AlphaCore: Data Depth based Core Decomposition

Friedhelm Victor
Technische Universität Berlin
Berlin, Germany
friedhelm.victor@tu-berlin.de

Cuneyt G. Akcora
University of Manitoba
Winnipeg, Canada
cuneyt.akcora@umanitoba.ca

Yulia R. Gel
UT Dallas
Richardson, USA
ygl@utdallas.edu

Murat Kantarcioglu
UT Dallas
Richardson, USA
muratk@utdallas.edu

ABSTRACT

Core decomposition in networks has proven useful for evaluating the importance of nodes and communities in a variety of application domains, ranging from biology to social networks and finance. However, existing core decomposition algorithms have limitations in simultaneously handling multiple node and edge attributes.

We propose a novel unsupervised core decomposition method that can be easily applied to directed and weighted networks. Our algorithm, AlphaCore, allows us to systematically and mathematically rigorously combine multiple node properties by using the notion of data depth. In addition, it can be used as a mixture of centrality measure and core decomposition. Compared to existing approaches, AlphaCore avoids the need to specify numerous thresholds or coefficients and yields meaningful quantitative and qualitative insights into the network structural organization.

We evaluate AlphaCore's performance with a focus on financial, blockchain-based token networks, the social network Reddit and a transportation network of international flight routes. We compare our results with existing core decomposition and centrality algorithms. Using ground truth about node importance, we show that AlphaCore yields the best precision and recall results among core decomposition methods using the same input features. An implementation is available at https://github.com/friedhelmvictor/alphacore.

CCS CONCEPTS

• **Mathematics of computing** → **Graph theory**; • **Information systems** → *Web searching and information discovery*.

KEYWORDS

Core decomposition; networks; data depth

ACM Reference Format:
Friedhelm Victor, Cuneyt G. Akcora, Yulia R. Gel, and Murat Kantarcioglu. 2021. AlphaCore: Data Depth based Core Decomposition. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467322

KDD '21, August 14–18, 2021, Virtual Event, Singapore.

ACM ISBN 978-1-4503-8332-5/21/08.
https://doi.org/10.1145/3447548.3467322

1 INTRODUCTION

Core decomposition is the primary machinery in network sciences to evaluate the importance of nodes and community (sub)structures in a wide range of application domains such as biology [19], social networks [1] and visualization [46]. One of the best-known representatives of core decomposition algorithms, k-core, finds the maximal subgraph where each node has at least k neighbors. Although the k-core algorithm [33] demonstrates high utility for analysis of graph structural properties, k-core does not account for important graph information such as the direction of edges, edge weights, and node features. To address these limitations, multiple modifications of k-core have been proposed to tackle task-specific graphs [21]. Among them, the weighted k-core [8] is one of the few algorithms that combine two node properties (degree and incident edge weights) into a single one: the weighted degree.

Depending on a specific task and application domain, however, we may need to incorporate other types of graph properties and allow for more than two node attributes to be combined. For example, in a financial transaction network where each node represents a different account, we may be interested to evaluate the total transaction amount sent to and from an account. In addition, the time-based activity may have significant implications with respect to the account's importance in the network. Such modern applications require the creation of new core decomposition algorithms that can systematically and reliably evaluate arbitrary node and edge features on weighted, directed graphs.

To address these challenges, we propose AlphaCore, a new core decomposition algorithm that combines multiple node properties using the statistical methodology of data depth [24]. The key idea of data depth is to offer a center-outward ordering of all observations by assigning a numeric score in $[0, 1]$ range to each data point with respect to its position within a cloud of a multivariate probability distribution. Using such a data depth function designed for directed and weighted graphs, we can easily map a node with different features to a single numeric score, while preserving its relative importance with respect to other nodes. Our AlphaCore algorithm iteratively removes the less important nodes and re-computes the depth of the remaining nodes. As a result, even if nodes have many features, we only need to iterate over depth values during decomposition steps. This new approach allows us to easily use AlphaCore in practice in an unsupervised fashion, without the need to specify numerous thresholds.

To illustrate the effectiveness of our AlphaCore algorithm in identifying important nodes and communities in directed weighted graphs, we evaluate its performance on three types of networks. Our primary focus is on financial networks, as these have yet seen little application of core decomposition methods, and at the same time are usually directed and weighted. To study such networks, we have collected asset transfers of 28 blockchain-based token networks. To illustrate AlphaCore's performance beyond financial networks, we examine a Reddit social network consisting of cross-links between subreddits, and lastly an international flight route network. For each network, we have collected ground-truth labels of importance, consisting of well-known cryptocurrency exchange accounts in token networks, the most popular subreddits by subscriber count, and the busiest airports by passenger count.

We compare the AlphaCore results with existing core decomposition algorithms, the underlying data depth mechanism itself, and centrality measures that use the same or similar input features. We empirically show that AlphaCore can significantly improve the identification of such central nodes compared to the state-of-art core decomposition techniques, performs better than centrality measures in our financial networks, and is on-par with them in the Reddit social network. We can summarize the key novelties as follows:

- *AlphaCore Algorithm*: We propose a new data depth-based core decomposition that can handle multidimensional node properties which can be obtained from a variety of node and edge property functions.
- *Mixture of Centrality and Core Decomposition*: We provide the option that AlphaCore can be used to perform core decomposition only among low data depth nodes, offering a mixture of centrality and core decomposition.
- *Extensive Empirical Evaluation*: By conducting node ranking experiments in financial, social, and transportation networks, we show that AlphaCore outperforms existing core decomposition algorithms that are based on the same input features.

2 PRELIMINARIES AND RELATED WORK

In this section, we introduce preliminaries on Blockchain, token networks, and recent Blockchain research, as these financial networks are the majority of networks studied in this paper. Afterward, we introduce core decomposition on networks and briefly cover network centrality measures.

2.1 Blockchain and Ethereum Token Networks

A blockchain is an immutable public ledger that records transactions in discrete data structures called blocks. The earliest blockchains are cryptocurrencies such as Bitcoin and Litecoin where a transaction is a transfer of coins.

Recently, other blockchains such as Ethereum have emerged. The Ethereum project [44] was created in July 2015 to provide Smart Contract functionality on a blockchain. Smart Contracts are Turing complete software codes that are replicated across a blockchain network. Smart Contracts ensure deterministic code execution that can be verified publicly. Some Smart Contracts implement mechanisms that allow trading digital assets, known as tokens [41], on the blockchain. We refer to such a Smart Contract as a *token contract*, and use the term token interchangeably. Similar to cryptocurrencies, a token is transferred publicly between accounts (addresses), and may have an associated value in fiat currency which is arbitrated by token demand and supply in the real world. A token network is a directed, weighted multigraph where an edge denotes the transferred token value. Each token network has its own edge weight scale (in $[1, \infty)$), typically ranging between 1 to 10^{13}.

