Characterizing and Utilizing the Interplay Between Core and Truss Decompositions

Penghang Liu
Department of Computer Science and Engineering
University at Buffalo
Buffalo, USA
penghang@buffalo.edu

Ahmet Erdem Sarıyüce
Department of Computer Science and Engineering
University at Buffalo
Buffalo, USA
erdem@buffalo.edu

***Abstract*—Finding the dense regions in a graph is an important problem in network analysis. Core decomposition and truss decomposition address this problem from two different perspectives. The former is a vertex-driven approach that assigns density indicators for vertices whereas the latter is an edge-driven technique that put density quantifiers on edges. Despite the algorithmic similarity between these two approaches, it is not clear how core and truss decompositions in a network are related. In this work, we introduce the vertex interplay (VI) and edge interplay (EI) plots to characterize the interplay between core and truss decompositions. Based on our observations, we devise CORE-TRUSSDD, an anomaly detection algorithm to identify the discrepancies between core and truss decompositions. We analyze a large and diverse set of real-world networks, and demonstrate how our approaches can be effective tools to characterize the patterns and anomalies in the networks. Through VI and EI plots, we observe distinct behaviors for graphs from different domains, and identify two anomalous behaviors driven by specific real-world structures. Our algorithm provides an efficient solution to retrieve the outliers in the networks, which correspond to the two anomalous behaviors. We believe that investigating the interplay between core and truss decompositions is important and can yield surprising insights regarding the dense subgraph structure of real-world networks.**

***Index Terms*—dense subgraph discovery, k-core decomposition, k-truss decomposition**

I. INTRODUCTION

Dense subgraphs in real-world networks contain significant and unusual information. There are many application domains where dense subgraphs are useful. A few use cases are finding price value motifs in the financial networks [1], locating spam link farms in webs [2]–[4], detecting DNA motifs in biological networks [5], and identifying the news stories from microblogging streams in real-time [6]. Dense regions are also used to improve the efficiency of computation heavy tasks like distance query computation [7] and materialized per-user view creation [8]. Core and truss decompositions are effective models to find dense regions with hierarchical relationships. In the core decomposition [9], [10], vertices are assigned density pointers, called core numbers, that indicate the cohesiveness in the neighborhood. Likewise, truss decomposition [11]–[14] yields truss numbers for edges which can be interpreted as the link strength.

Given the wide application space, core and truss decompositions are unified and extended by new models for different types of graphs [15], [16]. However, the relationship between the core and truss decompositions and its impact on the graph structure have been overlooked. By definition, a large truss number of an edge implies large core numbers on its endpoints. But, it is not clear what aspects of the graph structure are covered by each decomposition. Understanding the interplay between those measures can enable more effective network analysis.

In this work, we investigate the interplay between core and truss decompositions in real-world networks and random graphs. Our first contribution is the vertex interplay (VI) and edge interplay (EI) plots, which analyze the structure of dense subgraphs from two different perspectives. Then, we propose the CORE-TRUSSDD algorithm to identify the anomalies with respect to the core-truss interplay. The VI plot investigates the core-truss interplay by checking the core numbers of vertices and the truss numbers of their adjacent edges. The EI plot explores the interplay based on the truss numbers of edges and the core numbers of their two endpoints. We use several real-world networks from various domains and examine the

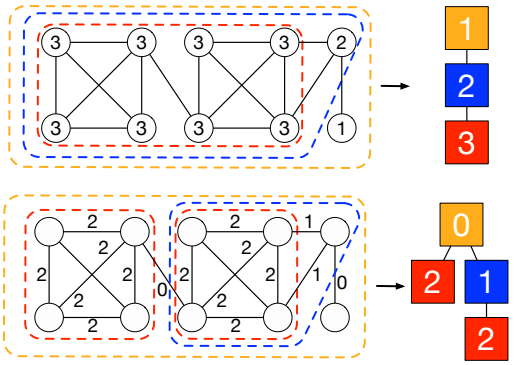


Fig. 1: **Examples for core (top) and truss (bottom) decompositions. At the top, core numbers are shown for each node and red, blue, and orange regions show the 3-, 2-, and 1-cores. They form a hierarchy by containment as denoted; 1-core contains 2-core and 2-core subsumes 3-core. For the same graph, trusses and truss numbers of edges are presented at the bottom. The entire graph is a 0-truss and the five nodes on the right form a 1-truss. There are two 2-trusses and one of them is a subset of the 1-truss, as denoted by the tree hierarchy.**

similarities and differences in the core-truss interplay. Our analysis on VI and EI plots gives interesting results for the core-truss interplay behavior in real-world networks. We also show that those behaviors cannot be captured when the edges are rewired with a realistic random graph model. Our algorithm provides an efficient solution to retrieve the outliers in the networks, which reveal the anomalous behaviors observed in the VI plots.

