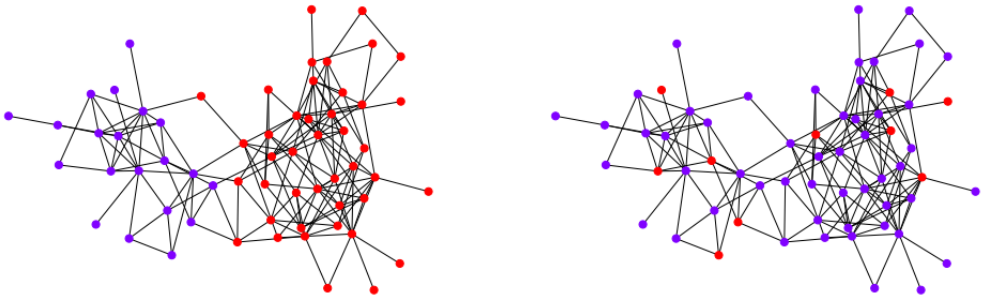


(A) 13 clusters after the initial clustering. (B) 4 clusters after the 2nd round of agglomeration.

FIGURE 5.7: An illustration of the hierarchical graph clustering algorithm on a 209 Bitcoin subgraph.

there were 4 clusters, and finally after third round there were 2 clusters. The clustering results for 13, respectively 4 clusters, are shown. There is no ground truth attached to this graph but a qualitative visual evaluation of the result looks reasonable. Of the 4 clusters shown, three clusters around three high degree nodes, and a component on the right of the graph which has a minimum cut of 1 forms a cluster of its own.

We next apply this clustering algorithm on a network with ground truth that we used in the previous chapter, namely, the dolphin network [11]. For clustering the dolphin network which comprises $N = 62$ nodes, we chose 10 initial cluster centers for every iteration of our experiments, drawn uniformly at random, and look at the second round of agglomeration. This is because after the initial clustering, we have 10 clusters, and we expect roughly half of them, namely between 4 to 6 clusters after the first round of agglomeration, and thus roughly 2 or 3 clusters after another round. Averaging over 1000 iterations of the clustering algorithm, with random cluster centers as described above, we obtain an F-score of ≈ 0.7. Recall from Chapter 4, that other popular community detection algorithms such as InfoMap, Louvain and Label propagation achieved F-scores of 0.545, 0.565 and 0.657 in comparison for the Dolphin network. Our algorithm's performance is thus competitive. Furthermore, if we look at the maximum F-score over these 1000 iterations, we get an F-score of 1 for several cluster center choices. An example of initial configuration that provides such perfect clustering is shown in Figure 5.8b, with nodes 6, 8, 26, 54, 22, 12, 19, 44, 29, 16 highlighted in red. This also indicates that the choice of the cluster centers has a huge influence on the quality of clusters. We tried the heuristic of lowest Markov entropic centrality nodes (though there was



(A) The dolphin network and the ground truth clustering.

(B) An example of initial choice of cluster centers that results in the ground truth.

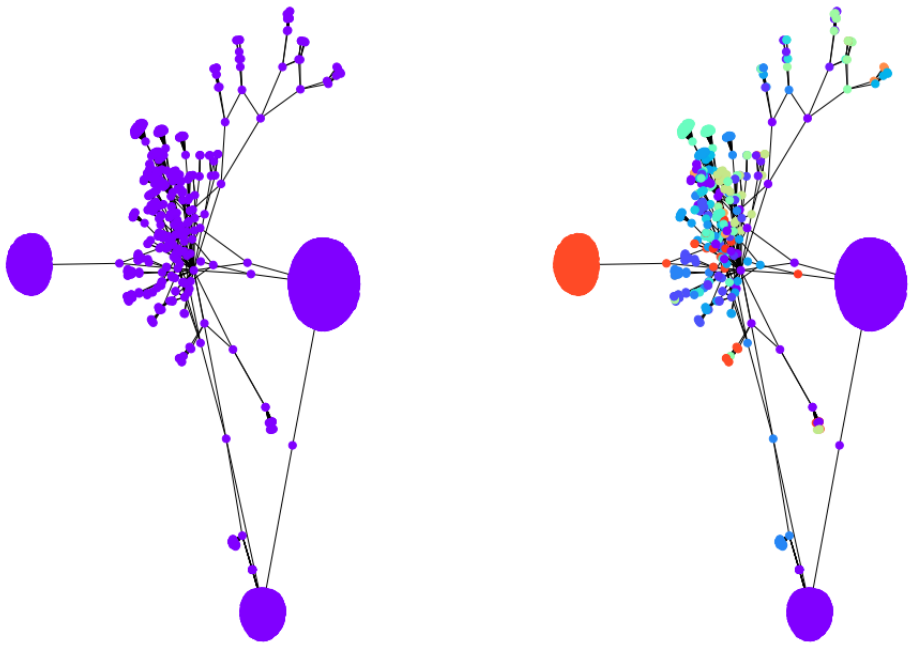
FIGURE 5.8: Clustering of the dolphin network.

no natural reason to expect it to work for this algorithm), and it did not provide any discernible benefit. Finding better ways to choose the initial cluster centers, while complementary in nature with respect to the focus of this chapter, will help make the algorithm much more useful, and is something we want to study in the immediate future.

Finally, we illustrate the initial clustering on a larger Bitcoin subgraph comprising 4571 nodes. We drew 1500 cluster centers uniformly at random, and it resulted in 67 clusters, as shown in Figure 5.9. The graphs are drawn using the spring layout of NetworkX, which tends to group nodes which are densely connected among each others. The clustering algorithm visually agrees: nodes which are drawn together tend to be clustered together. The algorithm was initiated with $C = 1500$, and when it stopped, the algorithm had find 67 clusters. This is because the algorithm gives the option to each point to change cluster, resulting in some clusters growing while others disappear.

5.3.2 Directed Graphs

We now consider the case of a directed graph $G = (V, E)$. For directed graphs, the shortest path does not provide a suitable distance metric. In the worst case, there could be a path from u to v, and yet no path from v to u, and in general, the shortest paths for the two directions may not be of equal length, violating the symmetry property of a distance metric. We are thus interested in a metric similar to a co-citation like behaviour (co-citation considers two nodes close if they are often out-neighbours of the same nodes) and adapt it to fit the proposed Renyi-entropy based mechanism. Said otherwise, we would like a metric that considers



(A) A subgraph of the Bitcoin network comprising 4571 nodes.

(B) An example of initial clustering made of 67 clusters.

FIGURE 5.9: Initial clustering of a Bitcoin subnetwork.

two nodes as close if both nodes are having many out-neighbours in common. For u, v two nodes in V, denote by $\mathcal{N}_{out}(u)$ and $\mathcal{N}_{out}(v)$ the set of out-neighbours of u and v respectively. Then, assuming that $\mathcal{N}_{out}(u) \cup \mathcal{N}_{out}(v)$ is non-empty, the quantity

$$\frac{|\mathcal{N}_{out}(u) \cap \mathcal{N}_{out}(v)|}{|\mathcal{N}_{out}(u) \cup \mathcal{N}_{out}(v)|}$$

gives the fraction of common out-neighbours among all the out-neighbours. It varies from 0 to 1, with it being 1 when all out-neighbours are mutually common. This is known as the Jaccard similarity coefficient, and it is related to a distance metric, namely the Jaccard distance

$$d_J(\mathcal{N}_{out}(u), \mathcal{N}_{out}(v)) = 1 - \frac{|\mathcal{N}_{out}(u) \cap \mathcal{N}_{out}(v)|}{|\mathcal{N}_{out}(u) \cup \mathcal{N}_{out}(v)|}.$$

We note that $d_J(u, v)$ does not exactly provide the measure we need, because two distinct nodes $u \neq v$ with the same set of neighbours would be at distance 0 (thus violating the condition of identity of indiscernibles). Therefore we slightly modify

the Jaccard distance, and use instead

$$\rho(u,v) = \frac{1}{2}(d_J(\mathcal{N}_{out}(u), \mathcal{N}_{out}(v)) + d_J(\mathcal{N}_{out}(u) \cup \{u\}, \mathcal{N}_{out}(v) \cup \{v\})).$$

Then $\rho(u,v) \geq 0$ with equality if and only if $d_J(\mathcal{N}_{out}(u), \mathcal{N}_{out}(v)) = d_J(\mathcal{N}_{out}(u) \cup \{u\}, \mathcal{N}_{out}(v) \cup \{v\}) = 0$. It follows that $d_J(\mathcal{N}_{out}(u), \mathcal{N}_{out}(v)) = 0 \iff \mathcal{N}_{out}(u) = \mathcal{N}_{out}(v)$ and therefore $d_J(\mathcal{N}_{out}(u) \cup \{u\}, \mathcal{N}_{out}(v) \cup \{v\}) = 0 \iff u = v$. We clearly have $\rho(u,v) = \rho(v,u)$, and the triangle inequality holds because it holds for d_J:

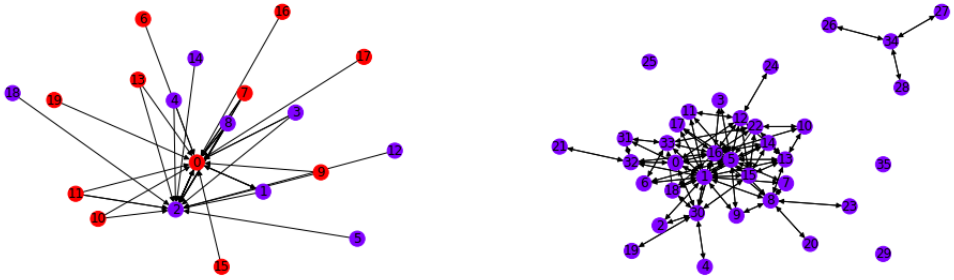
$$\begin{aligned} & \rho(u,v) + \rho(v,w) \\ = \; & \frac{1}{2}(d_J(\mathcal{N}_{out}(u), \mathcal{N}_{out}(v)) + d_J(\mathcal{N}_{out}(v), \mathcal{N}_{out}(w)) + \\ & d_J(\mathcal{N}_{out}(u) \cup \{u\}, \mathcal{N}_{out}(v) \cup \{v\}) + d_J(\mathcal{N}_{out}(v) \cup \{v\}, \mathcal{N}_{out}(w) \cup \{w\})) \\ \geq \; & \frac{1}{2}(d_J(\mathcal{N}_{out}(u), \mathcal{N}_{out}(w)) + d_J(\mathcal{N}_{out}(u) \cup \{u\}, \mathcal{N}_{out}(v) \cup \{w\})) = \rho(u,w). \end{aligned}$$

The normalization by $1/2$ in the definition of ρ just ensures that this quantity varies from 0 to 1. If now we have u an out-leaf, that is a leaf such that $\mathcal{N}_{out}(u)$ is empty, and v is another node with $\mathcal{N}_{out}(v)$ not empty, then there is no problem and $\rho(u,v) = 1$. However $\rho(u,u)$ cannot be computed and is set to 0. If both u, v are out-leaves, we again cannot compute $\rho(u,v)$, and set the distance to be 1, the reason being that 1 is the largest distance, which is attributed to all pairs of nodes that do not have neighbours in common.

We then can apply the same Algorithm 4 for generating an initial clustering, but with the distance now being the modified Jaccard distance instead of the shortest path. The only difference is the stop condition "if some nodes are still unlabelled": indeed, it could now be possible to just have reached a stage where all clusters have no out-neighbour, in which case the algorithm would stop. The algorithm is thus tweaked so that when choosing the neighbours, we consider both out- and in-neighbours.

We first apply our initial directed clustering algorithm on a toy example first, a random directed scale-free graph with 20 nodes. This allows us to explain the intuition behind the algorithm. In Figure 5.10a, nodes 0 and 2 both have high in-degrees, and the graph is clustered into two subgraphs, formed by the nodes pointing at 0 and those pointing at 2.

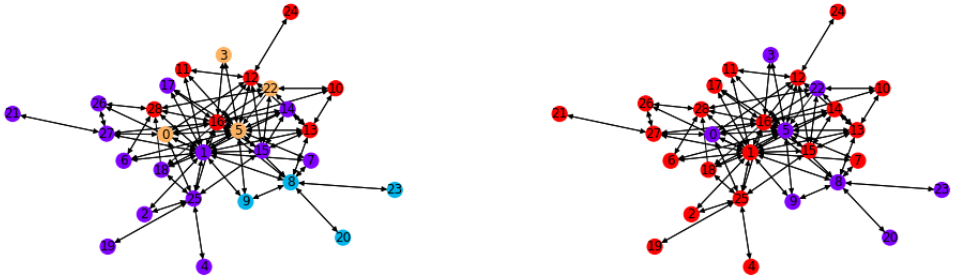
We next apply our hierarchical clustering on a real world data set, the Montreal gang network [187], shown on Figure 5.10b. Since the graph has different connected



(A) A clustering of a random graph using 2 original cluster centers, 12 and 15, as starting points.

(B) The Montreal Gang Network.

FIGURE 5.10: Interpretation of the directed clustering.



(A) 5 clusters after the initial clustering, cluster centers were 9,23,20,0,12.

(B) 2 clusters after 1 round of agglomeration.

FIGURE 5.11: Clustering of the Montreal Gang Network.

components, we apply the clustering on the main component. We ran instances where the cluster centers are 5 points chosen uniformly at random. One instance as shown in Figure 5.11a, this is the result after the initial clustering. We observe one main cluster, that contains both nodes 1 and 25, both have high in-degrees and their in-neighbours are in this cluster. Agglomerating these clusters gives the configuration showed on Figure 5.11b.

5.4 Summary

In the previous chapters we focused on designing flow based centrality computation and clustering techniques, all guided by information theoretic principles. In a departure from the flow-centric model, but to explore further how information theoretic insights can be leveraged for the purpose of community detection,

in this chapter we explored accordingly the use of Renyi entropy for graph clustering, which has previously been used to cluster images. We designed algorithms for clustering, and our experimental evaluation confirm that indeed, despite using very different kind of distance metrics for graphs, the principles originally explored for image clustering can be applied in the context of graphs. While the shortest path length is a natural distance metric that can be readily used for undirected graph, a challenge was to interpret a suitable distance metric for directed graphs. We propose a tweak of the Jaccard distance determined in terms of the neighbourhood of nodes to that end. Originally, we had tried a simulated annealing based approach to find suitable clusters, and while it does speed up the computations significantly with respect to a naive search of the solution space, unfortunately, it (or at least our implementation) did not scale adequately. Consequently, we designed hierarchical agglomerative algorithms which have inherent scalability traits. Since the computations involve localized information, the problem can be broken into smaller computational problems, and furthermore, it is also amenable to distribution and parallelization. Given the emphasis of this thesis in exploring novel (information theory guided) building blocks to interpret and analyze graphs, we have deferred the refinement of implementation optimizations for future study. Finally, our validation of the applicability of other distance metrics than what was originally used for Renyi entropy based clustering is empirical in nature. But in fact, not all distance metrics are necessarily interchangeable. Establishing a more principled understanding of the transformation across different objects (images and more generally, multi-dimensional space in the reals, versus graphs) and the corresponding notion of distances, and the distortion it introduces in the process, is another issue we want to study in the future.