Blockchain technology has created a new class of online financial institutions that are known as cryptocurrency exchanges. They can employ regular accounts or Smart Contracts to facilitate asset trades among blockchain addresses. Although blockchain users do not have to disclose their addresses, most exchanges publish or confirm their addresses to foster trust and security in their communities.

2.2 Blockchain Graph Analysis

Early efforts in Blockchain graph analysis de-anonymized and linked Bitcoin addresses by using heuristics on transaction behavior [22]. As the number of e-crime transactions increased on cryptocurrencies, algorithmic models have been developed to link bitcoin addresses by using node features on the network [11]. Ethereum transaction networks have been studied for their empirical properties [2], and [37, 38] explored token networks in terms of the degree distribution, power laws, and clustering. Recent measurement works [17, 41] study all Ethereum ERC20 tokens.

2.3 Core Decomposition Algorithms

The k-core decomposition of an undirected, unweighted graph $\mathcal{G}$ finds maximally connected subgraph(s) in which we connect every node to at least $k \in \mathbb{Z}^+$ nodes [33]. Sometimes, a connected graph can decompose into multiple disconnected k-core subgraphs. The k–shell of a graph $\mathcal{G}$ is the set of all nodes belonging to the k-core of $\mathcal{G}$ but not to the $(k+1)$-core [23].

A naïve implementation of k-core iteratively deletes nodes whose degree falls below a k, until it deletes all nodes from the graph. The implementation has a computational complexity of $O(m \log n)$, where m and n are the number of edges and nodes in the network, respectively. Batagelj and Zaversnik reduce the complexity to $O(m+n)$ "by keeping an in-memory array of all possible degree values and keeping track of bin boundaries" [4].

The k-core decomposition is a fundamental operation in graph similarity matching [28], graph clustering [9], network visualization [10], anomaly detection [35], robustness analysis [5] and many other applications [18, 21].

In many networks, we annotate nodes, or edges with weights that represent property values. Extending core decomposition to weighted graphs requires a robust method to combine multiple properties, which is not straightforward. Solutions weight-average [10] or take the geometric mean of node properties in an undirected network. For example, the weighted degree of a node k_i' has been defined as follows [8]:

$$k_i' = \left[k_i^a \left(\sum_j^{k_i} w_{ij} \right)^\beta \right]^{\frac{1}{\alpha+\beta}} \quad (1)$$

Here, k_i denotes a node's degree, w_{ij} its incident weights and α and β are weighting factors. However, existing approaches ignore data skew and network sparsity and do not generalize well beyond two features. *In contrast, our method can be applied to networks and features of any type without user supervision in feature scaling/normalization.*

One previously proposed generalization of core decomposition relies on defining node property functions that are based on each node's edges and extended neighborhood [3]. According to this generalization, a node property function p should be *monotone*, meaning for subgraphs $C_1, C_2 \subseteq G(V, E)$ it should hold that:

$$C_1 \subseteq C_2 \Rightarrow p(v, C_1) \leq p(v, C_2), \forall v \in V \tag{2}$$

An example of a node property function that is not monotonous would be the number of shortest paths that lead through a node u. The removal of other nodes in the network can lead to the increase or decrease of this number for node u. Adhering to the restriction however allows for a wide range of possible node properties, which can be based on a node's incident edges, its extended neighborhood, and various edge properties. See Table 1 for a list of example node property functions.

2.4 Network Centrality Measures

In contrast to k-core decomposition, where nodes with a high k are typically within a community of highly connected nodes, some centrality measures can assign high values to nodes that are not within such a community. In- or Out-Degree centrality for instance simply counts the number of incoming or outgoing edges of a node. A node with a high degree centrality does not need to be connected to other high degree centrality nodes. In the past, a multitude of centrality measures has been proposed [6], such as PageRank [31], betweenness centrality and closeness centrality, which utilizes the number of shortest paths that lead through a node, as well as variants that incorporate the weight of the edges [30].

3 DATA DEPTH

Depth functions have been initially introduced in the setting of non-parametric multivariate analysis to define affine invariant versions of median, quantiles, and ranks in higher dimensional spaces where there is no natural order (see historical overviews by Mosler [24], Nieto-Reyes and Battey [27]). The key idea of the depth approach is to offer a center-outward ordering of all observations by assigning a numeric score in $[0, 1]$ range to each data point with respect to its position within a cloud of multivariate or functional observations or a probability distribution. Nowadays, data depth is a rapidly developing field that gains increasing momentum due to the wide applicability of depth concepts to classification, visualization, high dimensional and functional data analysis [13, 25, 26, 34, 45]. Most recently, depth approaches have found novel applications in density-based clustering and space-time data mining [12, 14, 42], shape recognition and uncertainty quantification in computer graphics [36, 43], ordinal data analysis [15] and computational geometry for privacy-preserving data analysis [20]. Nevertheless, data depth is yet a largely unexplored concept in network sciences [7, 32, 39, 40].

Table 1: Example node property functions. Most functions are adopted from the related work. Functions that encode the community around a node, such as cycles, can help bringing a higher ordered structure of the network into use.

Function	Value / number of ...
$N(u)$	neighbors of u
$N_{out}(u)$	neighbors reachable with outgoing edges from u
$N_{in}(u)$	neighbors reachable with incoming edges to u
$deg(u)$	edges to/from u (Degree)
$deg_{out}(u)$	outgoing edges from u (Out-Degree)
$deg_{in}(u)$	incoming edges to u (In-Degree)
$\bigcirc(u, l)$	undirected cycles of length l that u is part of
$\circlearrowright\ (u, l)$	directed cycles of length l that u is part of
$t(u, l)$	length l timeframes that u has edges in
$S(u)$	sum of edge weights incident to a node (Strength)
$S_{out}(u)$	sum of outgoing edge weights (Out-Strength)
$S_{in}(u)$	sum of incoming edge weights (In-Strength)

Definition 1 (Data Depth). *Formally, let E be a Banach space (e.g., $E = \mathbb{R}^d$), $\mathcal{B}$ its Borel sets in E, and $\mathcal{P}$ be a set of probability distributions on $\mathcal{B}$. We view $\mathcal{P}$ as the class of empirical distributions giving equal probabilities $1/n$ to n data points in E. Then a data depth function is a function $\mathbb{D} : E \times \mathcal{P} \longrightarrow [0, 1]$, $(x, P) \longrightarrow \mathbb{D}(x|P)$, $x \in E, P \in \mathcal{P}$ that shall satisfy the following desirable properties: affine invariant, upper semi-continuous in x, quasiconcave in x (i.e., having convex upper level sets) and vanishing as $||x|| \to \infty$. Specifically, a data depth function $\mathbb{D}(x)$ measures how closely an observed point $x \in \mathbb{R}^d$, $d \geq 1$, is located to the center of a finite set $\mathcal{X} \in \mathbb{R}^d$, or relative to F, which is a probability distribution in $\mathbb{R}^d$. In complex network analysis, these points may correspond to the features of nodes or edges.*

Among many depth functions formulated to date, the Mahalanobis depth is one of the most prominent in the current practice.