The rest of this paper is organized as follows. Section II gives the background for core and truss decomposition and random graph models. Section III describes the datasets used in the paper. In Section IV and V, we characterize the core-truss interplay using VI and EI plots. Based on our observations, we propose our anomaly detection algorithm in Section VI. After reviewing related work in Section VII, we draw conclusions in Section VIII. Codes for reproducing the results and figures are available at `https://github.com/penghangliu/Core-Truss`.

II. Background

Our study explores real-world networks which can be represented as a simple undirected unweighted graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. We represent the neighborhood of a vertex u (the set of nodes connected to u) as $N(u)$.

A. Core decomposition

The k-core subgraph is introduced by Seidman [9] for social networks analysis, and also by Matula and Beck [10] for clustering and graph coloring. **The *k*-core is a connected, maximal subgraph such that every node in the subgraph has degree of at least *k* within the subgraph.** The core number of a node is the highest value of k such that the node belongs to a k-core. We denote the core number of a vertex u by $K(u)$. The maximum core number of all vertices in the graph is defined as the *(core) degeneracy* [17].

Core decomposition is the process of finding the core numbers of nodes, which are used to locate all the k-core subgraphs. Batagelj and Zaversnik introduced an efficient iterative peeling algorithm that uses a bucket data structure to find the core numbers of nodes in $O(|E|)$ time [18]. Starting from the node with the minimum degree, peeling algorithm assigns the degree of a node as its core number and remove it from the graph — thus decrementing the neighbors' degrees if larger. This process continues until the graph is empty. k-core subgraph (for any k) is found by a traversal that visits all reachable nodes whose core numbers are at least k. The nested structure of k-cores reveals a hierarchy by containment. Figure 1 (top) presents an example for the core numbers, k-core subgraphs, and hierarchy.

B. Truss Decomposition

Higher-order structures, also known as motifs or graphlets, have been used to locate dense regions that cannot be detected otherwise with edge-centric methods [19], [20]. Finding the frequency and distribution of triangles and other small motifs in real-world networks is a simple yet effective approach used in data analysis [21]–[24]. The truss decomposition is inspired by the k-core and considers the edges and the triangles they participate in [11]–[14]. ***k*-truss is a connected, maximal subgraph such that every edge in the subgraph participates in at least *k* triangles within the subgraph**. The truss number of an edge is the highest k such that the edge is part of a k-truss. In the following, we denote the truss number of an edge (u, v) by $T(u, v)$. The maximum truss number of all edges in the graph is defined as the *truss degeneracy*.

Similar to the core decomposition, finding the truss numbers has two phases; 1) Counting the triangles that each edge participates in, 2) Peeling those counts by choosing the edge with the minimum count, assigning that as truss number, and decrementing triangle counts of neighbors. This requires $O(\sum_{v \in V} d(v)^2)$ time. k-trusses also exhibit a hierarchy by containment. Figure 1 (bottom) presents the truss numbers of edges, k-truss subgraphs, and their hierarchy on a toy graph.

Relationship between k-clique, and k-core, k-truss. The k-clique is a fully connected subgraph that contains k nodes, and each node is connected to the other $k-1$ nodes. For any edge (u, v) in a k-clique, both u and v are connected to the other $k-2$ nodes, thus every edge in the k-clique participates in at least $k-2$ triangles within the clique. This shows that k-clique is also a $(k-1)$-core and a $(k-2)$-truss.

C. Random Graph Models

Random graph models are commonly used as the null model for analyzing real-world networks. The Erdős-Rényi model (ER), configuration model, and Block Two-Level Erdős-Rényi (BTER) model [25] are three random graph models that simulate real-world networks at different levels of authenticity. The ER model rewires the edges randomly while keeping the graph size. The configuration model generates random graphs from the given degree sequence $\overrightarrow{k}$, where $k_u \in \overrightarrow{k}$ is the degree of vertex u. In the BTER model, the degree distribution and the clustering coefficient per degree distribution is preserved to the best extent. A graph is generated through two phases in the BTER model. In the first phase, vertices are clustered into communities and ER model is applied to generate edges within the same community. In the second phase edges between communities are generated regarding the size of the communities.