All the discussions so far in this thesis have been about general purpose graph algorithms, which may also be applied to Bitcoin networks for macroscopic study, with which we can identify (groups of) nodes of interest. In the next chapter, we will discuss certain Bitcoin specific algorithms that facilitate closer study of Bitcoin subnetworks and nodes of interest.

Chapter 6

Graph Exploration Techniques for Bitcoin Network Analysis

In the previous chapters (see Chapters 4 and 5), we have shown how graph clustering can be used to discover communities in general, and in particular, how information flow (confinement) based clustering of the Bitcoin network can be used to identify community of Bitcoin wallet addresses within which a significant portion of money flow circulates in a confined manner. Likewise, in Chapters 3 and 4 we explored how to identify nodes with high centrality, which again is an indicator of importance and thus interest in many studies. Alternatively, one can also use side information (websites, forums, and so on) to discover a subset of Bitcoin nodes of initial interest.

Having identified subsets of nodes of interest in whichever manner, in order to gain more insights about them and possibly nodes in their neighbourhood, one would like to further analyse and visualize the subgraph which comprises the said subset of Bitcoin addresses, to understand relationships amongst them. Such exploration and analysis is the emphasis of this chapter.

Specifically, in this chapter, we discuss the design of Bitcoin graph specific algorithms, analyzing neighborhood subgraphs extracted and analysed in terms of path lengths among nodes and confluence of Bitcoin flow, to determine nodes that coordinate their actions amongst themselves (possibly because they are controlled by a single entity), and showcase a visual analytics tool (BiVA) for exploratory analysis of the Bitcoin network. We discuss how the Bitcoin transaction data can be explored, aided by visualization of subgraphs around nodes of interest, and integrating both standard and new algorithms, including the aforementioned flow

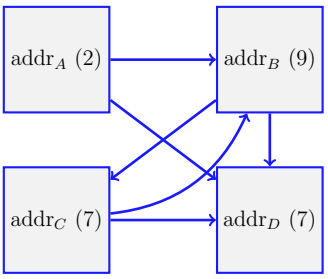


FIGURE 6.1: An address network corresponding to a set of Bitcoin transactions (see Table 2.2, Chapter 2). This is for the sake of illustration, when a transaction has several inputs and outputs, it is not always possible to reconstruct the list of exact transactions from the Bitcoin network.

based clustering for directed graphs (see Section 4.3.1 and Algorithm 1, in Chapter 4), and other Bitcoin network specific wallet address aggregation mechanisms.

We demonstrate the efficacy of our approach with a case study of extortion of Ashley Madison data breach victims [32] from August 2015. Specifically, we show how our approach can be used both to estimate the amount of money that was extorted by the suspected perpetrators under the specific blackmail campaign, and also estimate the amount of money handled by them during the same period of time. We also apply some of the common address aggregation heuristics [27, 31, 153] from the literature to demonstrate their shortcoming, putting in context the need for the flow confinement based approach studied in this work.

6.1 System Model

In our literature review (Section 2.2.2, Chapter 2), we have explained how the information from the Bitcoin blockchain can be modelled in various manner: (1) as a heterogeneous (2-mode) graph, where nodes comprise transactions, identified by their unique hash, and Bitcoin addresses, typically a user may own many Bitcoin addresses, (2) as a directed transaction network (see Subsection 6.5.2.5), and (3) as a directed address network (see e.g. [119]), where each node represents a Bitcoin address, and there is a directed link between two nodes if there was at least one transaction between the corresponding addresses, as shown in Fig. 6.1. The graph model we will use mostly next, unless otherwise stated, is the last (address network) one.

6.2 Notion of Directed Random Graphs

Recall that for a directed graph $G = (V, E)$ with $n = |V|$ nodes, and E its set of directed edges, we denote by k_i^{in} the in-degree of vertex i, and by k_i^{out} its out-degree. Since every edge of G must leave some node and enter another one, it is well known (and sometimes referred to as the Handshake Lemma or Theorem) that $\sum_{i=1}^{n} k_i^{in} = \sum_{i=1}^{n} k_i^{out} = |E|$.

6.2.1 Power-Law Directed Graphs

A power-law distribution is a probability distribution given by

$$p(x) = Cx^{-\alpha},\ x \geq x_{min}$$

where C is a normalization constant, so that, in the discrete case, $\sum_{x \geq x_{min}} p(x) = C \sum_{x \geq x_{min}} x^{-\alpha} = 1$. For the continuous case, it is easy to see that $\int_{x_{min}}^{\infty} p(x)dx = C \int_{x_{min}}^{\infty} x^{-\alpha}dx = C\frac{x_{min}^{-\alpha+1}}{\alpha-1}$, assuming that $\alpha > 1$ (otherwise the exponent $-\alpha + 1$ is positive and this quantify is infinite), from which $C = (\alpha - 1)x_{min}^{\alpha-1}$.

The mth moment of a random variable X distributed according to $p(x)$ is given by $\langle X^m \rangle = \sum_{x \geq x_{min}} x^m p(x)$, or $\langle X^m \rangle = \int_{x_{min}}^{\infty} x^m p(x)dx = C\frac{x_{min}^{-\alpha+1+m}}{\alpha-m-1}$ assuming that $\alpha > m + 1$, which simplifies to $x_{min}^m \frac{\alpha-1}{\alpha-m-1}$. In particular, the mean is given for $m = 1$.

A power-law (or scale-free) directed graph is a graph whose in-degrees $k_1^{in}, \ldots, k_n^{in}$ and out-degrees $k_1^{out}, \ldots, k_n^{out}$ are realizations of the random variables K^{in}, K^{out} which follow the respective power laws:

$$\begin{aligned} p_{in}(x) &= C_{in}x^{-\alpha_{in}},\ x \geq x_{min}^{in}, \\ p_{out}(x) &= C_{out}x^{-\alpha_{out}},\ x \geq x_{min}^{out}. \end{aligned}$$

Again, since every edge of such a graph must leave some node and enter another one, the average in-degree and out-degree are the same: $\langle K^{in} \rangle = \langle K^{out} \rangle$. Then by the law of large numbers, $\sum_{i \in V} k_i^{out} = \sum_{i \in V} k_i^{in} = n\langle K^{in} \rangle$ and the probability q_{ij} that nodes i and j are connected by a directed edge from i to j is

$$q_{ij} = \frac{k_i^{out} k_j^{in}}{\sum_{i \in V} k_i^{in}} = \frac{k_i^{out} k_j^{in}}{n\langle K^{in} \rangle}. \tag{6.1}$$

6.2.2 Average Path Length

We revisit the average path length analysis of [44] proposed for undirected graphs and extend the analysis for directed graphs. Let $G = (V, E)$ be a directed graph. An arbitrary sequence of vertices from V forms a directed walk on G, where vertices can be repeated, creating possibly directed cycles, and reaching a length which can possibly be arbitrarily large. In comparison, a directed path does not allow repetition of any vertex, thus it cannot contain any cycle. If there is a directed walk from i to j, skipping the cycles will give a path. Let $p_{ij}(l)$ be the probability that there is at least one walk of length l between i and j. Then in terms of statistical ensemble of random graphs [44], the probability $p_{ij}(l)$ of at least one walk of length l between i and j also captures the probability that the nodes i and j are neighbors of order not higher than l. Thus $p^*_{ij}(l) = p_{ij}(l) - p_{ij}(l-1)$ is the probability that i and j are at distance l from each other. Using (6.1) and Lemma 1 from [44], we have that

$$p_{ij}(l) \approx 1 - \exp(s_{ij}(l))$$

where

$$s_{ij}(l) = -\sum_{v_1 \in V} \cdots \sum_{v_{l-1} \in V} q_{iv_1} q_{v_1 v_2} \cdots q_{v_{l-1} j}$$

since in the directed graph G, at any node of the walk, the next step does not depend on the previous one. Then

$$\begin{aligned}
s_{ij}(l) &= -\sum_{v_1 \in V} \cdots \sum_{v_{l-1} \in V} \frac{k_i^{out} k_{v_1}^{in}}{n\langle K^{in}\rangle} \frac{k_{v_1}^{out} k_{v_2}^{in}}{n\langle K^{in}\rangle} \cdots \frac{k_{v_{l-1}}^{out} k_j^{in}}{n\langle K^{in}\rangle} \\
&= -\frac{k_i^{out} k_j^{in}}{n\langle K^{in}\rangle^l} \sum_{v_1 \in V} \cdots \sum_{v_{l-1} \in V} \frac{k_{v_1}^{in} k_{v_1}^{out} k_{v_2}^{in} \cdots k_{v_{l-1}}^{in} k_{v_{l-1}}^{out}}{n^{l-1}} \\
&= -\frac{k_i^{out} k_j^{in} \langle K^{in} K^{out}\rangle}{n\langle K^{in}\rangle^l} \sum_{v_1, \ldots, v_{l-2} \in V} \frac{k_{v_1}^{in} k_{v_1}^{out} \cdots k_{v_{l-2}}^{in} k_{v_{l-2}}^{out}}{n^{l-1}} \\
&= -\frac{k_i^{out} k_j^{in}}{n} \frac{\langle K^{in} K^{out}\rangle^{l-1}}{\langle K^{in}\rangle^l}
\end{aligned}$$

and

$$p^*_{ij}(l) = 1 - \exp(s_{ij}(l)) - 1 + \exp(s_{ij}(l-1)).$$

As in [44], averaging $p^*_{ij}(l)$ over all pairs of vertices one obtains the internode distance distribution $\langle\langle p^*_{ij}(l)\rangle_i\rangle_j$.

The expectation value for the average path length between i and j is

$$\begin{aligned} & l_{ij}(k_i^{out}, k_j^{in}) \\ = & \sum_{l=1}^{\infty} l p_{ij}^*(l) = \sum_{l=1}^{\infty} l(\exp(s_{ij}(l-1)) - \exp(s_{ij}(l))) \\ = & \sum_{l=1}^{\infty} l \exp(s_{ij}(l-1)) - \sum_{l=2}^{\infty} (l-1) \exp(s_{ij}(l-1))) \\ = & \sum_{l=0}^{\infty} \exp(s_{ij}(l)). \end{aligned}$$

This is happening because the graph can be arbitrarily large, and hence, so can be the length of a path. Since $\exp(s_{ij}(l))$ is a function of l which is non-negative, continuous, decreasing and Riemann integrable, we can invoke the following version of the Poisson summation formula [188, p.1]

$$\frac{1}{2} f(0) + \sum_{l=1}^{\infty} f(l) = \int_0^{\infty} f(t)dt + 2 \sum_{n=1}^{\infty} \int_0^{\infty} \cos(2\pi n t) f(t) dt$$

with $f(l) = \exp(s_{ij}(l))$. Since $\cos(2\pi n t)$ is integrable and does not change sign over the unit semi-circle, the generalized mean value theorem for integrals yields that $2 \sum_{n=1}^{\infty} \int_0^{\infty} \cos(2\pi n t) f(t) dt = 0$. Set

$$k = \frac{k_i^{out} k_j^{in}}{n \langle K^{out} K^{in} \rangle}, \; a = \frac{\langle K^{out} K^{in} \rangle}{\langle K^{in} \rangle}.$$

Then set $t = ka^l$, $dt = ka^l \ln a dl$, so that $\int_0^{\infty} \exp\left(-ka^l\right) dl = \int_k^{\infty} \frac{\exp(-t)}{\ln(a) t} dt$ and

$$l_{ij}(k_i^{out}, k_j^{in}) = \frac{1}{2} \exp(-k) - \frac{Ei(-k)}{\ln a}.$$

Furthermore [189, 8.214]

$$Ei(-k) = \gamma + \ln(k) + \sum_{m=1}^{\infty} \frac{(-k)^m}{m \cdot m!},$$

where $\gamma \approx 0.57721$ is Euler's constant. Since $-k \approx 0$ in most of the cases, $\exp(-k) \approx \exp(0)$ and $Ei(-k) \approx \gamma + \ln(k)$ (computations done on the considered sample graphs confirm that the approximations are actually tight), we finally

obtain

$$l_{ij}(k_i^{out}, k_j^{in}) \approx \frac{1}{2} - \frac{\gamma + \ln(\frac{k_i^{out} k_j^{in}}{n\langle K^{out} K^{in}\rangle})}{\ln(\frac{\langle K^{out} K^{in}\rangle}{\langle K^{in}\rangle})}. \tag{6.2}$$

6.3 Graph Algorithms for Exploring Bitcoin Subnetworks of Interest

We incorporate the insights from the above analysis in designing our Bitcoin network analysis methodology as follows. We first identify some address/transaction subgraphs of the Bitcoin network that seem relevant to the scam to be studied. We then look at the subgraph of interest as a realization of a random graph with a specific in-degree/out-degree node distribution, and look for abnormal patterns in terms of path behavior. To do so, we leverage on the extension of [44] that studies the expected behavior of (power-law) graphs in terms of average path length, to the case of directed graphs (as detailed above in Section 6.2). Nodes on the Bitcoin address graph that are connected to each other by a distance that is statistically significantly shorter than the expected average behavior as per the analysis, as well as nodes with high number of shortest paths among them can be discerned as outliers, and indicate that these nodes are involved in Bitcoin money flows with confined circulation. We accordingly design an algorithm (described in Subsection 6.3.1) to trace the confinement of money flows by establishing confluence of flow paths over the address network. This also enables us to identify new addresses to investigate, that need to be working in tandem with the original detected nodes, for realizing such specific circulations. We can thus aggregate such newly identified addresses with the original list of addresses.

Finally, in order to make the analysis techniques easily accessible, we have designed and implemented a modular tool for Bitcoin network visualization and analysis (BiVA [48]), incorporating our own algorithms, as well as standard graph algorithms and third party Bitcoin analysis heuristics.

Accordingly, we next elaborate the graph analytics algorithms to explore subnetworks of interest. Unless otherwise indicated, we principally deal with the address network, and in the following we will denote by $G = (V, E)$ the graph formed by having Bitcoin addresses as vertices in V, and directed edges in E representing a flow of Bitcoins from one address to another (using the convention explained earlier for multi-input multi-output transactions). We use the notation $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$ to

refer to an undirected graph obtained from G, such that $\forall(u,v) \in E$, $(u,v) \in \overleftrightarrow{E}$ and $(v,u) \in \overleftrightarrow{E}$. Likewise, we use the notation $\overleftarrow{G} = (V, \overleftarrow{E})$ to denote the directed graph obtained by reversing all the edge directions from G, so to say, $(v,u) \in \overleftarrow{E} \iff (u,v) \in E$.