Definition 2 (Mahalanobis (MhD) depth). *Let $x \in \mathbb{R}^d$ be an observed data point, then Mahalanobis (MhD) depth of x in respect to a d-variate probability distribution F with mean vector $\mu_F \in \mathbb{R}^d$ and covariance matrix $\Sigma_F \in \mathbb{R}^{d \times d}$ is given by*

$$MhD_{\mu_F}(x) = \left(1 + (x - \mu_F)^\top \Sigma_F^{-1} (x - \mu_F)\right)^{-1}. \tag{3}$$

Here $^\top$ denotes matrix transpose. The MhD depth measures the outlyingness of the point with respect to the deepest point of the distribution (here μ_F), and allows to easily handle the elliptical family of distributions, including a Gaussian case.

MhD offers flexibility in changing the reference point with respect to which we compute data rankings. For instance, instead of μ_F we can select an arbitrary point $x_0 \in \mathbb{R}^d$ and compute MhD in respect to this new reference point x_0

$$MhD_{x_0}(x) = \left(1 + (x - x_0)^\top \Sigma_F^{-1} (x - x_0)\right)^{-1}. \tag{4}$$

Furthermore, Σ_F can be substituted by any empirical estimator of covariance matrix $\hat{\Sigma}$ obtained from the observed data sample $x_1, x_2, \ldots, x_n$.

4 METHODOLOGY

AlphaCore relies on a single computed property, namely the data depth of multiple node properties, to determine core membership. During AlphaCore execution, these data depth values are iteratively updated by computing node property functions and applying data depth on the resulting values.

Let $G(V, E, w)$ be a *directed, weighted* multigraph where V is the set of nodes and E is a multiset of edges. Each edge has a weight where $w : E \rightarrow \mathbb{R}^+$ denotes a weight function assigning weights to each edge $e \in E$. In analogy to the generalized cores definitions [3], a node property function $p(v, C)$ with $p \in V$ and $C \subseteq E$ can assign a real value to each $v \in V$, which can be based on edge properties such as a weight. Instead of using the degree of a node as the core-criterion, we compute the Mahalanobis data depth to the origin $\mathbf{0}$. This collapses multiple node properties into one node property, the node depth, which is in the range of $[0, 1]$:

Definition 3 (Mahalanobis depth to the origin (MhDO)).

$$MhDO_F(x) = \left(1 + x^\top \Sigma_F^{-1} x\right)^{-1}, \tag{5}$$

where Σ_F is the covariance matrix of F. The MhDO depth measures the outlyingness of the point x (in our case the node property column vector) with respect to the origin $\mathbf{0}$.

The rationale behind computing depth with respect to the origin is the following: By construction, MhD is a symmetric function and does not distinguish among outlying actors with, for example, too high and too low trading volumes, relative to all other actors. Since our primary focus is on the most active actors, we rank our data with respect to the origin which allows us to focus on the right tail of the distribution, while still accounting for the important geometric structure of the whole data cloud.

We establish the core value α of a node by using data depth ϵ of its properties. Nodes with high property values (e.g., large edge weight) tend to have a low depth. Conversely, nodes with low property values tend to have a high depth (e.g., most blockchain nodes trade small token amounts). In both cases, node property values are not the only depth determinant, the community structure around the node plays a role as well. Nodes are in the same core if their depth is at most ϵ. When we define $\alpha = 1 - \epsilon$, then:

Definition 4 (AlphaCore). *The α-core of a graph is a maximal subgraph in which each node has at most a data depth of $1 - \alpha$ based on a set of node properties.*

To determine core membership, we iteratively remove nodes by moving a threshold ϵ from higher depths to lower depths. The idea is similar to iteratively removing the lowest degree nodes in k-core decomposition. However, compared to $k \in \mathbb{Z}^+$ in k-core, AlphaCore must iterate on depth values in $[0, 1]$. Depending on the type of network and the expected distribution of node property values, different step sizes and step size functions for ϵ can be used. Naïvely, a linear step size such as 0.1 may be chosen, which may result in up to 10 different cores. However, for heavily skewed distributions of node property values, this will result in a few very large shells, where a shell is the set of nodes that belong to a certain core, but not the next higher core. In contrast, a step size function that attempts to find similarly sized shells could pick the next ϵ based on a given number of nodes that should at least be contained. In a ranking task where the goal is to find the most important nodes in a graph, these shell sizes could follow an exponential decay, given an appropriate step-size function. Multiple iterations may occur at the same ϵ level until no further nodes can be removed at the same ϵ. To capture these distinct iterations, we link a continuously incremented batch id to the nodes removed in an iteration. That way, even though some nodes have the same α level, they can have different batch ids that imply a ranking. In other words, batch ids offer a more detailed look into the cores.

Finally, AlphaCore can be used with a custom ϵ_{start} value, which assigns all nodes with depth $> \epsilon$ to the 0-core in the first iteration. This delays the incorporation of community structure. If only one node feature is used, e.g. the number of incoming edges, this is equivalent to the initial execution of in-degree centrality, and core decomposition is then only executed between the nodes with higher centrality (low data depth). Using AlphaCore in this way can be regarded as a mixture of core decomposition and centrality measure.

Algorithm 1: AlphaCore Decomposition

Input: Directed, weighted, multigraph $G(V, E, w)$,
Set of node property functions $p_1, ..., p_n \in P$,
Starting epsilon ϵ_{start},
Step size function $\triangle_s$,
Step size s
Output: core value and batchID for each $v \in V$

```
// Compute feature matrix
F = [f_1, ..., f_n] = ∀p_i ∈ P : f_i = p_i(v, G), ∀v ∈ V;
Σ_F^{-1} = cov(F)^{-1};                    // compute only once
// Compute initial depth values
z = [z_1, ..., z_n] = ∀v_i ∈ V : z_i = [1 + (F_{i,*})' Σ_F^{-1} (F_{i,*})]^{-1};
ϵ = ϵ_start;
C_* ← 0;                                   // cores initialized to 0
B_* ← 0;                                   // batch ids initialized to 0
α ← 1 − ϵ;
α_prev ← α;
batchID ← 0;
while |V| > 0) do
    do
        foreach z_i ≥ ϵ do
            // Mark with α and batch id
            C_i ← α_prev;                  // set node core
            B_i = batchID;
            V = V \ {v_i};
        batchID = batchID + 1;             // iterate
        // recompute node properties
        F = ∀p_i ∈ P : p_i(v, G), ∀v ∈ V;
        // recompute depth
        z_i = [1 + (F_{i,*})' Σ_F^{-1} (F_{i,*})]^{-1}, ∀v_i ∈ V;
    while ∃z_i : (z_i ≥ ϵ) ∧ (v_i ∈ V);
    α_prev = α;
    ϵ = △_s(z, s);                         // decrease ϵ
    α = 1 − ϵ;
return C, B
```