III. Datasets

In order to explore patterns in various types of real-world networks, we cover datasets from five different categories: social, autonomous systems, citation, collaboration, and web hyperlink networks. The datasets are obtained from SNAP [26], DBLP [27], and Konect [28]. Various statistics are summarized in Table I.

Social networks. Our dataset includes `Catster`, `Dogster`, `Flickr`, `Hamster`, `LiveJournal`, `YouTube`, and `Orkut` friendship networks that are formed among the members of the social networking websites. In addition, we consider the `Email` network among the employees of Enron Corporation.

We also include Email-Eu-core, a network with ground-truth community representing the emails between members of a large European research institution.

Autonomous systems. In this category, we have the five router networks, As-733, Caida, Oregon-2, and Skitter, and the Gnutella graph, which is the network among the hosts of Gnutella website.

Citation Networks. Here we have five networks including Cora, CiteSeer network extracted from the CiteSeer library, DBLP network among the papers in DBLP website, HepTh network among the publications in the arXiv's High Energy Physics Theory section, and Patent citation network among the patents registered with the United States Patent and Trademark Office.

Collaboration Networks. Here we consider three co-authorship networks from DBLP. DBLP-dm includes the authors and papers from top data mining conferences (SIGKDD, WWW, WSDM, ICDM, and SDM) in last ten years. DBLP-dbs is similarly constructed for the top database venues (VLDB, SIGMOD, and ICDE) and DBLP-pp is likewise for the top parallel processing conferences (IPDPS, HPDC, SC, and ICS).

Web Networks. We include five hyperlink networks that are constructed among websites (edge directions are ignored). Stanford, NotreDame, and BerkStan networks are formed in the domain of Stanford University, University of Notre Dame, and UC Berkeley and Stanford. Blogs contains the front-page hyperlinks between blogs in the context of the 2004 US election and Google is another hyperlink network released in 2002 by Google as a part of the Google Programming Contest.

We also consider the random graph models to validate our findings in real-world networks. For each real-world network, we generate 10 corresponding random graphs using BTER model and consider the average numbers for core and truss degeneracy, as shown in Table I. We also generate random graphs with ER and configuration models, but the results are not shown as they fail to provide any close approximates for core and truss degeneracy. The core degeneracy of the real-world networks ranges from 6 to 568, and the truss degeneracy ranges from 2 to 350. The degeneracy of social networks are commonly higher than the other types. Core degeneracy can be approximated very well by the BTER model for most datasets. However, the BTER model is less successful in capturing the truss degeneracy.

IV. Vertex Based Analysis

The core number of a vertex quantifies the cohesiveness around it. Truss numbers serve the same purpose for the edges. By definition, we can say that a vertex with a low (high) core number should be connected to some vertices with low (high) core numbers. It is not clear though, how the core number of a vertex is related to the truss numbers of its edges. In this section, we analyze the interplay of core and truss numbers from the perspective of a vertex. We introduce the **vertex interplay (VI) plot** to demonstrate the spectrum of edges around vertices with a particular core number. We show that the VI plots of networks from the same domain present

TABLE I: **Statistics of real-world networks and random graphs. The last four columns show the core degeneracy and truss degeneracy numbers. In each group, *Exact* shows the core and truss degeneracy numbers of real-world graphs, and *BTER* presents the same numbers (on average) for the random graphs generated with the BTER model. BTER model shows good capability of approximating the core degeneracy, but it does not perform well in capturing the truss degeneracy.**