Earlier we have explored two general purpose information theory based algorithms, which, when applied to Bitcoin network, derive *macroscopic* information from the address graph - one for node centrality and one for clustering. Next, we discuss two *microscopic* (ego-centric) graph algorithms which are centered around a pair/set of address nodes (in contrast to the above described centrality and clustering algorithms which are macroscopic). These microscopic algorithms are applied when certain addresses of interest are already identified - either by the macroscopic algorithms (e.g., nodes within an identified cluster or certain high centrality nodes), or by some out-of-system mechanisms or for ad-hoc exploration of the network.

6.3.1 Path Confluence Based Address Aggregation

Algorithm 6 Money flow confluences involving $v_1, v_2 \in V$

1: **procedure** SEARCHCONFLUENCE($G = (V,E), v_1, v_2$)
2: Set $S_{flow} = \varnothing$.
3: Determine shortest path from v_1 to v_2 on $\overleftrightarrow{G}$ to form a set of nodes d_{v_1,v_2}.
4: **if** d_{v_1,v_2} is not a directed path on G **then**
5: Set V_{flip} : nodes in d_{v_1,v_2} whose both neighbors are in d_{v_1,v_2} and are both either direct successors (or direct predecessors) in G.
6: **for all** $v_c \in V_{flip}$ **do**
▷ The breadth first search (BFS) algorithm is tweaked to identify and return nodes that are revisited.
7: **if** v_c is successor of neighbors **then**
8: Compute V_{v_c} =BFS($\overleftarrow{G}, v_c$)
9: **else** ▷ v_c is predecessor of neighbors
10: Compute V_{v_c} = BFS(G, v_c)
11: **for all** $v_q \in V_{v_c}$ **do**
12: Compute all (length limited) paths starting from v_c to v_q (or vice-versa: similar to steps 7-10), to form multiple set of nodes $d_{v_c,v_q,i}$ (or $d_{v_q,v_c,i}$).
13: Exclude all paths between v_c and v_q if they are connected by a *cut edge*.
14: $S_{flow} = S_{flow} \bigcup_i d_{v_c,v_q,i}$
15: **return** S_{flow}

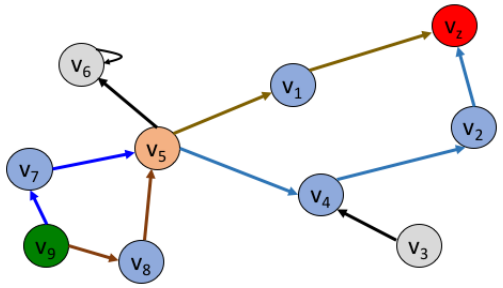


FIGURE 6.2: Example graph: A chain of path confluences between v_9, v_5 and v_z, discovered using Algorithm 6 with the graph and v_1 and v_2 as inputs.

A key intuition we pursue is that the money flow may be confined and circulate among certain addresses, indicating in turn coordination among said addresses. To identify such confinement of the flow systematically, we designed Algorithm 6 that finds path confluences over the directed address graph.

Consider a shortest path[1] d_{v_1,v_2} between two query vertices v_1 and v_2 in the undirected graph $\overleftrightarrow{G}$. The query vertices are nodes of initial interest for whichever reasons. If d_{v_1,v_2} is not a path in G, it means that the directionality flips along d_{v_1,v_2}. Starting at each such flipping points (only v_z in the example graph shown in Fig. 6.2), we run a variation of the breadth first search (BFS) algorithm (starting independently at each flipping point), and returns all the nodes that are revisited: which would be v_5, v_9 and v_6 (because of the self-loop) in our example. Note that the BFS algorithm variation is run over G or $\overleftarrow{G}$, depending on whether the flipping point had outgoing or incoming edges respectively (the latter in our example). This helps us identify (a chain of) path confluences. For instance, from v_9 to v_5, and again from v_5 to v_z, with the second group of path confluences including our original nodes of interest v_1 and v_2. Note that v_6 would seem to have two distinct paths to v_z, one through v_1 and another through v_2. However they are not independent, since both share a single edge v_6, v_5 (unlike v_9 which had multiple paths to v_5). We eliminate from our consideration all the paths that involve any *cut edge*. Such confluence, particularly if the path lengths (between v_x and v_y) are not very long, are statistically unlikely unless all the nodes on the paths are in fact acting in a coordinated manner.

[1]In the following we abuse the notation to indicate both the path (which is ordered list of nodes) and the unordered set.

6.3.2 Bitcoin Flow Traceback/Traceforward

Given a set S of address nodes, we may want to trace where they collectively get the money from. Accordingly we design Algorithm 7. We first identify the set $S_x \subseteq V$ that has any path to at least some node in S. S_x can be identified by traversing the edges in the reverse sense using a breadth first search (BFS). In terms of practicality, it may also make sense to limit the search within a distance threshold instead of exploring the whole graph. Next, we consider the graph G' which comprises only the vertices that are in $S \bigcup S_x$, and the edges that are among nodes in this subset. We consider that each edge $e \in E$ has a weight, determined by the Bitcoin amount. Consider now the case of a multi-input multi-output transaction, where an incoming address x_i has an edge of weight w_i to the transaction, and the transaction has directed edges with weight w_o to outgoing address y_o. Then, in the address graph, x_i will have a directed link to y_o with weight $w_i(w_o/\sum_i w_i)$. This weighted average acts as a proxy, in the absence of determinism due to mixing. The expectation is that even if one such a guess instance is just noise, aggregated volume can be used to draw attention to nodes of interest. We thus consider the sum of the weights of incoming edges to a vertex in G' as an indicator of the expected amount of money flowing through it to the original set of nodes of interest S. The same idea can be applied on $\overleftarrow{G}$ to deduce where the money from these nodes S flow to downstream.

Algorithm 7 Tracing money flow through set of address nodes $S \subseteq V$

procedure ExpectedMoneyFlowingThrough($G = (V, E), S$)
  Set $S_{flow} = \varnothing$.
  ▷ Initialize a set of nodes $S_x \subseteq V$ that has any path to at least some node in S.
  Set $S_x = \varnothing$.
  for all $v_c \in S$ **do**
  ▷ The breadth first search (BFS) algorithm identifies and returns nodes that are visited within length limit.
    Compute $S_x = S_x \bigcup \text{BFS}(\overleftarrow{G}, v_c)$
  $G' = \text{SUBGRAPH}(G, S \bigcup S_x.)$
  for all $v_q \in S \bigcup S_x$ **do**
    Compute w_q : sum of weight w_i of incoming edges $e_i(v_n, v_q, w_i) \in G'$ to form a pair (v_q, w_q). ▷ v_n is predecessor of neighbor
    $S_{flow} = S_{flow} \bigcup \{(v_q, w_q)\}$
  return S_{flow}

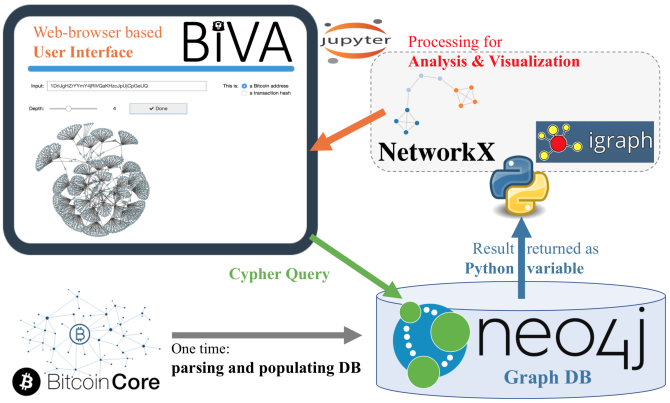


FIGURE 6.3: BiVA system architecture.

6.4 BiVA: Bitcoin Network Visualization and Analysis Tool

We next discuss the design of a tool (BiVA) that we implemented for facilitating Bitcoin network visualization and analysis BiVA, by incorporating some of the algorithms discussed in this thesis, as well as other third party graph analysis and visualization techniques. The overall architecture of BiVA is shown in Figure 6.3.

The front-end of BiVA is implemented in Python. We use Jupyter dashboard [190] to realize a web-browser based front-end, which embeds the codes necessary for querying and processing that can be easily hidden, to expose only the dashboard based user interface. Snippet of the interface so realized is shown in Figure 6.6. The back-end database is queried with the Cypher query language [191] using ipython-cypher, and the returned result is passed as a Python variable for further processing. This includes execution of graph algorithms (standard ones, as well as custom algorithms designed for BiVA, as described in Section 6.3) that leverage on the NetworkX [161] package for analysis, and the igraph [160] package for graph rendition.

The back-end database is populated by downloading the blockchain with Bitcoin Core [192], and parsed to be stored in the Neo4j (v3.4.1) [163] graph database management system. The graph model we consider is heterogeneous and consists of two types of nodes (2-mode network), transactions and (wallet) addresses, which we model using two different labels in Neo4j. Relationships are also modelled as two types, depending on whether a directed edge describes an input to or an output from a transaction. This is illustrated in Fig. 6.5: the upper part shows the 2-mode

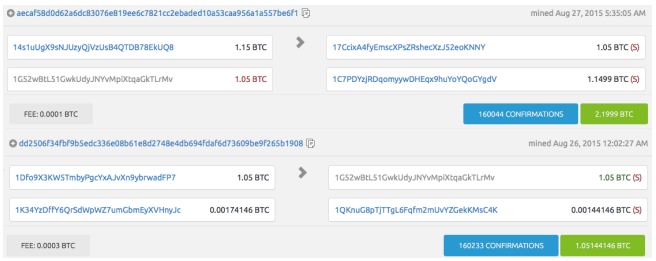


FIGURE 6.4: The wallet address *1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*'s record (from https://blockexplorer.com/).

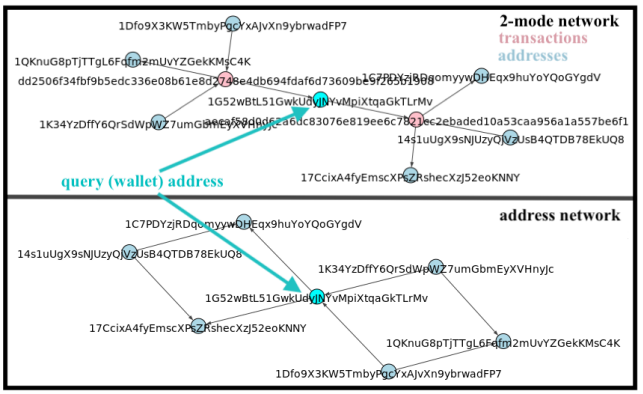


FIGURE 6.5: An ego-centric view of a 2-mode (transaction-address) network, and the corresponding address network.

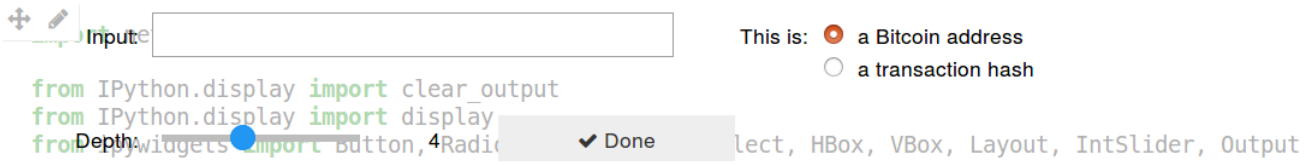


FIGURE 6.6: Jupyter dashboard based user & programmer interface snippet.

network, obtained as the neighborhood of a chosen query (wallet) address within a perimeter of 2. Pink nodes are transaction nodes, and the query address is shown to be both an output and an input to two respective transactions. The lower part contains the corresponding address network, obtained by removing the transaction nodes. Since we have multi-input and output transactions, the directed link going from the query address to the transaction is changed into two links to both transaction outputs. Similarly, the incoming edge to the query address becomes two incoming edges from addresses. Fig. 6.4 shows the corresponding address record as obtained from `blockexplorer.com`.

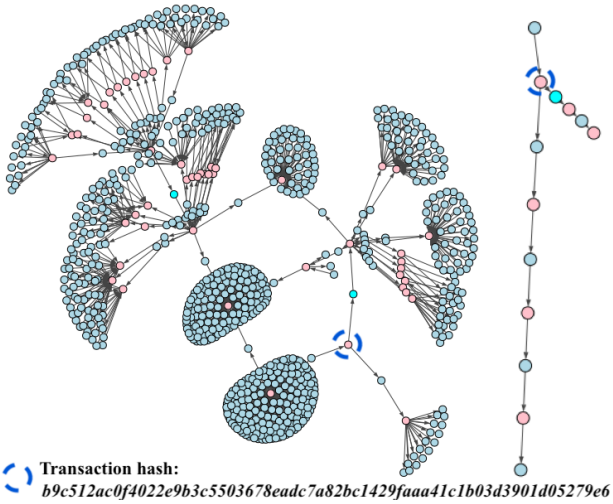


FIGURE 6.7: Filtering of a Bitcoin subgraph, shown on the left, that detects a chain of large ($\geq$ 2000฿) transactions, on the right.

6.5 Results

Some results obtained with the above algorithms are illustrated next with a case study.

6.5.1 Data Set Exploration

We first elaborate how the tool (BiVA) itself can be used, then we delve deeper to see the kind of information we can infer with our analysis techniques.

BiVA supports the visualization of standard queries on the Bitcoin network: display of the 2-mode network (or of the transaction only network, or of the address network) centered around a specific wallet address or transaction query within a given parameter or time frame, display of shortest/multiple paths among addresses on the 2-mode network or the address network, to name a few. BiVA can also filter the edges by the amount of Bitcoins involved, or find addresses of interest (for instance, addresses with high centrality, addresses that are aggregated together based on several existing and newly proposed algorithms) in the neighborhood of the specific query address. In Figure 6.7, we show a 2-mode network, centered around the address *1HxuYkYf9SXAD9raVV3UKe3AZvrREgviYs*. This node, in light blue, appears twice, and is involved in 4 transactions (shown in pink). By filtering the network shown on the left, looking for transactions of more than 2000฿, we identify a chain of large transaction amounts, shown on the right.

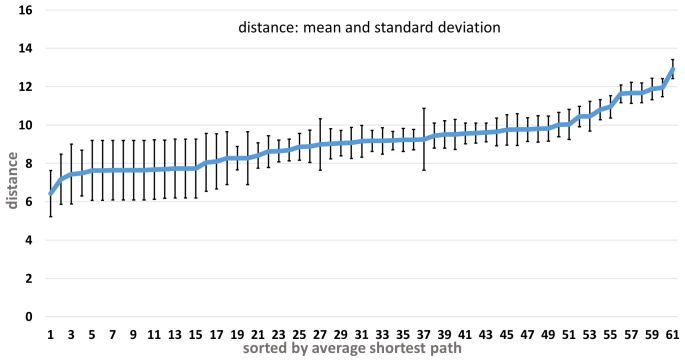


FIGURE 6.8: Pair-wise distance statistics (average and standard deviation) among 61 suspect Bitcoin wallet addresses.

We next focus on results from some of the new algorithms, by exploring an extortion of Ashley Madison data breach victims.