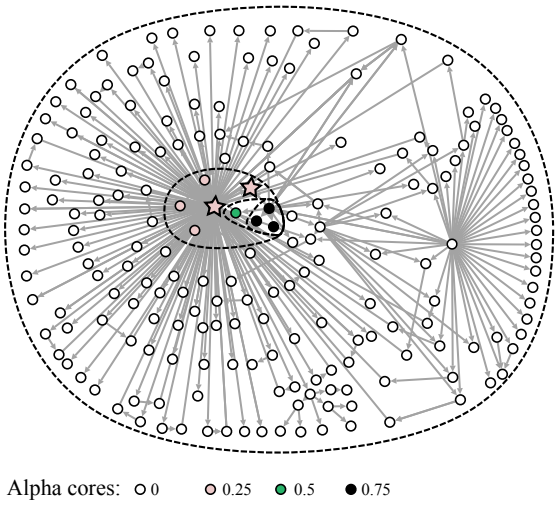

Figure 1: AlphaCore with a linear step size of $s = 0.25$ applied to the SoftChainCoin network. $\epsilon_{start} = 1$ and node features are N_{in}, S_{in}, S_{out}. This ensures that there are at most $1/s$ distinct cores, in this case 4. Here, the exchange nodes (stars) are found in the third highest core. The 0-core contains the majority of nodes.

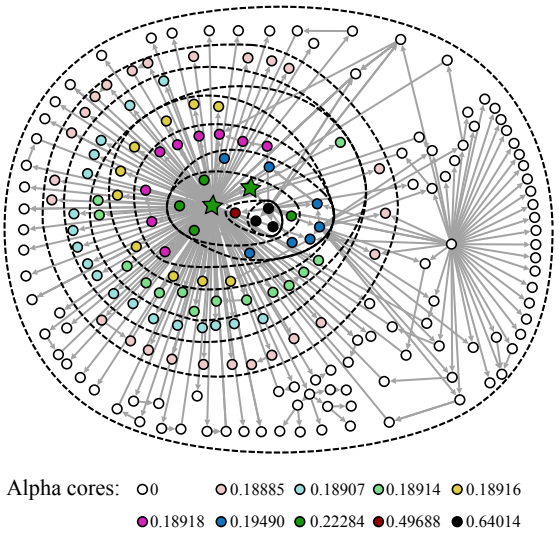

Figure 2: AlphaCore with exponentially decaying step size function ($s = 0.25$) applied to the SoftChainCoin network. $\epsilon_{start} = 1$ and node features are N_{in}, S_{in}, S_{out}. Each subsequent target core size is at least 25% of the remaining nodes. This approach always guarantees small sized cores at high α, suitable for determining a node importance ranking. The exchange nodes (stars) are found in the third highest core.

4.1 Algorithm Description

The AlphaCore decomposition is illustrated in Algorithm 1. In line 1, a feature matrix is computed based on each node property function that is to be used. For example, this could include a node's neighborhood size or the number of triangles it is part of. See Table 1 for more examples. Based on the feature matrix F, the inverse covariance matrix Σ_F is computed in line 2 and will be used for future data depth calculations. In line 3, the initial depth of each node is computed using the Mahalanobis depth with respect to the origin. Lines 4— 9 are used to initialize ϵ, α, α_{prev}, the starting batch id and the core values and batch ids for each node that are to be returned. The main algorithm starts in line 10, which iterates as long as there are nodes left in the graph. Each node that has a depth higher than or equal to the current ϵ, is assigned the previous α value at which it was still contained in the core. A batch id is also assigned and the nodes are then removed from the graph (lines 13— 15). As one batch of node removals has been completed, the batch id is incremented in line 16, and the feature matrix and depth values are recomputed in lines 17— 18. If any remaining nodes still have a depth greater or equal to the current depth, the next batch is started at the same ϵ level. Otherwise, ϵ and α are updated based on the depth level that has just been completed, in combination with a step size function $\triangle_s$ in lines 21— 22. If all nodes have been removed from the graph, the algorithm is complete. Vectors C and $\mathcal{B}$ are returned, which contain core values and batch ids for each node.

Computational complexity. AlphaCore requires inverting the covariance matrix at line 2 once, which can be computed by Cholesky decomposition in $O(n^3)$ for n features. Node property recalculations can have $O(v^2)$ complexity for properties based on the second-degree neighborhood (e.g., cycles of length 4) of v nodes. Each iteration needs recomputing node features, occurring m (number of edges) times at most. The worst-case complexity is $O(n^3) + O(m \times v^2 \times n)$. However, network sparsity reduces the v in the neighborhood, and n is a small value. Furthermore, the number of iterations is much smaller than m because multiple nodes are removed in batches.

4.2 Step Size Functions

Depending on the task for which core decomposition is used, we recommend using one of two different step size functions. Their application is also illustrated in Figure 1 and 2.

- *Linear function:* When one wants to ensure that nodes in different cores are different with respect to their node properties and community embedding, a linear step size function can be chosen. This function simply decrements the ϵ values with a fixed step size s such as $s = 0.1$. As a result, at most $1/s$ cores will be identified.
- *Depth-dependend exponential decay function:* If the task is to identify a node ranking, a step size function that ensures that the highest cores contain few nodes is appropriate. To achieve this, each subsequent core size should be smaller than the previous one. An exponential decay can be implemented by choosing the next ϵ based on the highest depth observed at a certain quantile, for example, the median, corresponding to a step size of 0.5. This method is shown in Algorithm 2, and requires the depth values of the remaining nodes, and a step size s as inputs to obtain the next ϵ.

Algorithm 2: Exponential decay step function $\triangle_s$

Input: Depth values of remaining nodes z,
Step size s, $(s \in (0, 1])$
Output: ϵ

```
count = |z|;                    // Count remaining nodes
z = sort(z, descending);
ϵ = z[⌈count · s⌉];             // Get highest depth
return ϵ
```

Figures 1 and 2 illustrate these step size functions. Both figures are based on a very small token network, the *SoftChainCoin*, consisting of 278 edges and 197 nodes. Two nodes are exchange accounts belonging to the HotBit exchange, marked with stars in both figures. We run AlphaCore on both networks, with the node property functions N_{in}, S_{in} and S_{out}. Figure 1 shows a linear step size of $s = 0.25$, Figure 2 an exponential decay with $s = 0.25$. Whereas in Figure 1, the core values are well separated, Figure 2 has multiple cores around the value 0.19 (including the exchange nodes), indicating that these cores may be rather similar. Yet, due to the decaying step size function in Figure 2, it is ensured that the highest cores only contain very few nodes. While in this example, this is also true for the linear step size function, it does not hold in general. Particularly in large graphs, it can happen that the highest core still contains a lot of nodes because they are quite similar. A solution would be to pick a very small linear step size function, but this, in turn, incurs much longer runtimes, as many more iterations need to be performed. Therefore, for the remaining experiments, we use our exponentially decaying step-size function.