Category	Name	$\|V\|$	$\|E\|$	Core degeneracy		Truss degeneracy	
				Exact	*BTER*	*Exact*	*BTER*
Social	Hamster	1.86K	12.5K	**20**	17	**7**	5
	Email	36.7K	184K	**43**	47	**20**	27
	YouTube	1.13M	2.99M	**51**	78	**17**	48
	Catster	150K	5.45M	**419**	312	**205**	164
	Dogster	427K	8.54M	**248**	300	**91**	192
	Flickr	1.72M	15.6M	**568**	310	**276**	110
	LiveJournal	4.00M	34.7M	**360**	33	**350**	6
	Orkut	3.07M	117M	**253**	80	**76**	40
	Email-Eu-core	1.00K	25.5K	**34**	36	**21**	11
Autonomous sys.	As-733	6.47K	12.6K	**12**	13	**8**	11
	Oregon-2	10.9K	31.2K	**31**	22	**23**	16
	Caida	26.5K	53.4K	**22**	28	**14**	23
	Gnutella	62.6K	148K	**6**	6	**2**	1
	Skitter	1.70M	11.1M	**111**	191	**66**	146
Citation	DBLP	12.6K	49.6K	**12**	13	**7**	5
	Cora	23.2K	89.2K	**13**	9	**9**	3
	HepTh	27.7K	352K	**37**	31	**28**	14
	CiteSeer	384K	1.74M	**15**	14	**11**	3
	Patent	3.78M	16.5M	**64**	9	**34**	1
Collaboration	DBLP_pp	8.41K	22.9K	**44**	16	**43**	3
	DBLP_dbs	8.10K	23.0K	**35**	10	**34**	2
	DBLP_dm	16.4K	33.9K	**24**	7	**23**	1
Web	Blogs	1.22K	16.7K	**36**	38	**23**	12
	NotreDame	326K	1.09M	**155**	144	**153**	47
	Stanford	282K	1.99M	**71**	123	**60**	93
	Google	876K	4.32M	**44**	86	**42**	72
	BerkStan	685K	6.65M	**201**	258	**199**	178

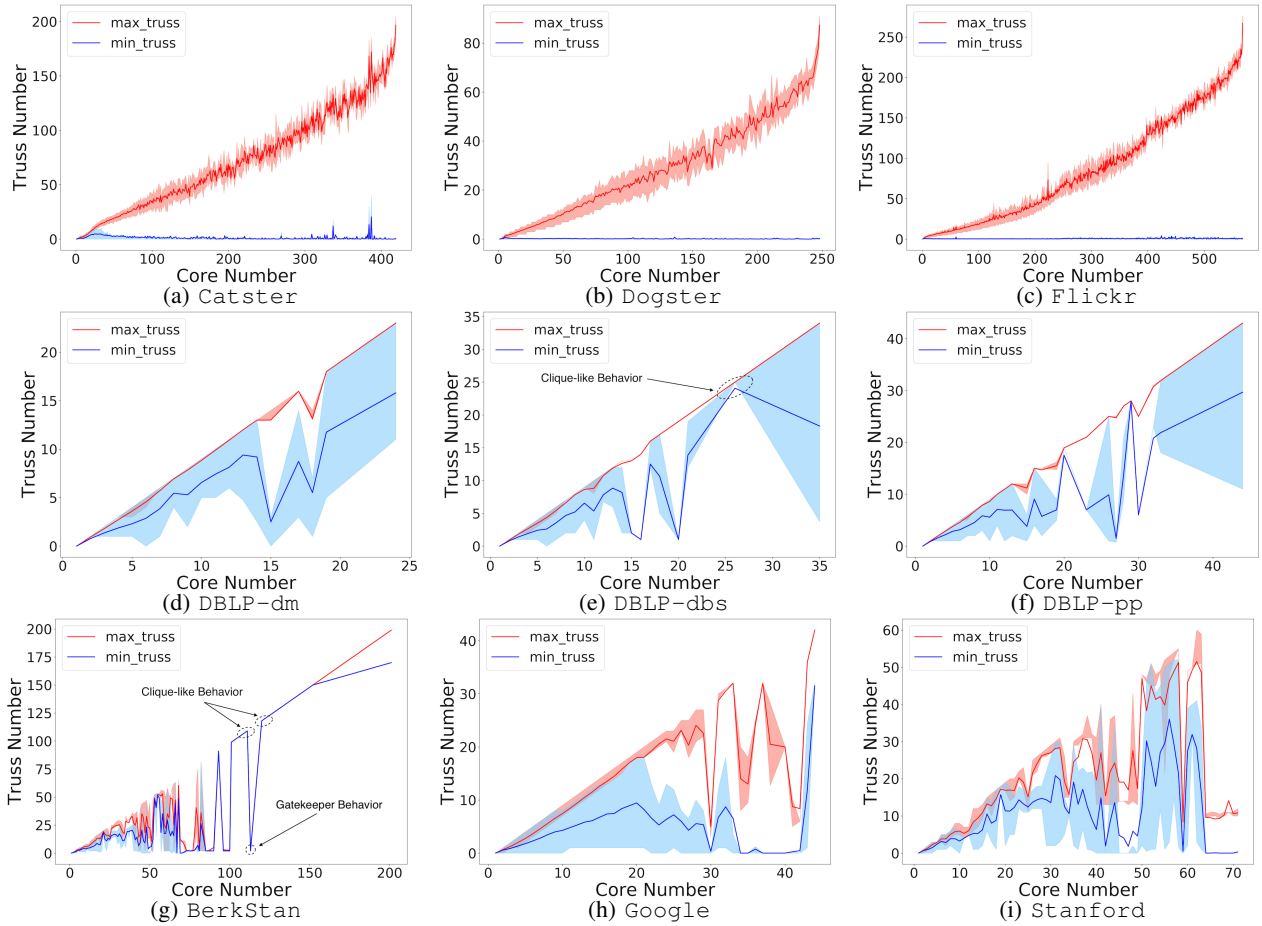
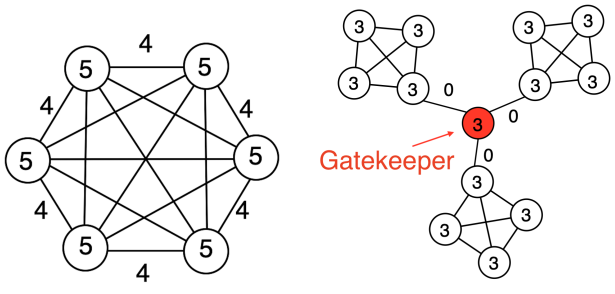