6.5.2 Case Study: Extortion of Ashley Madison Breach Victims

In this section we will look at the Ashley Madison extortion incident that followed the data breach of Ashley Madison website [32]. The Ashley Madison website billed itself as an enabler of extra-marital affairs. In the extortion incident, the perpetrator(s) asked for a given amount of 1.05฿ from Ashley Madison website data breach victims, so that their information was not brought to the attention of close friends and family. Thus, by looking at transactions of 1.05฿ for the period around 23 August 2015 when the extortion campaign was carried out, the author of [32] identified a list of 67 Bitcoin wallet addresses which could potentially have received payments for said blackmail.

We start our analysis with these 67 addresses, and find that 61 of the addresses are connected to each other in an undirected variant of the address graph, while the other six are totally isolated. In Fig. 6.8 we show the distribution of pair-wise distances among the 61 addresses, based on which we see that many of the addresses are still very isolated from the other addresses. Nevertheless, we studied graphs of radius 6 around each of them, to identify subsets of nodes within the specified distance: 39 of the nodes were each isolated from all other nodes in the group by a distance of at least 6. Only one subset comprising 22 of the original 61 nodes were identified to be close enough to each other, not only in the context of (undirected) distance, but also in being part of a local community structure, where a substantial flow of money is confined. Previously, we had identified these

addresses to belong to a cluster in our analysis in Section 4.5.5. Next, in Section 6.5.2.3 we see that the results from our macroscopic clustering and microscopic study cross-validate each other. Accordingly, we focus the rest of our study around these nodes. Specifically, our subsequent experimental studies are on subgraphs[2] (of radii 4-5) centered around the node closest on an average to the aforementioned group of 61 nodes, namely *1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*. They contained the 22 suspected nodes.

6.5.2.1 Degree Distributions

Power-law graphs are of interest in the context of the Bitcoin network. In [119], the authors show that over different one month periods after the fall of 2010 (a period that they call the trading phase), the in-degree and out-degree distributions do not change much (it is claimed that the network measures converge to their typical value by mid-2011) and are approximated by power-laws $p_{in}(x) \sim x^{-2.18}$ and $p_{out}(x) \sim x^{-2.06}$.

We thus consider two samples of the Bitcoin network, a (very) small sample $G_1 = (V_1, E_1)$ with $|V_1| = 4571$ and $|E_1| = 4762$, obtained by extracting a subgraph (considered undirected) of radius 4 around the given address of interest (*1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*) as discussed in Subsection 6.5.2, and a small sample $G_2 = (V_2, E_2)$ with $|V_2| = 82663$, $|E_2| = 119190$ obtained by considering a radius 5 instead (this corresponds to the period of Aug 13 to Sep 6 2017), and explore whether they are power-law graphs. We checked the fit of the in-degree and out-degree distributions with respect to power laws for both G_1 and G_2 using [168]. The results are shown in Fig. 6.9 and 6.10. The algorithm in [168] looks for either the best fit, or the best fit given a choice of x_{min}. We chose to show the results obtained by fixing x_{min} to be 1, since we are more interested in the overall behavior of the distributions rather than wanting the best fit.

We do observe, as expected, that both graphs exhibit a power-law behavior for both their in-degree and out-degree distributions. The exact values of α_{in} and α_{out} observed on both figures are however purely indicative, we do not draw conclusions compared to the values described in [119] since (1) we would need the value of x_{min} in [119] for a proper comparison, and more importantly (2), the graph sizes are not comparable. In fact, by extracting a subgraph of (much) smaller size, it is not

[2]We consider subgraphs from the wallet address network constructed from 13 August 2015 till 6 September 2015 (i.e., the studied period started 10 days before the extortion campaign, and ended 10 days after the extortion campaign).

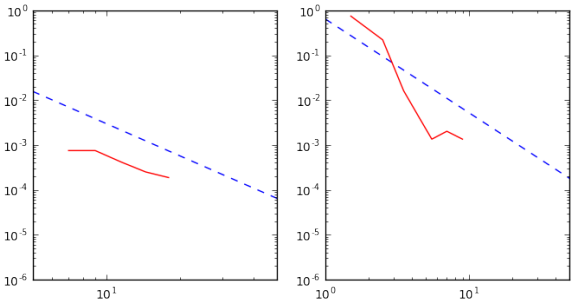


FIGURE 6.9: Fitting a power law for G_1: on the left, the in-degree distribution with $\alpha_{in} = 2.38$, $x_{min}^{in} = 1$, and on the right, the out-degree distribution $\alpha_{out} = 2.08$, $x_{min}^{out} = 1$.

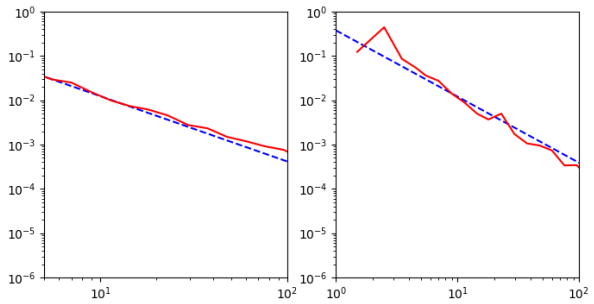


FIGURE 6.10: Fitting a power law for G_2: on the left, the in-degree distribution with $\alpha_{in} = 1.47$, $x_{min}^{in} = 1$, and on the right, the out-degree distribution $\alpha_{out} = 1.493$, $x_{min}^{out} = 1$.

clear that the graph characteristics are preserved, and by cutting the graph at a given radius, the boundary effects should be taken into accounts. The choice of a subgraph of radius r around a chosen address will be justified by the desired analysis on this specific address.

Please note that, as an immediate heuristic, we tried, for a graph obtained as a subgraph of radius r around an address, to give as in/out-degree to each boundary node what it would have in the graph of radius $r + 1$, however it tends to give too much a difference between $\langle K^{in} \rangle$ and $\langle K^{out} \rangle$ (which are supposed to be the same if the whole graph were to be considered) for this approach to make sense.

6.5.2.2 Average Length Analysis

In previous Subsection 6.5.2.1, we observed that the two subgraphs followed power-law degree distribution. Thus the average path length analysis from Section 6.2.2 is applicable to them.

We again consider two samples of the Bitcoin network: the graphs $G_1 = (V_1, E_1)$ with $|V_1| = 4571$ and $|E_1| = 4762$, and $G_2 = (V_2, E_2)$ with $|V_2| = 82663$,

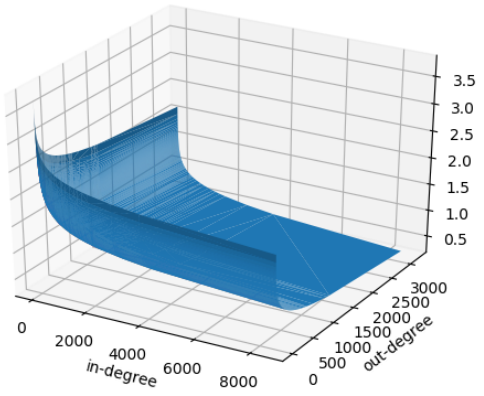


FIGURE 6.11: Average path length for G_2 according to (6.2).

$|E_2| = 119190$ obtained by extracting a subgraph (considered undirected) of respectively radius 4 and 5, around a given address of interest (*1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*). We chose to limit the radius of graph to study to 5, to keep the analysis tractable, particularly for manual validation of the results obtained. The graph $G_3 = (V_3, E_3)$ with radius 6 has $|V_3| = 1374498$ and $|E_3| = 4722501$.

Formula (6.2) can be used to evaluate the average path length in these graphs. In the case where in-degree/out-degree distributions are known to be power laws, closed form expressions for $\langle K^{in} \rangle$ can be used, assuming that $\alpha > 2$. However, since the derivation of (6.2) does not rely on the distributions to be power laws, we evaluate $\langle K^{in} \rangle$ and $\langle K^{out} K^{in} \rangle$ numerically (alternatively, since we know from the previous section that our graphs have distributions that are close to a power law, we could vary the choice of x_{min} to get the best fit in terms of power law, but still the closed-form formulas for the mean of a power law is available only for specific exponents α). To get a sense of (6.2), we provide some representative values in Table 6.1: we see that paths from nodes with out-degree 1 to nodes with in-degree 1 have approximated average lengths of 4, which appears meaningful, considering that the studied subgraphs have radii respectively 4 and 5, and that the nodes with these degrees are typically on the boundary (they are an artefact of cutting the edges after a chosen radius). The values for $(1, 1)$ are the maximum values found for G_1 and G_2. The minimum values are given for $(2807, 196)$ and $(8530, 3196)$ respectively: these are nodes with (very) high degrees, they are thus expected to be neighbors. Fig. 6.11 shows the average lengths determined using (6.2) for all the in-degree/out-degree combinations in G_2.

Since our analysis starts from a given address of interest (*1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv* for the illustration), we look for anomalies in terms of path

(k^{out}, k^{in})	(1,1)	(5,7)	(2807,196)	(8530,3196)
G_1	3.524	2.612	0.1335	NA
G_2	3.802	3.065	NA	0.254

TABLE 6.1: Maximum and minimum average path lengths approximated by (6.2) for G_1 and G_2.

in-address	k^{in}	out-address	k^{out}	length l_{ij}		actual average
1EuRpZCEoyRKWcEpv6jD42N2CxJWrrETcD	2	15hWpb3m5VXdn9KVsS4rSMnrQQJLhXVyN4	16	2	3.084	5.403
1HZErPx5XPrp8mH98rKRjyazQLYN7j34Lk	2	15hWpb3m5VXdn9KVsS4rSMnrQQJLhXVyN4	16	2	3.084	5.403
17CcixA4fyEmscXPsZRshecXzJ52eoKNNY	2	15hWpb3m5VXdn9KVsS4rSMnrQQJLhXVyN4	16	2	3.084	5.403
1PUt7bm1ZU6BRT85yDJxd651Fgneveuzs3	2	15hWpb3m5VXdn9KVsS4rSMnrQQJLhXVyN4	16	2	3.084	5.403
14gkHySfCnoV4Fy4nCxoLUgavMP3DMXKsS	2	15hWpb3m5VXdn9KVsS4rSMnrQQJLhXVyN4	16	2	3.084	5.403
137ENC6fNM97tawPMAGgyn9BKLEmkTx7Xd	2	1J97MX1kCHMgePTBwB97KaHyjSYnkTyt7V	14	2	3.111	5.602
12bbs3MZAsd9mseXKTA93PtHVd2nSkvjDy	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	2	3.161	5.328
1Hmjw6JcNDxCQuVo7pkUmRmJzRJe4yLze9	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	2	3.161	5.328
14YeaK34GE68S6YASdtHR1iThj8c8hBQ1C	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	2	3.203	5.490
1MAwvr21q95UBu6y4WhRM47K5RBijQ164G	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	2	3.203	5.346
1Kbj3tzAbtHgJHt6h6G5PuvWYRZkWXAfhH	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	2	3.203	5.346
1KFc1tekeQtZg48pKbCEU2j7J14N2XFTNW	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	2	3.203	5.346
13Mj4nmV127symEJ6GTTb4kdTWyQ884QoN	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	2	3.203	5.346
1ETmDWjyto3gGeqch41PCeYJiVx9pmsuv	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	2	3.203	5.346
1ErwF3T6QCdZ75PBXZdyxXJPN2k7bBV9fd	2	1HZErPx5XPrp8mH98rKRjyazQLYN7j34Lk	2	2	3.514	5.566
1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv	2	17CcixA4fyEmscXPsZRshecXzJ52eoKNNY	2	2	3.514	5.566
18euqRRpC2Zp9i9dwrT7Qp3M8jfbu9TUn6	2	14gkHySfCnoV4Fy4nCxoLUgavMP3DMXKsS	2	2	3.514	5.566
1Q3AvrmZ8cykHGQ7kCkSZaopnWiHPzc1Qw	2	14gkHySfCnoV4Fy4nCxoLUgavMP3DMXKsS	2	2	3.514	5.566
1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy	2	1PUt7bm1ZU6BRT85yDJxd651Fgneveuzs3	2	2	3.514	5.566
1BkUNiTfJRTTG1iycaWGZavmLnLVHorUUo	2	14YeaK34GE68S6YASdtHR1iThj8c8hBQ1C	2	2	3.514	5.566

TABLE 6.2: Paths of length shorter than average, described by their start (in-address with out-degree k^{out}) and their end (out-address with in-degree k^{in}). The column *l*ength gives the actual path length, l_{ij} is the value for (6.2), and *a*verage is the actual average computed numerically on G_2.

length around this node in the graph G_2. The most prominent pairs of nodes with paths shorter than expected are given in Table 6.2. The first and the third columns give respectively the beginning and end of the path, with their in/out-degree. Since l_{ij} is an approximation based on (6.2), for sanity check, we compute the actual average path length between nodes of given degrees, and observe that l_{ij} is in fact a lower bound on the actual average length.

6.5.2.3 Probabilistic Flow Based Clustering Analysis

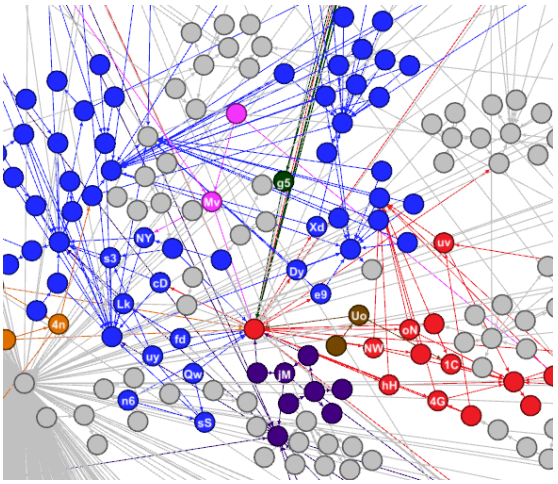
Our implementation of BiVA includes aforementioned flow based clustering for directed graphs (see Section 4.3.1 and Algorithm 1, in Chapter 4). The discussion below repeats some information regarding the experiment which has been discussed at length in Section 4.5.5, to highlight the connection with the rest of the work that we discuss in this chapter. Accordingly, we omit most of details and highlight only that the macroscopic analysis can be used to identify subgraphs or nodes of interest, which are then studied with Bitcoin specific microscopic analysis (the focus of this Chapter) as discussed in next Subsection 6.5.2.4.

In Figure 6.12b[3], we report the same clustering result (but with different layout

[3]The current implementation of BiVA relies on igraph for visualization, however, some of the visuals shown in this thesis are recreated with Gephi [10], since, in our experience, Gephi allowed greater flexibility in manual adjustment of layout and coloring, which facilitated better presentation of the results in this thesis.