Uniqueness and stability. Given a graph and a step size, AlphaCore results are deterministic. Furthermore, the map $\mathbb{D} : F \longrightarrow [0, 1]$, where $F \in \mathbb{R}^d$ is a set of node features is surjective. As discussed by Mosler [24], Mahalanobis depth is unique within any affine family of nondegenerate d-variate distributions with finite second moments, in the sense that a single contour set of the Mahalanobis depth is sufficient to identify the distribution from this family of distributions.

Figures 1 and 2 illustrate that AlphaCore is stable in terms of core membership at a certain α. For example, both step size variants identify the same nodes at an α of 0.5 and 0.49688 respectively.

5 EXPERIMENTAL RESULTS

We evaluate the performance of AlphaCore on token networks of the Ethereum blockchain, a network based on the Reddit social network, and an international airport routes network. Our goal is to evaluate the AlphaCore's performance on different types of networks to gain insights into the AlphaCore's applicability. Although the structure of each network is different, all networks are weighted and directed and have ground truth labels assigned to each node based on node importance. These networks are thus suitable for an evaluation across several node properties (see Table 1). We compare our results to k-core and weighted k-core algorithms, the underlying data depth method, and popular centrality measures, allowing us to illustrate the utility of core decomposition algorithms.

5.1 Datasets

Token networks. Using the tool ethereum-etl,[1] we have extracted token transfers between Oct-16-2018 and May-04-2020. As there are thousands of token networks contained in these transfers, we first consider the top 100 networks by the number of transfers, excluding the token network *USD Tether*, as it is too large for efficient algorithm comparison ($> 4M$ nodes). We downloaded node labels in May 2020 from Etherscan,[2] a prominent Ethereum block explorer, that curates and maintains address labels. In total, 296 addresses from 149 centralized and decentralized exchange addresses are listed publicly, which are likely used frequently. Not all the top token networks have multiple exchange nodes taking part in them. For the following evaluation, we only consider those token networks that contain at least 10 such labeled exchange nodes, which reduces the number of networks considered to 28.

Reddit crosslinks. This network originates from a study on community interaction on the Reddit social network [16]. Each node represents a subreddit, and edges are comments that contain a (cross)link from one subreddit to another, making it a directed network. We have transformed this network into a weighted network by counting the number of the same source/destination crosslinks. To be able to use labels, we have retrieved a list of the top 100 subreddits by subscriber count at the time of the original dataset.[3]

Airport flight routes. This dataset has been curated from Openflights.org and consists of routes between international airports, where the edge weight indicates the number of routes. It has been used to study (weighted) centrality measures [29]. In addition, we have obtained labels of the top 30 busiest airports by passenger counts, at the time of the original dataset[4].

For reproducibility, we are making the code[5] and data[6] alongside this paper available. The data consists of the 28 token networks and labels. The Reddit and airport routes dataset and matching labels are already publicly available, and referenced via links.

5.2 Preprocessing

The token networks dataset needed further preprocessing. Most token networks are financial in nature, and can have very large differences in transferred amounts. This also holds true for transferred token amounts on the Ethereum platform, which can theoretically range between 0 and $2^{256} - 1$. In practice, amounts often range between 1 and 10^{13}. However, this is not guaranteed and implies there can be orders of magnitudes differences in the amounts, which in turn can lead to very long run times for algorithms such as weighted k-core. To avoid such issues, we remove edges with very small weights, where $weight < \frac{max(weight)}{10^{10}}$.

Note that the weighted k-core algorithm applies its own weight scaling: Normalization by the mean weight, and then division by the minimum weight.

[1] https://github.com/blockchain-etl/ethereum-etl
[2] https://etherscan.io/labelcloud
[3] https://web.archive.org/web/20170407193210/http://redditlist.com/
[4] https://web.archive.org/web/20110430004800/http://www.aci.aero/cda/aci_common/display/main/aci_content07_c.jsp?zn=aci&cp=1-5-212-1376-1379_666_2__
[5] https://github.com/friedhelmvictor/alphacore
[6] https://zenodo.org/record/4898412

Table 2: Performance comparison with a ranking task using 28 token networks, the Reddit crosslinks network, and the flight routes network. Performance is indicated using (average) precision and recall at k. We compare core decomposition methods, the Mahalanobis depth, and popular centrality measures with a variety of input features. Our work (alphaCore) is always used with a starting epsilon of 0.1 and a step size of 0.1. It performs best in comparison with other core decomposition methods and exceeds centrality measures at lower k. The best results are highlighted with a yellow background.