Fig. 2: **Vertex interplay (VI) plots for some real-world graphs (all are available in Figures ?? and ??). For each vertex with a particular core number, the maximum and minimum of the truss numbers of surrounding edges are shown. Average and interquartile ranges are computed over the vertices with the same core number. Social networks in Figures 2a, 2b, 2c exhibit a common (and expected) behavior, where maximum truss numbers of surrounding edges is larger for vertices with large core numbers whereas the minimum truss numbers of those stay low for all the core numbers. Collaboration networks, shown in Figures 2d, 2e, 2f, present a different picture; vertices with large core numbers are only surrounded by edges with large truss numbers (increasing minimum truss numbers). Lastly, web networks' characteristic behaviors (presented in Figures 2g, 2h, 2i) are unlike others, where the maximum and minimum truss numbers demonstrate non-monotonic behavior since some vertices with large core numbers serve as a gatekeeper among dense regions (thus surrounded with low truss numbers.)**

a consistent behavior and the ones from different domains show a variety. We first investigate the VI plots of real-world networks, and then compare our findings to the VI plots of the corresponding randomized networks.



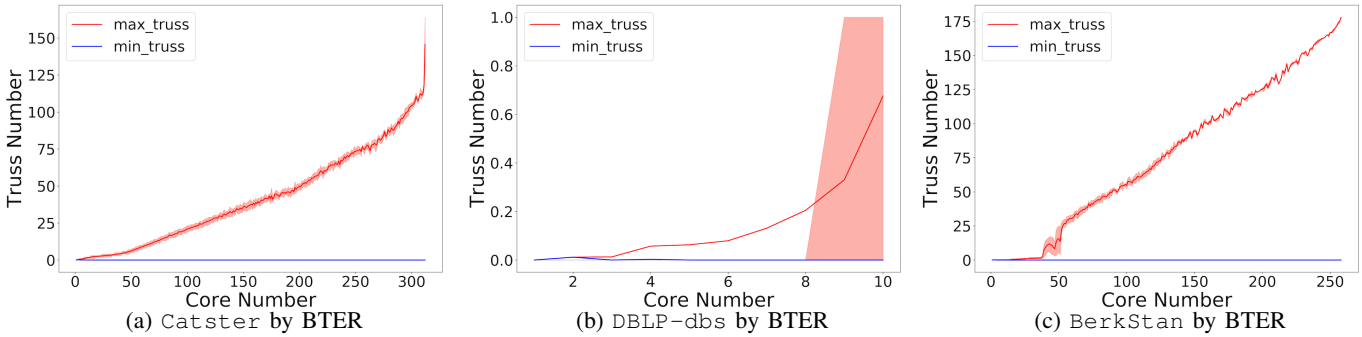
(a) Clique-like structure (b) Gatekeeper structure

Fig. 3: **Examples of clique-like and gatekeeper structure behaviors. In the 6-clique, the core numbers of vertices are 5 and the truss numbers of edges are 4. The gatekeeper (red) has three neighbors that are isolated from each other. Although the core number (3) shows that the gatekeeper belongs to a cohesive subgraph, its neighborhood structure is not cohesive, as truss numbers are zero.**