1C	14YeaK34GE68S6YASdtHR1iThj8c8hBQ1C
4n	1Gscvx3ugexW4xKjBFogQtQRsHR9Eh8v4n
4G	1MAwvr21q95UBu6y4WhRM47K5RBijQ164G
cD	1EuRpZCEoyRKWcEpv6jD42N2CxJWrrETcD
Dy	12bbs3MZAsd9mseXKTA93PtHVd2nSkvjDy
e9	1Hmjw6JcNDxCQuVo7pkUmRmJzRJe4yLze9
fd	1ErwF3T6QCdZ75PBXZdyxXJPN2k7bBV9fd
g5	1JBygewRQuHuh4qJnpQtb1qCfwNJ2zNrg5
hH	1Kbj3tzAbtHgJHt6h6G5PuvWYRZkWXAfhH
jM	1LnxsZDPNkU96mrYoS1CQwxPmrhJyrgujM
Lk	1HZErPx5XPrp8mH98rKRjyazQLYN7j34Lk
Mv	1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv
n6	18euqRRpC2Zp9i9dwrT7Qp3M8jfbu9TUn6
NW	1KFc1tekeQtZg48pKbCEU2j7J14N2XFTNW
NY	17CcixA4fyEmscXPsZRshecXzJ52eoKNNY
oN	13Mj4nmV127symEJ6GTTb4kdTWyQ884QoN
Qw	1Q3AvrmZ8cykHGQ7kCkSZaopnWiHPzc1Qw
s3	1PUt7bm1ZU6BRT85yDJxd651Fgneveuzs3
sS	14gkHySfCnoV4Fy4nCxoLUgavMP3DMXKsS
uv	1ETmDWjyto3gGegqch41PCeYJiVx9pmsuv
uy	1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy
Uo	1BkUNiTfJRTTG1iycaWGZavmLnLVHorUUo
Xd	137ENC6fNM97tawPMAGgyn9BKLEmkTx7Xd

(A) Suspect address index.



(B) Probabilistic flow based clustering.

FIGURE 6.12: Identifying clusters and zooming into suspect addresses.

and coloring), as the result previously investigated in Section 4.5.5 (see Figure 4.12 in Chapter 4), on a directed address subgraph containing nodes within a distance of 5 (in an undirected variant of the address graph) centered around one of the suspect addresses, namely, *1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*. This subgraph had 4571 nodes and 4762 directed edges, and 23 of the aforementioned suspected nodes[4]. Our probabilistic flow based clustering algorithm is implemented in a bottom-up agglomerative manner, and we show the results from an intermediate fifth iteration of agglomeration (we focus on the portion of the subgraph containing the 23 suspect addresses post clustering), just before the two (red and blue) clusters (with largest number of suspect nodes) of interest coalesce. The nodes with labels indicate the originally suspect addresses. Nodes in clusters with no originally suspected address are all shown in gray. 12 suspect nodes are in the blue cluster comprising 120 nodes (11 of these 12 were clustered together already in iteration 1, which at that point had 24 nodes), while the red cluster of 15 nodes contains 6 of the suspected nodes. The unsupervised aggregation of some of the suspected addresses by our flow based clustering is both an initial validation of the clustering approach itself; conversely, the other members of these clusters, even though not originally suspected, become new addresses of interest.

[4]Overall, 61 of the suspected nodes were in a single connected component of the undirected variant of the address graph.

in-address	k^{in}	out-address	k^{out}	length l	$\tau(l)$	l_{ij}
1ErwF3T6QCdZ75PBXZdyxXJPN2k7bBV9fd	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	4	1.333	3.161
1ErwF3T6QCdZ75PBXZdyxXJPN2k7bBV9fd	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	4	0.8	3.203
1JBygewRQuHuh4qJnpQtb1qCfwNJ2zNrg5	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	4	2.0	3.161
1JBygewRQuHuh4qJnpQtb1qCfwNJ2zNrg5	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	4	0.8	3.203
1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	4	1.333	3.161
1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	4	0.8	3.203
18euqRRpC2Zp9i9dwrT7Qp3M8jfbu9TUn6	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	4	1.333	3.161
18euqRRpC2Zp9i9dwrT7Qp3M8jfbu9TUn6	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	4	0.8	3.203
1Q3AvrmZ8cykHGQ7kCkSZaopnWiHPzc1Qw	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	4	1.333	3.161
1Q3AvrmZ8cykHGQ7kCkSZaopnWiHPzc1Qw	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	4	0.8	3.203
1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	4	1.333	3.161
1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy	2	**1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2**	9	4	0.8	3.203
1BkUNiTfJRTTG1iycaWGZavmLnLVHorUUo	2	15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF	11	4	1.333	3.161

TABLE 6.3: Paths of low normalized length $\tau(l)$, defined in (6.3), described by their start (in-address with out-degree k^{out}) and their end (out-address with in-degree k^{in}).

6.5.2.4 Confluence Analysis

We next consider another indicator, where instead of a length l shorter than average, we take into account the normalized length, which we define as

$$\tau(l) = \frac{l}{\text{no. of shortest paths}} \tag{6.3}$$

where l is the length of the shortest path between two nodes of given degrees. This can be used as a metric of how closely knit two nodes are. In fact, the idea could also be generalized to count paths capped within a distance, rather than using only shortest paths. But, as a first approach, we focus on the case where we use shortest paths only. Table 6.3 lists the most prominently close knit nodes found through this heuristic.

Comparing the entrees in Tables 6.2 and 6.3, we identified wallet addresses *1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy* and *1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2* to be repeated (among other addresses) across the tables. This piqued our interest to study these addresses, and we use them to showcase our methodology of using flow confluences to study relationship among wallet addresses, where we consider more generally short paths but not necessarily the shortest ones (and paths of different lengths could be involved in the confluence).

We applied the path confluence algorithm (described in Section 6.3.1) among pairs of nodes from the shortlist of 22 nodes (described in Section 6.5.2), to determine if they in-fact interact with each other, even if indirectly, but in a manner which suggests coordination among their activities. Based on the confluence algorithm, for this specific set of addresses, we indeed observe that the money is circulated among them, and the addresses are in-fact close knit (more precisely, 102 paths of confluence were found). The confluence algorithm also helps us identify 23 further addresses that the original suspect addresses seem to be liaising

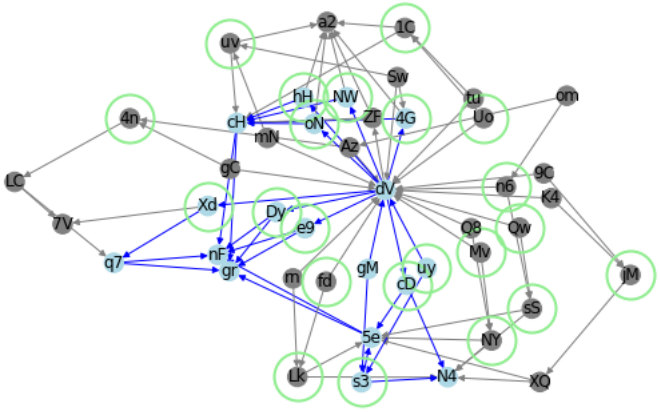


FIGURE 6.13: We show a superimposition of path confluences involving the original 22 suspect addresses (encircled in green), and the 23 new addresses identified in the process. We also highlight (in blue) the confluences in which wallet address *1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy* indicated as *uy* is involved.

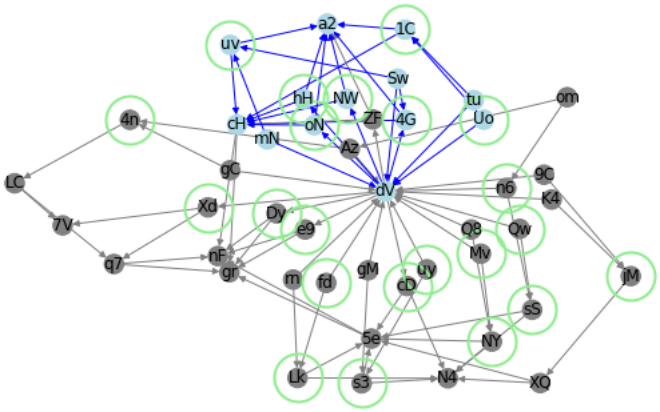


FIGURE 6.14: Same as Fig. 6.13, but here we highlight the confluences in which wallet address *1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2* indicated as *a2* is involved.

with. This thus yields a new group of 45 suspect addresses. We note that the existence of such a tightly connected group is not in itself a definitive proof that this group of addresses was indeed the one used for collecting the extortion amount: it is only indicative of the fact that this group of addresses - the originally suspected ones, as well as the other addresses they are liaising with, are controlled by the same entity (it could be a single person or a group).

In Fig. 6.13 we show a superimposition of several such chains of path confluences (in blue) involving the wallet address *1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3-zY5uy* indicated by the last letters *uy*. This is one of the two addresses singled out from Tables 6.2 and 6.3. This address is one of the 20 in-addresses from Table 6.2,

and 9 other in-addresses from this table appear as part of these confluences (*cD*, *s3*, *Xd*, *Dy*, *e9*, *4G*, *hH*, *NW* and *oN*). Also 3 out-addresses from the same table appear (*N4*, *nF* and *s3*). Note the appearance of *s3* twice. Furthermore, while *uy* was an original suspect address, the path confluence around it contains 8 of the 23 new suspicious addresses (these are the addresses not encircled in green).

The superimposition of path confluences involving the other outstanding address *1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2* labelled by the last letters *a2* is shown in Fig. 6.14. We observe that this new path confluence also includes *cH*, *dV*, but further adds *mN*, *tu* and *Sw* to the list of new addresses to look at, thus between the two of them, *1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy* and *1JujBBkRGAEm7JvdnCDfGw939cEtbuuWa2* implicate about half of the 23 new addresses.

Given that none of the other originally suspected addresses seem to have such behavior patterns and appear to be operating in isolation, while the group of 22 addresses and other associated 23 addresses as identified using Algorithm 6 are operating in a manner which resemble laundering activity, we assume (with the above caveat) these 45 (22+23) addresses to be involved in collecting and laundering the extortion money.

When we study the transactions in which the 45 addresses are involved in, and eliminate 1.05฿ transactions among this group of addresses, we identify 18 payments that are made by other unrelated wallet addresses. We use this as an indicator of the amount (18.9฿) of money collected by the specific extortion campaign (under the above indicated assumption).

As a point of reference for our proposed approaches, we provide results derived by application of some of the common heuristics [27, 31, 153] for address aggregation (which we apply for the current study[5]) are as follows. For transactions with multiple inputs and one or two outputs, the addresses corresponding the multiple inputs are aggregated together. Furthermore, for such transactions, if there is only one (distinct address) output then the corresponding address is clubbed with the input addresses, while if there are two (distinct address) outputs, then the address receiving lower of the two amounts is clubbed with the input addresses. If the two outputs are of equal amount, but only one of them is assigned to a hitherto unused address, then it is chosen to be clubbed with the input addresses. For single input single output transactions, the involved addresses are also grouped together, and

[5]The cited related works have certain minor variations amongst themselves. We thus specify the particular set of heuristics used in this work.

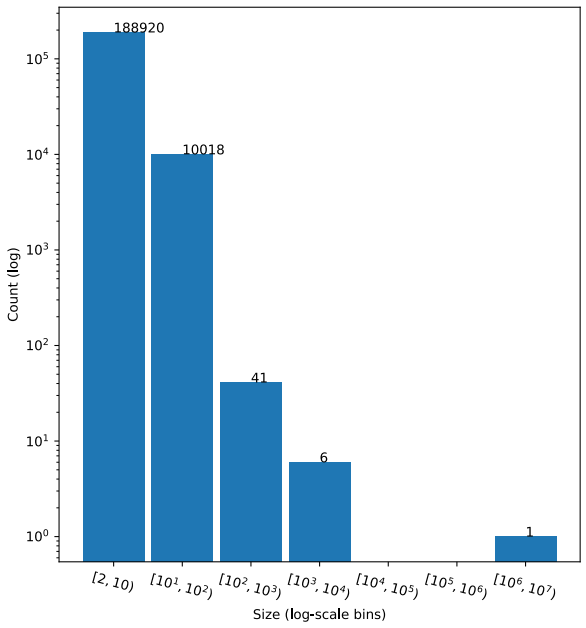


FIGURE 6.15: Histogram of the size of groups of aggregated wallet addresses (log-log scale).

for single input 2 output transactions, the identical heuristics described above for clubbing input and output addresses apply. Groups so formed across transactions, if they have any common addresses among themselves, are merged together.

We apply the above described heuristics for the same period (13 August 2018 to 6 September 2018), to carry out address aggregation of the wallet addresses.

Histograms of the size distributions of the aggregated address groups is shown in Figure 6.15. The size of the largest group is 2772343. When we matched the group of 22 suspected addresses (from Section 6.5.2), we find that in fact all of them are members of this largest aggregated address group. Not surprisingly, the 23 new suspicious addresses identified by our technique are also member of the largest aggregated address group. Thus, one may say that the address aggregation heuristics yield full recall compared to our result. However, given the very large pool of irrelevant addresses, the result from such heuristics is of little practical value in terms of Bitcoin forensics.

Such Bitcoin wallet address aggregation heuristics are inaccurate, and are based on assumptions and simplifications of Bitcoin wallet usage, which are not always true. The aforementioned heuristics totally ignore the multi-input/output

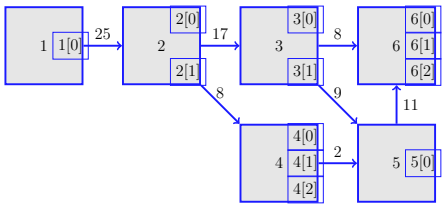


FIGURE 6.16: The transaction network corresponding to the transactions from Table 2.2 (in Chapter 2).

transactions, and the scope of these heuristics is thus confided to aggregating addresses from non-multi-input/output transactions. But even for the family of transactions which fit the scope, in general, usage of such heuristics may not yield full recall (they do not identify all the addresses that belong to same entity), nor are they precise (they may aggregate addresses that in fact belong to different entities). Furthermore, in our current study, we have noticed that, aware of such heuristics being used for address aggregation, people deliberately carry out transactions (e.g., both the output addresses belong to the same user, but the aggregation heuristic follows the trail of the smaller output amount, ignoring the address receiving the larger amount) to muddle the results from such heuristics.

6.5.2.5 Estimation of the Total Money Controlled

Having estimated the amount of money the extortionists received based on the particular amount of 1.05฿ per blackmail victim, we wanted to determine how much money they might be handling during the same time period - be it money they may have received under the same blackmail campaign but by receiving other amounts, or money they received and handled for other reasons, but using the same set of 45 addresses we have previously identified. To that end, we need to study the transaction network, which we describe next, before continuing our investigation. We note that the same extortionists may be handling other groups of addresses at the same time, and the estimate here is limited to only the amount handled by the specific cluster of 45 addresses we identified, and for the specific period of time we studied. However, following the money flow over a longer period of time, using the methodology discussed in this chapter, one can discover more addresses, and repeat the exercise.