				28 Token Networks						Reddit Crosslinks Network						Flight Routes Network					
		Algorithm	Input features	AP@10	AP@20	AP@50	AR@10	AR@20	AR@50	P@10	P@20	P@50	R@10	R@20	R@50	P@10	P@20	P@50	R@10	R@20	R@50
Core Decomposition	1	**alphaCore**	N_{in}, N_{out}	0.54	0.45	0.30	0.10	0.16	0.25	0.30	0.35	0.36	0.03	0.07	0.18	0.70	0.75	0.38	0.23	0.50	0.63
	2	**alphaCore**	N_{out}	0.50	0.44	0.27	0.10	0.16	0.23	0.30	0.30	0.14	0.03	0.06	0.07	1.00	0.55	0.32	0.33	0.37	0.53
	3	**alphaCore**	N_{out}, S_{in}, S_{out}	0.42	0.33	0.21	0.07	0.11	0.18	0.50	0.45	0.34	0.05	0.09	0.17	0.20	0.35	0.40	0.07	0.23	0.67
	4	**alphaCore**	$N_{in}, N_{out}, S_{in}, S_{out}$	0.41	0.33	0.22	0.07	0.11	0.19	0.60	0.55	0.48	0.06	0.11	0.24	0.40	0.40	0.26	0.13	0.27	0.43
	5	**alphaCore**	N_{in}	0.50	0.41	0.24	0.10	0.15	0.21	1.00	0.95	0.58	0.10	0.19	0.29	0.90	0.50	0.30	0.30	0.33	0.50
	6	**alphaCore**	N_{in}, S_{in}	0.39	0.30	0.19	0.06	0.10	0.17	0.30	0.30	0.50	0.03	0.06	0.25	0.10	0.35	0.52	0.03	0.23	0.87
	7	k-core	N	0.36	0.26	0.14	0.06	0.08	0.11	0.10	0.05	0.08	0.01	0.01	0.04	0.40	0.50	0.36	0.13	0.33	0.60
	8	k-core	N_{out}	0.22	0.16	0.11	0.03	0.05	0.10	0.10	0.05	0.08	0.01	0.01	0.04	0.40	0.45	0.34	0.13	0.30	0.57
	9	k-core	N_{in}	0.13	0.09	0.06	0.03	0.04	0.06	0.10	0.05	0.14	0.01	0.01	0.07	0.40	0.45	0.34	0.13	0.30	0.57
	10	w. k-core	N, S	0.37	0.30	0.19	0.06	0.10	0.16	0.30	0.45	0.48	0.03	0.09	0.24	0.40	0.40	0.46	0.13	0.27	0.77
	11	w. k-core	N_{in}, S_{in}	0.33	0.23	0.15	0.06	0.07	0.13	0.10	0.05	0.22	0.01	0.01	0.11	0.40	0.40	0.42	0.13	0.27	0.70
	12	w. k-core	N_{out}, S_{out}	0.28	0.29	0.21	0.05	0.10	0.18	0.10	0.05	0.10	0.01	0.01	0.05	0.40	0.40	0.44	0.13	0.27	0.73
	13	w. k-core	N_{in}, N_{out}	0.04	0.03	0.02	0.01	0.01	0.02	0.10	0.05	0.10	0.01	0.01	0.05	0.20	0.20	0.36	0.07	0.13	0.60
Data Depth	14	Mahalanobis depth	N_{out}	0.45	0.35	0.27	0.08	0.13	0.23	0.10	0.20	0.18	0.01	0.04	0.09	0.80	0.80	0.54	0.27	0.53	0.90
	15	Mahalanobis depth	N_{in}, N_{out}	0.44	0.35	0.25	0.08	0.13	0.23	0.60	0.50	0.42	0.06	0.10	0.21	0.70	0.75	0.46	0.23	0.50	0.77
	16	Mahalanobis depth	N_{out}, S_{in}, S_{out}	0.41	0.34	0.23	0.07	0.12	0.20	0.50	0.50	0.36	0.05	0.10	0.18	0.50	0.60	0.46	0.17	0.40	0.77
	17	Mahalanobis depth	N_{in}, S_{in}	0.38	0.35	0.23	0.07	0.12	0.20	0.90	0.80	0.68	0.09	0.16	0.34	0.60	0.65	0.52	0.20	0.43	0.87
	18	Mahalanobis depth	$N_{in}, N_{out}, S_{in}, S_{out}$	0.38	0.37	0.24	0.06	0.13	0.21	0.60	0.55	0.42	0.06	0.11	0.21	0.40	0.45	0.44	0.13	0.30	0.73
	19	Mahalanobis depth	N_{in}	0.36	0.30	0.22	0.07	0.12	0.21	0.90	0.90	0.76	0.09	0.18	0.38	0.80	0.80	0.54	0.27	0.53	0.90
Centrality Measures	20	InDeg. Centr.	deg_{in}	0.28	0.24	0.17	0.06	0.10	0.16	0.90	0.90	0.76	0.09	0.18	0.38	0.80	0.80	0.54	0.27	0.53	0.90
	21	w. InDeg. Centr.	deg_{in}, S_{in}	0.42	0.38	0.26	0.08	0.14	0.23	1.00	0.90	0.76	0.10	0.18	0.38	0.90	0.85	0.56	0.30	0.57	0.93
	22	OutDeg. Centr.	deg_{out}	0.34	0.30	0.20	0.06	0.11	0.18	0.10	0.20	0.18	0.01	0.04	0.09	0.80	0.80	0.54	0.27	0.53	0.90
	23	w. OutDeg. Centr.	deg_{out}, S_{out}	0.50	0.39	0.28	0.09	0.15	0.25	0.20	0.20	0.18	0.02	0.04	0.09	0.90	0.85	0.56	0.30	0.57	0.93
	24	Pagerank	deg_{in}, S_{in} (neighbors)	0.47	0.40	0.24	0.08	0.14	0.22	0.90	0.80	0.68	0.09	0.16	0.34	0.70	0.65	0.48	0.23	0.43	0.80
	25	Betw. Centr.	shortest paths	Computationally too expensive for larger graphs						0.60	0.40	0.40	0.06	0.08	0.20	0.80	0.60	0.46	0.27	0.40	0.77
	26	w. Betw. Centr.	w. shortest paths							0.60	0.60	0.48	0.06	0.12	0.24	0.80	0.70	0.44	0.27	0.47	0.73
	27	Clo. Centr.	shortest paths							0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.85	0.56	0.33	0.57	0.93
	28	w. Clo. Centr.	w. shortest paths							0.10	0.05	0.12	0.01	0.01	0.07	0.90	0.90	0.56	0.30	0.60	0.93

5.3 Evaluation

We evaluate the task of finding top n labeled nodes in Table 2, using precision and recall measures (shown as $P@n, R@n$ respectively), where $AP@n$ and $AR@n$ denote the averages across multiple networks. Nodes are ranked by their core membership, depth value, or centrality value in descending order. AlphaCore is used with a starting epsilon and an exponential decay step size of 0.1. If there are different batch ids at the same alpha level, we additionally rank nodes by their batch id. Weighted k-core and weighted centralities are used with default weighting parameters, which are all 0.5.

Both k-core and weighted k-core are defined for undirected networks. As the token networks are directed multigraphs, we use a node's neighborhood size instead of their degree. For the other networks, degree equals neighborhood size. Furthermore, we compare against modified variations of k-core (rows 8 and 9) and weighted k-core (rows 11 – 13), using different input features. The first insight is that AlphaCore performs best in comparison to the other two core decomposition algorithms. While overall, weighted centralities seem to be a good choice, AlphaCore exceeds their performance for token networks and is on-par with them in the Reddit crosslinks network. The reason why AlphaCore performs better than k-core and weighted k-core, is likely due to usage of the starting epsilon of 0.1. As mentioned in Section 4, this is similar to first running a centrality measure, and afterwards performing core decomposition on high centrality nodes, yielding a mixture of centrality and core decomposition. We can see further evidence of this by looking at the data depth results. Using only the Mahalanobis depth on the input features leads to similar results like the centrality measures. Row 19 and 20 yield the exact same results for the Reddit and flight networks. This illustrates that the Mahalanobis depth can act like a centrality measure. The results are not the same for the token networks, because neighborhood size and degree are not the same when multi edges are present.

The Reddit dataset, at lower k, benefits from performing core decomposition as shown in row 5. This does not hold true for the flight routes network, which is likely due to the fact that the number of passengers at an airport is unrelated to the presence of a core community, as non-direct flight routes typically do not offer many different connecting airports. The takeaway is that the flight routes network may be unsuitable for core decomposition in general.

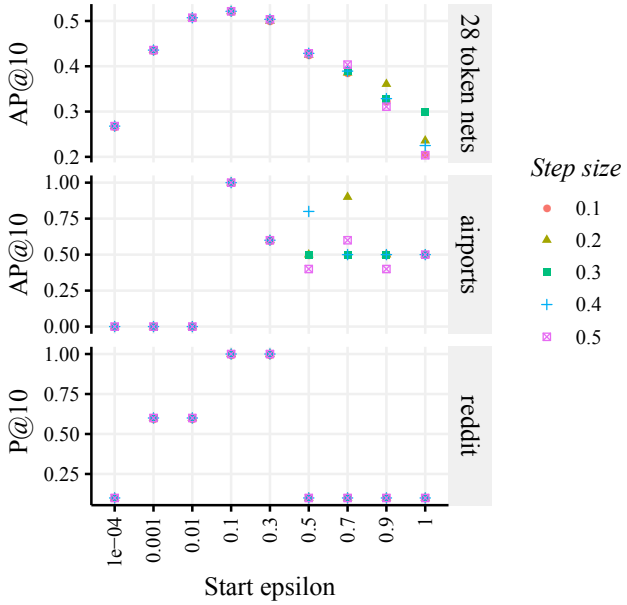

Figure 3: AlphaCore (average) precision at 10 for a range of starting epsilon and step sizes across all three network types. The input features for each dataset are the best performing taken from Table 2 (rows 1, 5 and 2), and the step size function is decaying exponentially. At a starting epsilon of 0.1, the best average precision at 10 is achieved, regardless of network type and step size.