A. *Real-world Networks*

Fig. 2 presents the VI plots for some real-world networks in our dataset (all are available in Figures **??** and **??**). We examine the neighborhood of vertices with a particular core number and consider the maximum and minimum truss numbers of edges adjacent to a given vertex for a fixed core number. By this, we aim to understand how the core number of a vertex shapes the spectrum of neighborhoods around it. If there are multiple vertices with the same core number, we report the average and interquartile ranges. Formally, for each core number c, we find the set $S = \{u \in V : K(u) = c\}$ and compute $\frac{1}{|S|}\sum_{u \in S} min\{T(u,v) : v \in N(u)\}$ (likewise for maximum) along with the interquartile ranges.

One general behavior we observe is that the maximum truss number of adjacent edges is strongly correlated to the core number of the vertex; larger core numbers yield larger truss



(a) Catster by BTER (b) DBLP-dbs by BTER (c) BerkStan by BTER

Fig. 4: **Vertex interplay (VI) plots for random graphs generated from three real-world datasets. For each vertex with a particular core number, the maximum and minimum of the truss numbers of surrounding edges are shown. Average and interquartile ranges are computed over the vertices with same core number. Fig. 4a, Fig. 4b, and Fig. 15f give the VI plots of BTER model for Catster, DBLP-dbs, and BerkStan. BTER model can approximate the most general behavior of core-truss patterns to some extent, as we observed in Catster. However, it fails to preserve the interesting behavior in DBLP-dbs and BerkStan, i.e., the clique-like and gatekeeper structures.**

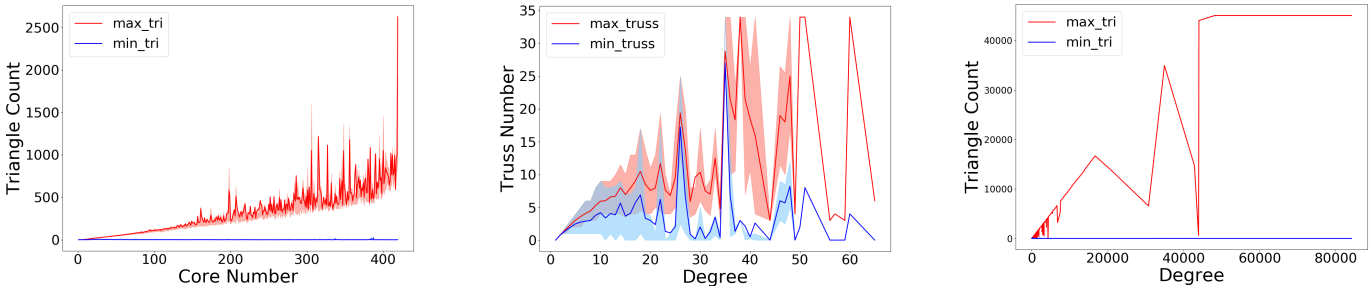
numbers overall. Figures 2a, 2b, and 2c show this pattern in the VI plots for social networks: Catster, Dogster, and Flickr. Regarding the minimum truss numbers, on the other hand, there is a consistent trend for all the vertices regardless of their core numbers. Every vertex in the network is connected to at least one edge with a very low truss number. This is in line with the core-periphery structure [29]; each vertex in the core block is connected to both core and periphery blocks – truss numbers of the edges between core and periphery are likely to be low. Note that the variation for vertices with the same core number is also very small, i.e., interquartile ranges are narrow, suggesting a high similarity among those vertices. We observe this behavior in 8 (of 9) social networks – LiveJournal exhibits a zigzag trend for the minimum truss numbers, probably because of the large range of core and truss values (it has the largest degeneracy numbers among all the networks). Autonomous systems exhibit a similar behavior as well; the minimum truss number line has small zigzags in some cases.

Collaboration networks exhibit a different behavior in the VI plots. Figures 2d, 2e, and 2f present the DBLP-dm, DBLP-dbs, and DBLP-pp networks. The minimum truss numbers are very close to the maximum ones, as opposed to the consistent trend in the social networks. Vertices with large core numbers are not connected to any edge with a low truss number. This implies that cliques of vertices with large core numbers are surrounded by some other cliques with close core numbers. In some cases the minimum truss number, maximum truss number, and core number are almost equal, indicating both k-core and k-truss are derived from a clique-like structure (Fig. 3a), as annotated in Fig. 2e. In fact, a collaboration network can be seen as a union of cliques, where authors of each paper form a clique, which in turn yields closed clique-like structures that are not connected to the nodes in the periphery.