A transaction network, as used e.g. in [139], represents the flow of Bitcoins between transactions over time. Each vertex represents a transaction, and each

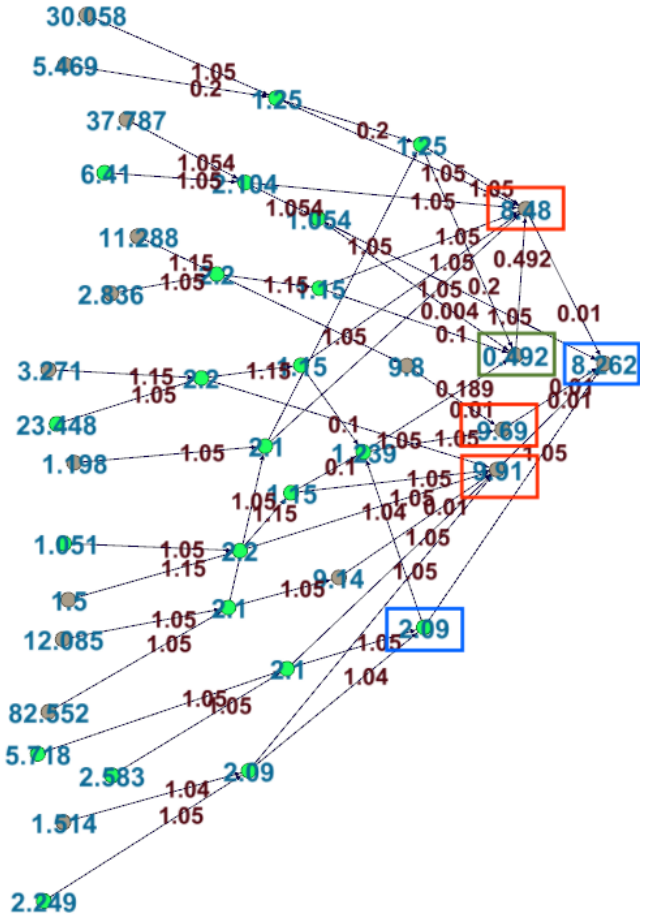


FIGURE 6.17: Transaction network: Only edges involving the 45 suspected Bitcoin wallet addresses are shown.

directed edge between a source vertex to a target vertex connects an output transaction of the source, to the target where it is used as input, as illustrated on Fig. 6.16: we have 6 transactions thus 6 nodes. The first transaction has no input, but one output, transaction 2 has one input and two outputs, each of them are inputs to respectively transactions 3 and 4.

In Fig. 6.17, we show a subset of the transaction graph for the period of 2015-08-24 08:07:13 to 2015-08-27 00:18:11 GMT. We only show those transaction edges that involve at least one of the 45 suspected addresses (the subset of transaction graph can be identified by using Algorithm 7 on transaction network.). The transaction nodes which contain at least one of the original 22 suspected addresses are shown in green, and the others are shown in gray. The total amounts of the transactions, and the specific amount of Bitcoins which are flowing from one transaction are indicated over the nodes and edges respectively, and have been rounded to three digits after decimal for perspicuity. The transactions highlighted by the

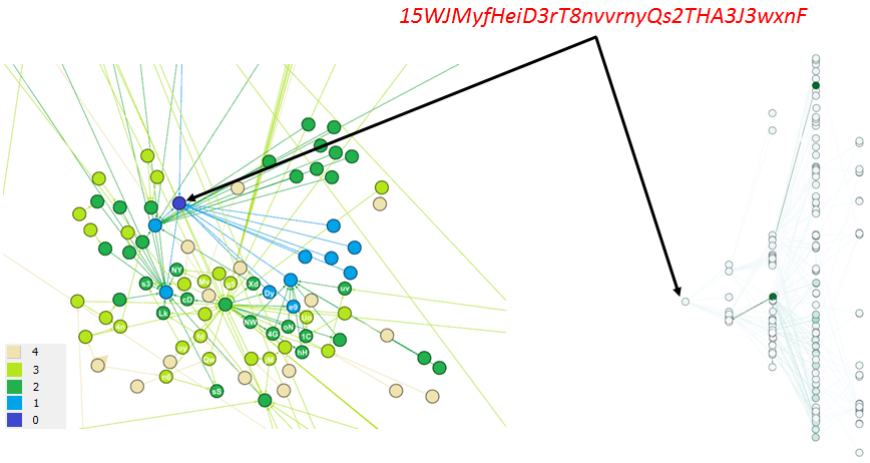


FIGURE 6.18: Tracing expected money flows with color indication of node level in rooted tree, shown on the left, that constructs spanning tree of (wallet) addresses, on the right.

red rectangles in Fig. 6.17 are sinks where the suspected addresses are aggregating their funds. Summing the corresponding amounts, we obtain an initial estimate of 36.56฿ as the sum of money the suspect addresses are handling. Note that, we overlook the amount in the transactions encompassed in green rectangle to avoid double counting, because the money flows further to the node on top. However, we also see that our initial estimate missed 2.09฿ (including which, the volume of money the 45 suspect nodes were handling during the time period of interest can be estimated as 38.65฿) from the bottom transaction highlighted by a blue rectangle, because we did not consider it as a sink. Yet, the rightmost transaction also encompassed in blue includes this amount but also partial amounts from other nodes we considered as sinks. As mentioned above, the study can be carried out for a larger time-window to get a better estimate of the overall money being handled by the extortionists, by identifying further addresses that can be aggregated with the initial group of 45 addresses, in a manner similar to how we discovered 23 new addresses based on the initial shortlist of 22 suspects.

Lastly, we apply our Bitcoin flow traceback mechanism on the address, namely *15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF* which appears multiple times in Table 6.3. In Figure 6.18, we report our result with (directed) distance of 4. On the right, the query node is in darkest blue. On the left, we show the spanning tree which has our query node as root, and nodes are colored according to their expected aggregated amount of Bitcoin flow through them. The darker the node

color, the higher the amount of Bitcoin. We discard the amount of Bitcoin passing through the root node. We then observe that there are 10 addresses which have exactly 1.05฿ passing through them. Among these 10 addresses, 6 are the already shortlisted suspect addresses. This gives us an idea that some suspected addresses are passing points for moving extorted money. Also, there are two darkest green color addresses: (1) *1J4s64FZPW8uRE5NvDFws39gE1ga4XGnSy* with 8.000221039฿; which was not yet discovered by the other algorithms. (2) The second darkest colored node is *1C7PDYzjRDqomyywDHEqx9huYoYQoGYgdV* with 7.735293584฿; which was in fact already identified previously as a centre of connection among close-knit suspected addresses (see Figure 6.12b, the leftmost red node). Compared to other nodes, these two expected amounts of Bitcoin are closed to actual amount of Bitcoin, 8.25฿, received by the root node. These newly identified nodes can be considered as epicentre nodes of interest for further investigations repeating the same methodology.

6.6 Summary

Motivated by the escalation of extortion scams, using cryptocurrencies to receive and launder the money, this chapter looks at possible ways to extend known heuristic for detecting/aggregating perpetrators' wallet addresses for the Bitcoin network. The three main ingredients of the proposed methodology are (1) detecting outliers with respect to two measures, the expected and the normalized expected path lengths between two nodes, (2) proposing an algorithm to track flow confluences between node addresses of interest, (3) and design a visualization and analytics tool (BiVA) for exploratory analysis of the Bitcoin network. The 2015 Ashley Madison extortion scam has been analysed as our case study. We further evaluated the amount of money involved both in the scam itself, but also estimated a lower bound on the amount of money the suspicious addresses were controlling altogether during the time window for which we conducted our study.

While mixing services had already appeared long before August 2015, this group of perpetrators did not deem necessary to cover their tracks through mixing services, a scenario that may not hold true anymore for more recent scams. We expect our flow path confluence based approach to be robust even in the presence of mixing. Nevertheless, this is yet to be validated by studying if and up to

what extent, accuracy of the path confluence based address aggregation mechanism deteriorates in the presence of mixing, and accordingly refine the technique if necessary. Finally, abstracting out the principles and adapting them to analyse other cryptocurrencies is another interesting direction.

Chapter 7

Conclusion and Future Directions

The study of the Bitcoin network originally motivated our investigations of graphs based on an (information) flow based paradigm. Specifically, we explored general purpose graph analysis and algorithms, and demonstrated their versatility by applying them over a wide range of real world graphs.

Motivated by the escalation of extortion scams using cryptocurrencies to receive and launder money, we also delved into graph analysis algorithms specific to Bitcoin forensics.

Overall, the contributions of this thesis are thus two-fold: graph analysis algorithms and their application to Bitcoin forensics.

For the *first aspect* (graph analysis algorithms), we address challenges in analyzing flow based networks with the following approaches:

A node level analysis - entropic centrality measure. We define a split-and-transfer entropic centrality measure, first with a path (no cycles) based approach, which provides element-level analysis in a flow based network, where a flow splits while conserving volume of the flow (Chapter 3). This centrality measure allows us to identify important nodes which distributed signification amount of information over a portion of network. We also demonstrate that in the special case where flows split uniformly at random, the centrality measures are identical to the result of the original work (that assumed flow by transfers only) we extend [18]. We demonstrate the versatility of this generalized split-and-transfer entropic centrality model by applying it to several networks, particularly a Maine airport dataset [19], a company cross-shareholding dataset [20] and a Bitcoin network dataset [22]. The case studies demonstrate one important property of the proposed entropic centrality approach, namely, it captures well the transitive aspect of influence or spread.

That is, a node even with low-degree may gain high centrality if it connects to a hub with high spread.

The condition that flows cannot have cycles was relaxed in [5], which allowed the use of a Markov model which significantly simplifies the analytical model as well as the computational overheads. The flow in this case is modelled as a random walker starting at v, and reaching other nodes with a probability inversely proportional to the number of neighbors a node has. The walker walks for t times, and there is an entropic centrality $C_H^t(v)$ for every choice of vertex v and time t. The Markov entropic centrality was used in [5] as an ingredient for a clustering algorithm. We revisit this reinterpreted model of Markov entropic centrality, and extend it to capture weighted and directed graphs (Chapter 4). We used Cocaine network [6] and Bitcoin subgraphs [21, 22] to expose the concepts and to validate the efficacy of our centrality computation.

On these, there are three obvious future directions of study: (1) explore scaling factors (to represent edge weights or a priori node importance) for a flow so as to best capture the underlying real world phenomenon being abstracted; (2) improve efficiency for computing entropic centrality for large networks, for instance, for the first model, we want to observe how the threshold currently being used in our computations to terminate the search path affects weighted entropy centrality; while for the second, explore the space-time trade-offs of using sparse matrix representation of the graph; (3) explore whether it is possible, and if so, then how to efficiently recompute entropic centrality for a graph leveraging on prior computation, where there are incremental changes in the graph.

A group level analysis - community detection. We design graph clustering based on confined circulation of probabilistic flows (see Chapter 4). Identifying Bitcoin flow confinement was again a driving factor behind the exploration of such a clustering algorithm. We start by finding nodes with low entropic centralities, and then look at the probability to visit neighbors of such nodes, and cluster these probabilities. This identifies neighbors of low entropic nodes where the flow is most likely to go, in a sense identifying local communities. We note that this model is applicable to all combinations of graphs (un/directed and un/weighted graph).

We also analyzes the network with a distance metric, and demonstrated one way to adapt Renyi entropy measure [43] for graph clustering, and proposed a practical clustering algorithm using simulated annealing [180] (in Chapter 5). Though randomized algorithms such as simulated annealing are supposed to (suboptimally)

accomplish computationally heavy tasks at a reasonable computational cost, and the simulated annealing does indeed provide tremendous reduction with respect to an exhaustive search for optimal, our experiments show that it nevertheless may not be practical for large graphs. We have started investigating hierarchical algorithms as a means to address the scalability issue. Furthermore, for this approach, accommodating directed or weighted graphs are open challenges, that need further attention.

A network level analysis - average path length in directed power-law graphs. We extend the work from [44] that studies the expected behavior of (power-law) graphs in terms of average path length, to the case of directed graphs (see Chapter 6). We derive an exact formula for calculating average path length for random directed networks with power-law in-degree/out-degree distributions.

One possible direction of this study is to explore boundary effects when analyzing a subgraph which is extracted by cutting the graph at a given radius, which is often the case in practice.

For the *second aspect* (application of graph analysis to Bitcoin forensics) of this thesis, we showcase how some of these general graph algorithms, particularly, our proposed entropic centrality measure (at a macroscopic level), our proposed flow confinement based community detection (at a macroscopic level) and our average path length analysis in directed network (at a microscopic level, to detect outliers) can be applied in the context of Bitcoin network analysis on subgraphs and nodes of interest. We furthermore design more Bitcoin specific algorithms as well as a tool (BiVA) for visualization aided exploratory analysis of the Bitcoin network and for Bitcoin forensics (see Chapter 6).

Macroscopic analysis. In the particular context of Bitcoin analysis, both of our above entropic centrality measures can be used to identify shortlist of nodes or hubs of interest for further investigations. The experiment on Bitcoin network (with reverse directionality) related to Ashley Madison incident shows that the proposed mechanism provides reasonable shortlist of nodes which comprises original suspect addresses (see Chapter 3). Moreover, we observe that the graph clustering, based on confined circulation of probabilistic flows, appears to be well suited to capture the probabilistic flow of Bitcoins because of mixing in the directed address network, where multi-input multi-output transactions are modelled by assuming that the Bitcoin flow goes from every incoming address to every outgoing one. The experiments on Bitcoin network related to WannaCry [33] and Ashley Madison

extortion incidents [32] (see Chapter 4), illustrate that the proposed algorithm helps discover community of nodes which can be used as a new shortlist of suspect addresses for further investigation. Furthermore, simulation based experiments on simplistic synthetic data with ground truth also provide a preliminary validation of the flow based clustering algorithm's robustness in the presence of multiple mixing events.

For more rigorous validation of the two novel flow-based algorithms in the context of Bitcoin analysis, two possible approach would be to (i) cross-validate the results from the algorithms against results from third party studies of other scams, and (ii) to generate more complex and larger synthetic dataset with ground-truth for observing precision and recall of macroscopic information derived by the proposed algorithms.

Microscopic analysis. We look at possible ways to gain more insights within subgraphs centred around node addresses of interest. Specifically, we design two mechanisms. We first design an outliers detection mechanism by applying network level analysis with respect to two measures, the expected and the normalized expected path lengths between two nodes in Bitcoin (directed) network. We propose a mechanism for address aggregation by tracking flow confluence between node addresses of interest. We also carry out a case study (dataset from the 2015 Ashley Madison extortion scam [32]) to illustrate applicability of our proposed methodology. Furthermore, we design Bitcoin flow traceback/traceforward heuristics to evaluate the amount of money controlled by the suspect wallet addresses, which can be used to estimate the amount of money the suspicious addresses were controlling (this could include money from other sources than the specific scam being studied) altogether during the time window for which we conduct our study.

An obvious line of future work to be investigated is the use of the same microscopic information analysis but for other scams. Once it becomes known that simple heuristics such as detecting a particular amount of money (1.05฿ in our case study) can leak information, perpetrators are quick to update their scams, and in fact, even for the Ashley Madison scam, subsequent amounts of Bitcoin requested became more random. Likewise, we expect more extensive use of mixing by people laundering money, in order to cover their tracks.