Insights into parameter selection. To get a better understanding of AlphaCore's parameters, we explore different start epsilons and step sizes in Figure 3, where we run AlphaCore with their best performing input features for each dataset (rows 1, 5, and 2) on all networks. Across all datasets, a starting epsilon of 0.1 yields the best precision, which means it is optimal if a large number of high depth nodes are removed first, before iterating on the ϵ values and incorporating the community structure. An even smaller epsilon already has a negative effect, because too many nodes have already been removed. As the step size function follows an exponential decay, the actual step size itself does not have a large impact. As a rule of thumb, we recommend low starting epsilons, when wanting to focus only on the community of low depth nodes.

Figure 4 shows the precision for all 28 networks by vertex count. We see that AlphaCore reaches a precision of $\geq$ 0.8 for 9 token networks. Overall, AlphaCore improves precision over competitors when using a single feature as an input; also, it achieves the highest precision and recall values when using multiple input features.

5.4 Discussion

Data challenges. Traditional core decomposition algorithms propose little in alleviating data challenges. For example, weighted k-core proposes using $\alpha = \beta = 0.5$ for combining properties, but improving over this arbitrary choice is left as future work. In most networks, edge weights are skewed and long tails complicate scaling and normalization issues. For example, in token networks, max

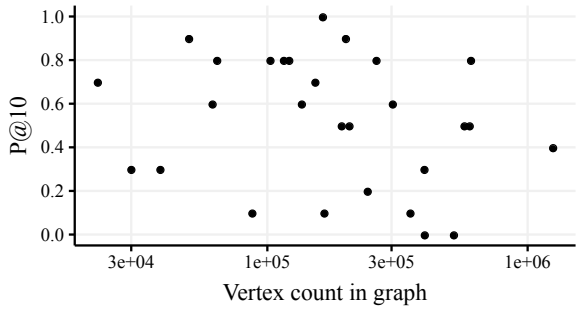

Figure 4: Graph order on the 28 token networks that have at least 10 nodes of interest. Precision at 10 for each of the 28 token networks with at least 10 nodes of interest, illustrated by the number of nodes in the network. The input features are N_{in} and N_{out}, and start epsilon and step size are 0.1. The precision does not appear to depend on the network size.

edge weights can reach 10^{13}, whereas node degrees are typically less than 3. With such a difference in scales, node property combination becomes a challenge in weighted k-core. AlphaCore is not beset with such issues as it benefits from the robustness of data depth in ordering data.

Limits on precision. We have seen that not all networks are suitable for core decomposition. This is true for the flight routes network. In addition, the labels that we have acquired may not directly correspond to high core nodes. In token networks for example, it is possible that high core nodes contain automated traders performing arbitrage and are thus by definition, influential/important nodes in the network. Our external label set does not contain such nodes because unlike exchanges, arbitrageurs do not disclose their identities. As a result, precision in detecting only exchanges may never reach 100%.

Applicability of AlphaCore. AlphaCore requires no coefficients to combine node properties, nor multiple thresholds to iterate during decomposition. As a result, it can be applied to any network with no supervision. For example, Figure 4 shows that AlphaCore precision for large networks is similar to that of small networks. Furthermore, AlphaCore achieves better performance using just one property. For example, Table 1 shows that AlphaCore precision is better than that of k-core for degree-based properties. We believe that this performance improvement is due to the usage of data depth, which is robust against data skew due to its ordering of data points in a center-outward fashion. Skewed node properties are naturally ordered in AlphaCore which obviates combining node properties.

Application domains of AlphaCore. AlphaCore allows unsupervised core decomposition on an arbitrary number of node properties defined on any network type. For example, in protein interaction networks, a pathway node property can be defined in terms of genes, which can encode certain post-translational modifications. By using AlphaCore, multiple such properties can be utilized and we can benefit from integrating domain expertise with network analysis.

6 CONCLUSION

In this work, we have provided a novel core decomposition algorithm named AlphaCore that is developed to handle directed, weighted graphs with multiple node features. We have evaluated AlphaCore's performance using blockchain-based token networks, a Reddit social network, and a flight routes network using ground truth labels that indicate the importance of different nodes in the network. We have shown that AlphaCore can identify important nodes more accurately compared to state of art core decomposition algorithms, and can, in some networks, outperform centrality measures as well, as it can act as a mixture of centrality measure and core decomposition. Unlike existing decomposition algorithms, AlphaCore does not require multiple weighting parameters to be specified in order to perform well on a given task. In the future, we plan to explore the usage of AlphaCore algorithm to determine network robustness.

7 ACKNOWLEDGEMENTS

This work is supported in part by National Science Foundation under Grant No. ECCS 2039701, ECCS 1824716, DMS 1925346, CNS 1837627, OAC 1828467, IIS 1939728, CNS 2029661 and Canadian NSERC Discovery Grant RGPIN-2020-05665.

REFERENCES

[1] Mohammed Ali Al-garadi, Kasturi Dewi Varathan, and Sri Devi Ravana. 2017. Identification of influential spreaders in online social networks using interaction weighted K-core decomposition method. *Physica A: Statistical Mechanics and its Applications* 468 (2017), 278–288.

[2] A. Anoaica and H. Levard. 2018. Quantitative Description of Internal Activity on the Ethereum Public Blockchain. In *NTMS*. IEEE, 1–5.

[3] Vladimir Batagelj and Matjaž Zaveršnik. 2002. Generalized cores. *CoRR* cs.DS/0202039 (2002).

[4] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[5] Kate Burleson-Lesser, Flaviano Morone, Maria S Tomassone, and Hernán A Makse. 2020. K-core robustness in ecological and financial networks. *Scientific reports* 10, 1 (2020), 1–14.

[6] Ernesto Estrada. 2012. *The structure of complex networks: theory and applications*. Oxford University Press.

[7] D. Fraiman, F. Fraiman, and R. Fraiman. 2015. Statistics of dynamic random networks: a depth function approach. *arXiv:1408.3584v3* (2015).

[8] Antonios Garas, Frank Schweitzer, and Shlomo Havlin. 2012. A k-shell decomposition method for weighted networks. *New Journal of Physics* 14, 8 (2012), 083030.