We observe yet another distinct behavior in web networks. Figures 2g, 2h, and 2i show the VI plots for BerkStan, Google, and Stanford networks. The maximum truss numbers are very small for some vertices with large core numbers, implying that the truss numbers of all adjacent edges are small. Most of the neighbors are isolated from each other, indicating a sparse neighborhood despite the cohesiveness suggested by the large core number. Note that a great amount of the neighbors also have a large core number, by definition. Those vertices serve as structural holes in the network [30], as illustrated in Fig. 3b where a node with a large core number is connecting multiple cores isolated from each other. In Fig. 2g, there are 271 vertices with core number 113 but the largest truss number adjacent to those is only 2. Note that, in the same network, the vertices with core number 111 and 120 are surrounded by some very large truss numbers, indicating a clique-like structure.

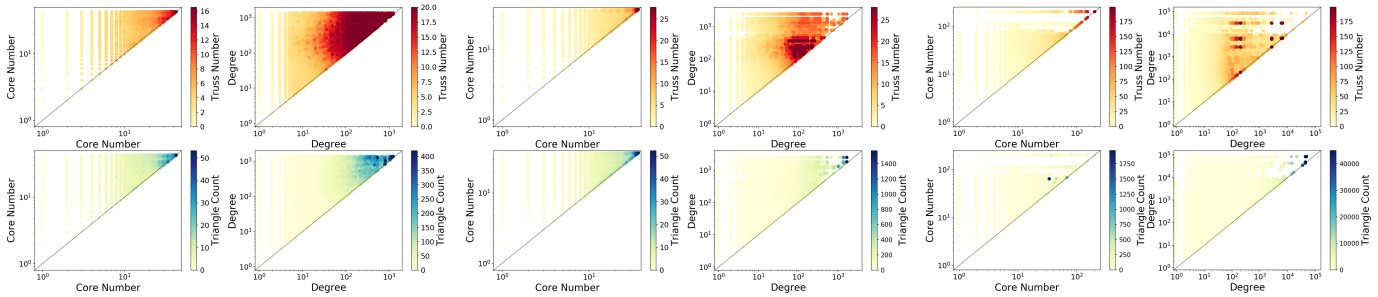
B. Random Graphs

Here we investigate if the observed behaviors in the VI plots are artifacts of some graph characteristics. For this purpose,



(a) Core numbers and triangle counts (b) Degrees and truss number (c) Degrees and triangle counts

Fig. 5: **Interplays between alternative measures. Fig. 5a, Fig. 5b, and Fig. 5c show the interplay in Catster, DBLP-dbs, and BerkStan. All fail to generate a consistent and pervasive pattern.**



(a) Email (b) HepTh (c) BerkStan

Fig. 6: **Edge interplay (EI) plots for some real-world graphs. For each pair of endpoints with particular core numbers/degrees, the average of the truss numbers/triangle counts of the edges are shown. Social network in Figure 6a exhibits a common and expected behavior, where the truss number/triangle count of edge is larger for two endpoints with large core numbers/degrees. Citation network, shown in Figures 6b, presents a different behavior where the edges with large truss numbers are connecting two endpoints with non-maximum degrees. Web network, shown in Figure 6c, presents another behavior where the two endpoints of the edges with large truss number have very different degrees.**

we randomize all the real-world networks with the BTER model [25]. As explained in Section II, the BTER model rewires the edges in a reference network by preserving the clustering coefficient per degree distribution to the best extent.

Fig. 4 presents the results for Catster, DBLP-dbs, and BerkStan networks, each represents a different behavior in Fig. 2. Random graphs generated by BTER model capture the common pattern observed in social networks but fail to capture the other interesting behaviors in collaboration and web graphs The low-maximum and high-minimum behaviors, which are driven by the unusual structures (cliques and structural holes), seem to be distinctive characteristics in real-world graphs.