More experiments are needed to validate or alternatively identify the limitations of our flow confluence based mechanism in the presence of numerous mixings, and accordingly refine the technique as necessary. One idea we want to explore is

the use of a Bayesian network [193] to infer the likelihoods of certain wallet addresses to be belonging to a given entity, to determine a degree of confidence on the aggregation, and apply the Bayesian network tracing technique in conjunction with existing heuristics for taint analysis [194], address aggregation approaches [195] and the path confluence address aggregation mechanism [47].

Finally, abstracting out the principles and adapting them to analyze other cryptocurrencies is another interesting direction.

Publications to Date

Patents

- A. Datta, F. Oggier, S. Phetsouvanh. Information flow based graph clustering algorithms. *Singapore Provisional Patent Application No. 102018102185*, applied on 16 November 2018.

Conference Proceedings

- S. Phetsouvanh, F. Oggier, and A. Datta. EGRET: Extortion graph exploration techniques in the Bitcoin network. In *IEEE ICDM Workshop on Data Mining in Networks (DaMNet)*, 2018.
- F. Oggier, S. Phetsouvanh, and A. Datta. BiVA: Bitcoin network visualization and analysis. In *IEEE ICDM Demo on Data Mining Workshop (ICDMW)*, 2018.
- F. Oggier, S. Phetsouvanh, and A. Datta. Entropy-based graph clustering - A simulated annealing approach. In *International Symposium on Information Theory and Its Applications (ISITA)*, 2018.
- F. Oggier, S. Phetsouvanh, and A. Datta. Entropic centrality for non-atomic flow networks. In *International Symposium on Information Theory and Its Applications (ISITA)*, 2018.

Manuscripts under submission for review

Submissions to journals

- S. Phetsouvanh, A. Datta, F. Oggier. Analysis of multi-input multi-output transactions in the Bitcoin network. *under submission*, 2019.
- F. Oggier, S. Phetsouvanh, A. Datta. A split-and-transfer flow based entropic centrality. Manuscript submitted for publication. *under submission*, 2018.
- S. Phetsouvanh, A. Datta, A. Tiu. On defending a whistleblower submission system against denial of service attacks. *under submission*, 2018.

- F. Oggier, S. Phetsouvanh, A. Datta. Centrality and clustering for weighted directed graphs - An entropy-based approach. *under submission*, 2018.

Bibliography

[1] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004. xix, 15, 17

[2] M. Barrère, R. V. Steiner, R. Mohsen, and E. C. Lupu. Tracking the bad guys: An efficient forensic methodology to trace multi-step attacks using core attack graphs. In *Network and Service Management (CNSM), 2017 13th International Conference on*, pages 1–7. IEEE, 2017. xix, 20

[3] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *European Symposium on Research in Computer Security*, pages 345–364. Springer, 2014. xix, 23, 24

[4] P. K. Sen. Estimates of the regression coefficient based on kendall's tau. *Journal of the American statistical association*, 63(324):1379–1389, 1968. xx, 49, 62

[5] A. G. Nikolaev, R. Razib, and A. Kucheriya. On efficient use of entropy centrality for social network analysis and community detection. *Social Networks*, 40:154–162, 2015. xxi, xxvi, 5, 7, 13, 16, 17, 20, 34, 64, 67, 69, 70, 71, 78, 80, 81, 82, 87, 88, 95, 158

[6] J. Coutinho. Cocaine dealing network. *Accessed on 02 December 2018 at URL: https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/ cocainedealingnatarajan*. xxi, 5, 20, 77, 82, 83, 158

[7] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009. xxii, xxvi, 7, 15, 16, 17, 68, 90, 91, 92, 95, 96, 97, 99, 100

[8] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory*

and experiment, 2008(10):P10008, 2008. xxii, xxvi, 27, 68, 90, 91, 92, 95, 96, 97, 99, 100

[9] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006. xxii, xxvi, 16, 17, 68, 90, 91, 92, 95, 96, 97, 99, 100

[10] M. Bastian, S. Heymann, M. Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *Icwsm*, 8(2009):361–362, 2009. xxii, xxiii, 26, 86, 91, 96, 99, 100, 119, 145

[11] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003. xxii, 95, 96, 108, 123

[12] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002. xxii, 81, 95, 98, 100, 101, 108, 117

[13] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977. xxiii, 77, 78, 81, 82, 115, 117, 118

[14] National Research Council (2005). *Network Science*. Washington, DC: The National Academies Press, available at URL: https://doi.org/10.17226/11516, accessed 06 January 2019. 1

[15] E. Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(1): 181–186, 1990. 1

[16] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. 3, 5, 8, 22, 28, 55

[17] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003. 3

[18] F. Tutzauer. Entropy as a measure of centrality in networks characterized by path-transfer flow. *Social networks*, 29(2):249–265, 2007. 5, 7, 13, 31, 32, 33, 34, 36, 64, 69, 157

[19] F. Oggier, S. Phetsouvanh, and A. Datta. Maine airport network in january 2018. *Accessed on 02 December 2018 at URL: https://doi.org/10.21979/N9/WM0K5W, DR-NTU (Data), V1*, . 5, 45, 157

[20] H. Dastkhan and N. Shams Gharneh. Determination of systemically important companies with cross-shareholding network analysis: A case study from an emerging market. *International Journal of Financial Studies*, 4(3):13, 2016. 5, 50, 52, 157

[21] F. Oggier, S. Phetsouvanh, and A. Datta. A 178 node directed bitcoin address subgraph. *Accessed on 02 December 2018 at URL: https://doi.org/10.21979/N9/TJMQ8L, DR-NTU (Data), V1*, . 5, 78, 85, 88, 115, 118, 158

[22] F. Oggier, S. Phetsouvanh, and A. Datta. A 4571 node directed weighted bitcoin address subgraph. *Accessed on 02 December 2018 at URL: https://doi.org/10.21979/N9/IEPBXV, DR-NTU (Data), V1*, . 27, 58, 78, 92, 157, 158

[23] F. Oggier, S. Phetsouvanh, and A. Datta. A 5026 node directed bitcoin address subgraph. *Accessed on 14 January 2019 at URL: https://doi.org/10.21979/N9/JQUY8Q, DR-NTU (Data), V1*, . 62

[24] F. Oggier, S. Phetsouvanh, and A. Datta. A 5251 node directed bitcoin address subgraph. *Accessed on 14 January 2019 at URL: https://doi.org/10.21979/N9/HUBNNX, DR-NTU (Data), V2*, . 5, 62

[25] M. Spagnuolo, F. Maggi, and S. Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security*, pages 457–468. Springer, 2014. 5, 26, 28

[26] M. Lischke and B. Fabian. Analyzing the bitcoin network: The first four years. *Future Internet*, 8(1):7, 2016. 26

[27] M. Harrigan and C. Fretter. The unreasonable effectiveness of address clustering. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 368–373. IEEE, 2016. 5, 29, 130, 149

[28] K. Liao, Z. Zhao, A. Doupé, and G. J. Ahn. Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin. In *Electronic Crime Research (eCrime), 2016 APWG Symposium on*, pages 1–13. IEEE, 2016. 27

[29] H. Kuzuno and C. Karam. Blockchain explorer: An analytical process and investigation environment for bitcoin. In *Electronic Crime Research (eCrime), 2017 APWG Symposium on*, pages 9–16. IEEE, 2017. 5, 28

[30] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013. 5, 28, 105

[31] J. D. Nick. Data-driven de-anonymization in bitcoin. Master's thesis, ETH-Zürich, 2015. 5, 28, 105, 130, 149

[32] T. Nishimura. Does blackmailing pay? signs on the bitcoin blockchain of responses to ashley madison extortion emails. *Accessed on 14 July 2018 at URL: https://blog.cloudmark.com/2015/09/01/does-blackmailing-pay-signs-in-bitcoin-blockchain-of-responses-to-ashley-madison-extortion-emails/.* 6, 92, 130, 141, 160

[33] K. Collins. The hackers behind the wannacry ransomware attack have finally cashed out. *Accessed on 06 January 2019 at URL: https://qz.com/1045270/wannacry-update-the-hackers-behind-ransomware-attack-finally-cashed-out-about-140000-in-bitcoin/.* 6, 27, 159

[34] S. E. Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007. 7, 14

[35] F. D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013. 14, 15

[36] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008. 15, 17

[37] Y. Kim, S. W. Son, and H. Jeong. Finding communities in directed networks. *Physical Review E*, 81(1):016103, 2010. 15, 17

[38] Y. Bian, J. Ni, W. Cheng, and X. Zhang. Many heads are better than one: Local community detection by the multi-walker chain. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 21–30. IEEE, 2017. 16, 17, 68, 75

[39] Y. Bian, Y. Yan, W. Cheng, W. Wang, D. Luo, and X. Zhang. On multi-query local community detection. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 9–18. IEEE, 2018. 16, 17

[40] J. David Cruz, C. Bothorel, and F. Poulet. Entropy based community detection in augmented social networks. In *Computational aspects of social networks (cason), 2011 international conference on*, pages 163–168. IEEE, 2011. 16, 17, 20

[41] X. Deng, G. Li, M. Dong, and K. Ota. Finding overlapping communities based on markov chain and link clustering. *Peer-to-Peer Networking and Applications*, 10(2):411–420, 2017. 16, 17

[42] S. H. Bae, D. Halperin, J. West, M. Rosvall, and B. Howe. Scalable flow-based community detection for large-scale network analysis. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 303–310. IEEE, 2013. 7, 15, 17, 68

[43] E. Gokcay and J. C. Principe. Information theoretic clustering. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2):158–171, 2002. 7, 18, 109, 110, 113, 158

[44] A. Fronczak, P. Fronczak, and J. A. Hołyst. Average path length in random networks. *Physical Review E*, 70(5):056110, 2004. 8, 18, 19, 132, 134, 159

[45] F. Oggier, S. Phetsouvanh, and A. Datta. Entropic centrality for non-atomic flow networks. In *International Symposium on Information Theory and Its Applications (ISITA)*, 2018. 8, 35, 42

[46] F. Oggier, S. Phetsouvanh, and A. Datta. Entropy-based graph clustering - a simulated annealing approach. In *International Symposium on Information Theory and Its Applications (ISITA)*, 2018. 8

[47] S. Phetsouvanh, F. Oggier, and A. Datta. Egret: Extortion graph exploration techniques in the bitcoin network. In *IEEE ICDM Workshop on Data Mining in Networks (DaMNet)*, 2018. 8, 58, 60, 105, 161

[48] F. Oggier, S. Phetsouvanh, and A. Datta. Biva: Bitcoin network visualization and analysis. In *IEEE ICDM Demo on Data Mining Workshop (ICDMW)*, 2018. 8, 134

[49] J. Travers and S. Milgram. The small world problem. *Phychology Today*, 1 (1):61–67, 1967. 9

[50] J. A. Davis and S. Leinhardt. *The structure of positive interpersonal relations in small groups.* ERIC, 1967. 9

[51] M. S. Granovetter. The strength of weak ties. In *Social networks*, pages 347–367. Elsevier, 1977.

[52] N. Notebook. The international network for social network analysis. *CONNECTIONS*, 1(2), 1978. 9

[53] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994. 9, 20

[54] C. Aggarwal and K. Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)*, 47(1):10, 2014. 9

[55] S. Kadry M. Z. Al-Taie. *Python for Graph and Network Analysis.* Cham, Switzerland : Springer, 2017. 10

[56] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 10, 12, 16, 21, 48, 68

[57] M. B. Era. *Structural Holes: The Social Structure of Competition.* Cambridge, Harvard University Press, 1992. 10, 20

[58] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski. Centrality indices. In *Network analysis*, pages 16–61. Springer, 2005. 11

[59] M.E.J. Newman. The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, 2009. 12

[60] P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematics Sociology*, 2:113–120, 1972. 12

[61] P. Bonacich and P. Lloyd. Eigenvector-like measures of centrality for asymmetric relations. *Social networks*, 23(3):191–201, 2001. 12, 31, 48

[62] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 1953. 12, 31, 48

[63] M. Benzi and C. Klymko. On the limiting behavior of parameter-dependent network centrality measures. *SIAM Journal on Matrix Analysis and Applications*, 36(2):686706, 2015. 12

[64] S. P. Borgatti. Centrality and network flow. *Social networks*, 27(1):55–71, 2005. 12, 32, 49

[65] U. Brandes and D. Fleischer. Centrality measures based on current flow. In *Annual symposium on theoretical aspects of computer science*, pages 533–544. Springer, 2005. 12

[66] S. Gao, Y. Wang, Y. Gao, and Y. Liu. Understanding urban traffic-flow characteristics: a rethinking of betweenness centrality. *Environment and Planning B: Planning and Design*, 40(1):135–153, 2013.

[67] D. Gómez, J. R. Figueira, and A Eusébio. Modeling centrality measures in social network analysis using bi-criteria network flow optimization problems. *European Journal of Operational Research*, 226(2):354–365, 2013. 20

[68] A. Kazerani and S. Winter. Can betweenness centrality explain traffic flow. In *12th AGILE International Conference on Geographic Information Science*, pages 1–9, 2009. 12

[69] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977. 13

[70] M. E. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005. 13, 31

[71] E. Zio and R. Piccinelli. Randomized flow model and centrality measure for electrical power transmission network analysis. *Reliability Engineering & System Safety*, 95(4):379–385, 2010. 13, 31

[72] Y Li, J Fan, Y Wang, and K-L Tan. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 30(10): 1852–1872, 2018. 13, 31

[73] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003. 13

[74] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[75] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *2010 IEEE international conference on data mining*, pages 88–97. IEEE, 2010.