[9] C. Giatsidis, F. Malliaros, D. Thilikos, and M. Vazirgiannis. 2014. Corecluster: A degeneracy based graph clustering framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[10] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. 2011. Evaluating cooperation in communities with the k-core structure. In *2011 International conference on advances in social networks analysis and mining*. IEEE, 87–93.

[11] D. Y. Huang and D. McCoy. 2018. Tracking Ransomware End-to-end. In *Tracking Ransomware End-to-end*. IEEE, 1–12.

[12] X. Huang and Y.R. Gel. 2017. CRAD: Clustering with Robust Autocuts and Depth. In *ICDM*.

[13] R. J. Hyndman and H. L. Shang. 2010. Rainbow plots, bagplots, and boxplots for functional data. *Journal of Computational and Graphical Statistics* 19 (2010), 29–45.

[14] Myeong-Hun Jeong, Yaping Cai, Clair J Sullivan, and Shaowen Wang. 2016. Data depth based clustering analysis. In *SIGSPATIAL*.

[15] M. Kleindessner and U. von Luxburg. 2017. Lens Depth Function and *k*-Relative Neighborhood Graph: Versatile Tools for Ordinal Data Analysis. *Journal of Machine Learning Research* 18, 18 (2017), 1–52.

[16] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2018. Community interaction and conflict on the web. In *Proceedings of the 2018 world wide web conference*. 933–943.

[17] Xi Tong Lee, Arijit Khan, Sourav Sen Gupta, Yu Hann Ong, and Xuan Liu. 2020. Measurements, analyses, and insights on the entire ethereum blockchain network. In *Proceedings of The Web Conference 2020*. 155–166.

[18] R.-H. Li, G. Wang, W. Yang, and J. X. Yu. 2020. Ordering Heuristics for k-clique Listing. In *Proceedings of the VLDB Endowment*.

[19] Feng Luo, Bo Li, Xiu-Feng Wan, and Richard H Scheuermann. 2009. Core and periphery structures in protein interaction networks. In *BMC bioinformatics*, Vol. 10. Springer, S8.

[20] Hassan Mahdikhani, Rasoul Shahsavarifar, Rongxing Lu, and David Bremner. 2020. Achieve privacy-preserving simplicial depth query over collaborative cloud servers. *Peer-to-Peer Networking and Applications* 13, 1 (2020), 412–423.

[21] Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. 2020. The core decomposition of networks: Theory, algorithms and applications. *The VLDB Journal* 29, 1 (2020), 61–92.

[22] S. Meiklejohn, M. Pomarole, G. Jordan, D. Levchenko, K.and McCoy, G. M Voelker, and S. Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. In *IMC*. ACM, 127–140.

[23] Daniele Miorandi and Francesco De Pellegrini. 2010. K-shell decomposition for dynamic complex networks. In *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*. IEEE, 488–496.

[24] Karl Mosler. 2012. *Multivariate Dispersion, Central Regions, and Depth: The Lift Zonoid Approach*. Vol. 165. Springer Science & Business Media.

[25] Pavlo Mozharovskyi, Julie Josse, and François Husson. 2020. Nonparametric imputation by data depth. *J. Amer. Statist. Assoc.* 115, 529 (2020), 241–253.

[26] N.N. Narisetty and V. Nair. 2016. Extremal Depth for Functional Data and Applications. *J. Amer. Statist. Assoc.* 111 (2016), 1505–1714.

[27] A. Nieto-Reyes and H. Battey. 2016. A topologically valid definition of depth for functional data. *Statist. Sci.* 31, 1 (2016), 61–79.

[28] Giannis Nikolentzos, Polykarpos Meladianos, Stratis Limnios, and Michalis Vazirgiannis. 2018. A Degeneracy Framework for Graph Similarity.. In *IJCAI*. 2595–2601.

[29] Tore Opsahl. 2011. Why anchorage is not (that) important: Binary ties and sample selection. *online] https://toreopsahl.com/2011/08/12/why-anchorage-is-not-that-important-binary-ties-and-sample-selection/* (2011).

[30] Tore Opsahl, Filip Agneessens, and John Skvoretz. 2010. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social networks* 32, 3 (2010), 245–251.

[31] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[32] Mukund Raj, Mahsa Mirzargar, Robert Ricci, Robert M Kirby, and Ross T Whitaker. 2017. Path boxplots: A method for characterizing uncertainty in path ensembles on a graph. *Journal of Computational and Graphical Statistics* 26, 2 (2017), 243–252.

[33] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[34] Carlo Sguera and Sara López-Pintado. 2020. A notion of depth for sparse functional data. *arXiv:2007.15413* (2020).

[35] M. Shanahan, V. P. Bingman, T. Shimizu, M. Wild, and O. Güntürkün. 2013. Large-scale network organization in the avian forebrain: a connectivity matrix and theoretical analysis. *Frontiers in computational neuroscience* 7 (2013), 89.

[36] Ali Sheharyar, Alexander Ruh, Maria Aristova, Michael Scott, Kelly Jarvis, Mohammed Elbaz, Ryan Dolan, Susanne Schnell, Kai Lin, James Carr, et al. 2019. Visual analysis of regional myocardial motion anomalies in longitudinal studies. *Computers & Graphics* 83 (2019), 62–76.

[37] S. Somin, G. Gordon, and Y. Altshuler. 2018. Network Analysis of ERC20 Tokens Trading on Ethereum Blockchain. In *ICCS*. 439–450.

[38] S. Somin, G. Gordon, and Y. Altshuler. 2018. Social Signals in the Ethereum Trading Network. *arXiv:1805.12097* (2018).

[39] Yahui Tian and Yulia R Gel. 2017. Fast Community Detection in Complex Networks with a K-Depths Classifier. In *Big and Complex Data Analysis*. Springer, 139–157.

[40] Y. Tian and Y. R. Gel. 2019. Fusing data depth with complex networks: Community detection with prior information. *Computational Statistics & Data Analysis* 139 (2019), 99–116.

[41] Friedhelm Victor and Bianca Katharina Lüders. 2019. Measuring ethereum-based erc20 token networks. In *International Conference on Financial Cryptography and Data Security*. Springer, 113–129.

[42] G. Vinue and I. Epifanio. 2020. Robust archetypoids for anomaly detection in big functional data. *Advances in Data Analysis and Classification* (2020), 1–26.

[43] R.T. Whitaker, M. Mirzargar, and R.M. Kirby. 2013. Contour Boxplots: A Method for Characterizing Uncertainty in Feature Sets from Simulation Ensembles. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2713–2722.

[44] G. Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151 (2014), 1–32.

[45] X. Zhang, Y. Tian, G. Guan, and Y. R. Gel. 2021. Depth-based classification for relational data with multiple attributes. *Journal of Multivariate Analysis* 184 (2021), 104732.

[46] Yang Zhang and Srinivasan Parthasarathy. 2012. Extracting analyzing and visualizing triangle k-core motifs within networks. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 1049–1060.