[76] Y. Singer. How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 733–742. ACM, 2012. 13

[77] T. Qiao, W. Shan, and C. Zhou. How to identify the most powerful node in complex networks? a novel entropy centrality approach. *Entropy*, 19(11): 614, 2017. 13

[78] T. Qiao, W. Shan, G. Yu, and C. Liu. A novel entropy-based centrality approach for identifying vital nodes in weighted networks. *Entropy*, 20(4): 261, 2018. 13

[79] T. Nie, Z. Guo, K. Zhao, and Z. M. Lu. Using mapping entropy to identify node centrality in complex networks. *Physica A: Statistical Mechanics and its Applications*, 453:290–297, 2016. 13

[80] M. Stella and M. De Domenico. Distance entropy cartography characterises centrality in complex networks. *Entropy*, 20(4):268, 2018. 14

[81] A. Zareie, A. Sheikhahmadi, and A. Fatemi. Influential nodes ranking in complex networks: An entropy-based approach. *Chaos, Solitons & Fractals*, 104:485–494, 2017. 14

[82] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. 14

[83] R. Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

[84] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992. 14

[85] A. Schenker. *Graph-theoretic techniques for web content mining*. PhD thesis, Tampa, FL, USA, 2003. 14, 27

[86] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016. 15

[87] J. Han, W. Li, Z. Su, L. Zhao, and W. Deng. Community detection by label propagation with compression of flow. *The European Physical Journal B*, 89 (12):272, 2016. 16, 17, 68

[88] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76 (3):036106, 2007. 16, 68

[89] R. Andersen and K. J. Lang. Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, pages 223–232. ACM, 2006. 16, 17, 68

[90] R. Venkatesaramani and Y. Vorobeychik. Community detection by information flow simulation. *arXiv preprint arXiv:1805.04920*, 2018. 16, 17

[91] L. M. Smith, L. Zhu, K. Lerman, and A. G. Percus. Partitioning networks with node attributes by compressing information flow. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(2):15, 2016. 16, 17

[92] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002. 17

[93] J. A. Peacock. *Cosmological physics*. Cambridge university press, 1999. 17

[94] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003. 17

[95] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004. 17

[96] X.M. Wu, Z. Li, A. M. So, J. Wright, and S.F. Chang. Learning with partially absorbing random walks. In *Advances in Neural Information Processing Systems*, pages 3077–3085, 2012. 17

[97] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015. 18

[98] J. Shi and J. Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000. 18

[99] G. W. Flake, S. Lawrence, C. L. Giles, et al. Efficient identification of web communities. In *KDD*, volume 2000, pages 150–160, 2000. 18

[100] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5): 75–174, 2010. 18

[101] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663, 2004. 18

[102] D. Pfitzner, R. Leibbrandt, and D. Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(3):361, 2009. 18, 75, 88, 90, 97

[103] J. Meier, P. Tewarie, A. Hillebrand, L. Douw, B. W. van Dijk, S. M. Stufflebeam, and P. Van Mieghem. A mapping between structural and functional brain networks. *Brain connectivity*, 6(4):298–311, 2016. 18

[104] M. Malladi, V. Mithun, K. B. N. Naveen, N. Punith, J. M. Roogi, M. J. Sarma, and S. K. Routray. Statistical analysis of path length in optical networks. In *Innovations in Electronics and Communication Engineering*, pages 459–466. Springer, 2019. 18

[105] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005. 18

[106] P. ERDdS and A. R&WI. On random graphs i. *Publ. Math. Debrecen*, 6: 290–297, 1959. 18

[107] L. De Haan and A. Ferreira. *Extreme value theory: an introduction*. Springer Science & Business Media, 2007. 19

[108] H. Rinne. *The Weibull distribution: a handbook*. Chapman and Hall/CRC, 2008. 19

[109] D. Guo, H. Yin, C. Li, and X. Zhang. Average hopcount of the shortest path in tree-like components with finite size. *Physica A: Statistical Mechanics and its Applications*, 2019. 19

[110] M. E. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical review E*, 64 (2):026118, 2001. 19

[111] F. Chen, Z. Chen, X. Wang, and Z. Yuan. The average path length of scale free networks. *Communications in Nonlinear Science and numerical simulation*, 13(7):1405–1410, 2008. 19

[112] L. Ying, C. Hong-Duo, S. Xiu-Ming, and R. Yong. An estimation formula for the average path length of scale-free networks. *Chinese Physics B*, 17(7): 2327, 2008. 19

[113] G. Mao and N. Zhang. Analysis of average shortest-path length of scale-free network. *Journal of Applied Mathematics*, 2013. 19

[114] B. Chen and L. Cao. An optimized algorithm for calculating the average path length of complex network. In *Computational Intelligence and Design (ISCID), 2017 10th International Symposium on*, volume 1, pages 334–337. IEEE, 2017. 19

[115] S. A. Myers, A. Sharma, P. Gupta, and J. Lin. Information network or social network?: the structure of the twitter follow graph. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 493–498. ACM, 2014. 19, 20

[116] E. Mocibob, S. Martinčić-Ipšić, and A. Meštrović. Revealing the structure of domain specific tweets via complex networks analysis. In *Information*

and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 39th International Convention on, pages 1623–1627. IEEE, 2016.

[117] J. Villazon-Terrazas, S. Aparicio, and G. Alvarez. Study on twitter as a complex network. In *The Third International Conference on Digital Enterprise and Information Systems (DEIS2015)*, volume 54, 2015. 19, 20

[118] T. Matsumura, K. Iwasaki, and K. Shudo. Average path length estimation of social networks by random walk. In *Big Data and Smart Computing (BigComp), 2018 IEEE International Conference on*, pages 611–614. IEEE, 2018. 19

[119] D. Kondor, M. Pósfai, I. Csabai, and G. Vattay. Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PloS one*, 9(2):e86197, 2014. 19, 25, 26, 56, 58, 105, 130, 142

[120] H. Du, X. He, W. Du, and M. W. Feldman. Optimization of the critical diameter and average path length of social networks. *Complexity*, 2017. 19

[121] A. Meyerson and B. Tagiku. Minimizing average shortest path distances via shortcut edge addition. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 272–285. Springer, 2009.

[122] Z. Bilgin, M. Gunestas, O. Demir, and S. Buyrukbilen. Optimal link deployment for minimizing average path length in chain networks. In *International Conference on Wired/Wireless Internet Communication*, pages 348–359. Springer, 2016.

[123] A. Gozzard, M. Ward, and A. Datta. Converting a network into a small-world network: Fast algorithms for minimizing average path length through link addition. *Information Sciences*, 422:282–289, 2018. 19

[124] W. De Nooy, A. Mrvar, and V. Batagelj. *Exploratory social network analysis with Pajek*. Cambridge University Press, 2018. 20

[125] A. R. Mclntosh and F. Gonzalez-Lima. Structural equation modeling and its application to network analysis in functional brain imaging. *Human brain mapping*, 2(1-2):2–22, 1994. 20

[126] G. Wu, X. Feng, and L. Stein. A human functional protein interaction network and its application to cancer data analysis. *Genome biology*, 11(5):R53, 2010. 20

[127] H. Y. Shih. Network characteristics of drive tourism destinations: An application of network analysis in tourism. *Tourism Management*, 27(5):1029–1039, 2006. 20

[128] S. Saura, C. Estreguil, C. Mouton, and M. Rodríguez-Freire. Network analysis to assess landscape connectivity trends: application to european forests (1990–2000). *Ecological Indicators*, 11(2):407–416, 2011. 20

[129] S. Porta, P. Crucitti, and V. Latora. The network analysis of urban streets: a primal approach. *Environment and Planning B: planning and design*, 33 (5):705–725, 2006. 20

[130] M. K. Sparrow. The application of network analysis to criminal intelligence: An assessment of the prospects. *Social networks*, 13(3):251–274, 1991. 20

[131] A. P. Dawid and J. Mortera. Graphical models for forensic analysis. In *Handbook of Graphical Models*, pages 491–514. CRC Press, 2018. 20

[132] A. L. Terrettaz-Zufferey, F. Ratle, O. Ribaux, P. Esseiva, and M. Kanevski. Pattern detection in forensic case data using graph theory: Application to heroin cutting agents. *Forensic science international*, 167(2-3):242–246, 2007. 20

[133] M. Natarajan. Understanding the structure of a drug trafficking organization: a conversational analysis. *Crime Prevention Studies*, 11:273–298, 2000. 20

[134] I. Palmer, R. Campbell, and B. Gelfand. Exploring digital evidence with graph theory. In *Proceedings of the Conference on Digital Forensics, Security and Law*, pages 197–206. Association of Digital Forensics, Security and Law, 2017. 21

[135] L. Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 99(2):15–23, 2003. 21

[136] W. Wang. *A graph oriented approach for network forensic analysis*. PhD thesis, Digital Repository at Iowa State University, 2010. 21

[137] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016. 22

[138] G. Maxwell. Coinjoin: Bitcoin privacy for the real world. In *Post on Bitcoin forum*, 2013. 24, 29

[139] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013. 24, 26, 27, 29, 30, 151

[140] S. Bistarelli and F. Santini. Go with the-bitcoin-flow, with visual analytics. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 38. ACM, 2017. 26

[141] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun. Is bitcoin a decentralized currency? *IEEE security & privacy*, 12(3):54–60, 2014. 26

[142] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992. 26

[143] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013. 26

[144] A. Baumann, B. Fabian, and M. Lischke. Exploring the bitcoin network. In *WEBIST (1)*, pages 369–374, 2014. 26

[145] M. Möser and R. Böhme. Anonymous alone? measuring bitcoins second-generation anonymization techniques. In *EuroS&P Workshops*, pages 32–41, 2017. 27, 29, 30

[146] P. Koshy, D. Koshy, and P. McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, 2014. 29, 30

[147] M. Fleder, N. S Kester, and S. Pillai. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*, 2015. 27

[148] M. C. K. Khalilov and A. Levi. A survey on anonymity and privacy in bitcoin-like digital cash systems. *IEEE Communications Surveys & Tutorials*, 2018. 27

[149] B. Zheng, L. Zhu, M. Shen, X. Du, J. Yang, F. Gao, Y. Li, C. Zhang, S. Liu, and S. Yin. Malicious bitcoin transaction tracing using incidence relation clustering. In *International Conference on Mobile Networks and Management*, pages 313–323. Springer, 2017. 27

[150] D. Zambre and A. Shah. Analysis of bitcoin network dataset for fraud. *Unpublished Report*, 2013. 27

[151] J. J. Xu. Are blockchains immune to all malicious attacks? *Financial Innovation*, 2(1):25, 2016.

[152] H. U. Salvi and R. V. Kerkar. Ransomware: A cyber extortion. *Asian Journal for Convergence in Technology (AJCT)-UGC LISTED*, 2, 2016. 27

[153] T. H. Chang and D. Svetinovic. Improving bitcoin ownership identification using transaction patterns analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (99):1–12, 2018. 29, 108, 130, 149

[154] Y. Hong, H. Kwon, J. Lee, and J. Hur. A practical de-mixing algorithm for bitcoin mixing services. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, pages 15–20. ACM, 2018. 29

[155] G. Maxwell. Coinswap: Transaction graph disjoint trustless trading. *CoinSwap: Transactiongraphdisjointtrustlesstrading (October 2013)*, 2013. 29

[156] Stealth addresses. *Accessed on 02 December 2018 at URL: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-December/006982.html*. 29

[157] F. Parino, M. G. Beiró, and L. Gauvin. Analysis of the bitcoin blockchain: socio-economic factors behind the adoption. *EPJ Data Science*, 7(1):38, 2018. 29, 30

[158] P. L. Juhász, J. Stéger, D. Kondor, and G. Vattay. A bayesian approach to identify bitcoin users. *PloS one*, 13(12):e0207000, 2018. 29

[159] D. Ermilov, M. Panov, and Y. Yanovich. Automatic bitcoin address clustering. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*, pages 461–466. IEEE, 2017. 30

[160] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006. 48, 138

[161] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. 48, 116, 138

[162] M. Alamgir and U. Von Luxburg. Multi-agent random walks for local clustering on graphs. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 18–27. IEEE, 2010. 68

[163] Neo4J Developers. Neo4j. *Graph NoSQL Database [online]*, 2012. 68, 138

[164] S. Guiaşu. Weighted entropy. *Reports on Mathematical Physics*, 2(3):165–179, 1971. 73

[165] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct): 2825–2830, 2011. 75, 117

[166] A. Soliman, F. Rahimian, and S. Girdzijauskas. Stad: Stateful diffusion for linear time community detection. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1074–1085. IEEE, 2018. 90

[167] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002. 105

[168] J. Alstott, E. Bullmore, and D. Plenz. powerlaw: a python package for analysis of heavy-tailed distributions. *PloS one*, 9(1):e85777, 2014. 106, 142

[169] A. Rényi. On measures of entropy and information. Technical report, Hungarian Academy of Sciences, Budapest, Hungary, 1961. 109, 110

[170] J. A. Hartigan. *Clustering algorithms*. John Wiley and Sons, New York, 351 p, 1975. 109

[171] L. Faivishevsky and J. Goldberger. Nonparametric information theoretic clustering algorithm. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 351–358, 2010. 110

[172] M. Wang and F. Sha. Information theoretical clustering via semidefinite programming. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 761–769, 2011.

[173] A. C. Müller, S. Nowozin, and C. H. Lampert. Information theoretic clustering using minimum spanning trees. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 205–215. Springer, 2012.

[174] M. Sugiyama, G. Niu, M. Yamada, M. Kimura, and H. Hachiya. Information-maximization clustering based on squared-loss mutual information. *Neural Computation*, 26(1):84–131, 2014.

[175] G. Ver Steeg, A. Galstyan, F. Sha, and S. DeDeo. Demystifying information-theoretic clustering. In *International Conference on Machine Learning*, pages 19–27, 2014. 110

[176] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962. 111

[177] K. B. Petersen and M. S. Pedersen. The matrix cookbook tech. univ. denmark, kongens lyngby. Technical report, Denmark, Tech. Rep., Ver, 2012. 112

[178] R. Jenssen, K. E. Hild, D. Erdogmus, J. C Principe, and T. Eltoft. Clustering using renyi's entropy. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 1, pages 523–528. IEEE, 2003. 113

[179] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE transactions on communication technology*, 15(1):52–60, 1967. 113

[180] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983. 114, 158

[181] Y. C. Zhao, L. Tao, K. Thulasiraman, and M. N. S. Swamy. An efficient simulated annealing algorithm for graph bisectioning. In *Applied Computing, 1991.,[Proceedings of the 1991] Symposium on*, pages 65–68. IEEE, 1991. 114

[182] F. Rahimian, A. H. Payberah, S. Girdzijauskas, M. Jelasity, and S. Haridi. Ja-be-ja: A distributed algorithm for balanced graph partitioning. In *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 51–60. IEEE, 2013. 114

[183] F. Rahimian, A. H. Payberah, S. Girdzijauskas, M. Jelasity, and S. Haridi. A distributed algorithm for large-scale graph partitioning. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(2):12, 2015. 114

[184] T. Poranen. A simulated annealing algorithm for determining the thickness of a graph. *Information Sciences*, 172(1-2):155–172, 2005. 114

[185] A. Kose, B. A. Sonmez, and M. Balaban. Simulated annealing algorithm for graph coloring. *arXiv preprint arXiv:1712.00709*, 2017. 114

[186] A. Narayanan, E. Shi, and B. I. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1825–1834. IEEE, 2011. 114

[187] Montreal street gangs, 2016. URL `https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/montrealstreetgangs`. 126

[188] A. P. Guinand. On poisson's summation formula. *Annals of Mathematics*, pages 591–603, 1941. 133

[189] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products*. Academic press, 2014. 133

[190] Jupiter dashboard. *Accessed on 12 July 2018 at URL: https://jupyter-dashboards-layout.readthedocs.io/en/latest/*. 138

[191] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1433–1445. ACM, 2018. 138

[192] Bitcoin core. *Accessed on 12 Jan 2019 at URL: https://bitcoin.org/en/bitcoin-core/*. 138

[193] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997. 161

[194] S. Meiklejohn and C. Orlandi. Privacy-enhancing overlays in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 127–141. Springer, 2015. 161

[195] N. Conti, S. Kumar, C. Lal, and S. Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 2018. 161