Remark. Given the benefit of analyzing the interplay of core and truss numbers, it is natural to think about using just the degrees of vertices and triangle counts of edges for a similar analysis. However, the skewed distributions of degrees/triangle counts prevent such analyses; the VI plots become inconsistent for the networks from the same domain and behaviors can be gamed with simple changes in the graph. Fig. 5 presents a few examples for the networks analyzed above. In all variations (degree-triangle count, degree-truss, or core-triangle count) the VI plots fail to generate a consistent and pervasive pattern. **Core and truss numbers can be considered as the regularized and more robust versions of the vertex degrees and edges' triangle counts.**

Summary. VI plots present a meaningful graph summary by showing the interplay between core and truss numbers. Considering the cohesiveness around vertices (i.e., core numbers) with respect to various neighborhoods they are involved in (i.e., truss numbers) is an effective way to understand the dense regions and structural holes. VI plots of networks belonging to the same domain exhibit consistent behaviors whereas the ones from different domains suggest diverse characteristics. We believe that VI plots would be handy for domain practitioners to analyze the network structure.

V. Edge Based Analysis

In Section IV, we have explored the neighborhood around a vertex with a specific core number. It is not clear though, how the truss number of an edge is related to the core numbers of its two endpoints. In this section we address the interplay from the perspective of an edge. We introduce the **edge interplay (EI) plot** to capture the truss number of an edge between two endpoints with specific core numbers. We observe consistent behavior in the EI plots of networks from the same domain, and significant difference between the networks from different domains. Similar to the previous section, we apply our analysis on both real-world graphs and random graphs, and we also consider the degree and triangle count as alternative measures.

A. Real-world Networks

Figure 6 presents the EI plots for some real-world networks in our dataset (all are available in Figures 13 and 14). Here we examine the truss number of edges between two vertices with particular core numbers. If there are multiple edges having the same pair of core numbers for their endpoints, we show the mean value of the truss numbers. Formally, for each pair of core numbers $c_1 \leqslant c_2$, we find the set $S = \{(u,v) \in E : K(u) = c_1, K(v) = c_2\}$ and compute $\frac{1}{|S|}\sum_{(u,v)\in S} T(u,v)$. In addition to the core-truss interplay, we consider the vertex degrees and edges' triangle counts for a similar analysis. For all EI plot variations (degree-truss, degree-triangle count, and core-triangle), we observe consistent behaviors in networks from the same domains. Those alternative EI plots allow us to discover some interesting behaviors.

One general behavior we observe is that the truss number of an edge is strongly correlated to the core numbers of the two endpoints. This also holds true for all the EI plot variations regarding vertex degrees and edge triangle counts. Figure 6a shows this pattern in the EI plot of Email network, and we observe this pattern in all social networks, autonomous systems, and collaboration networks. Note that the dense red regions are small, suggesting a high similarity between the endpoints of edges with large truss numbers.

Citation networks present a different behavior in the EI plots. Figure 6b presents the EI plot of `HepTh` network. In the degree-truss interplay, edges with large truss number are connecting two endpoints with non-maximum degrees. Another interesting behavior appears consistently in web networks. Figure 6c presents the EI plot of `BerkStan` network. Edges with large truss number are connecting two vertices with very different degrees. This implies that there are cohesive structures connecting core and periphery blocks. This structure seems to be artificially constructed for special purposes, e.g., spam link farms.

B. Random Graphs

Here we generate the random graphs with BTER model, by preserving the clustering coefficient distribution per degree. The EI plots of random graphs are shown in Fig. 15 due to the space limit. The BTER model captures the general pattern observed in social networks, autonomous systems, and collaboration networks, but fails to capture the interesting behaviors in web networks and citation networks. This indicates the interesting behaviors we observed in EI plot are, again, distinctive characteristics of real-world networks which are driven by specific network structures.

Summary. Similar to the VI plots, the EI plots present a meaningful graph summary by showing the interplay from the edge perspective. In addition to the core and truss interplay, the EI plots are also capable to capture consistent behavior in the interplay between vertex degrees and edges' triangle counts. This grants us the ability to distinguish diverse characteristics of networks from more domains.

VI. Anomaly Detection by Core-Truss Discrepancy

Based on our observations in Section IV and Section V, we design the Core-Truss discrepancy detection algorithm (Core-TrussDD) to detect the vertices showing anomalous behaviors of core-truss interplay. We first compute the truss-profile of each vertex, and then cluster the vertices based on their truss-profiles. Within each cluster, we identify the outliers by checking the core number distribution and the Z-scores of the vertices. We apply our proposed algorithm on `Email-Eu-core` and `BerkStan`, and the results show that our algorithm reveals interesting anomalies in real-world network structures.

A. Core-TrussDD *algorithm*

Algorithm 1 provides the pseudocode of Core-TrussDD. We first run the core and truss decompositions (Line 2 and 3) to compute the core numbers of all vertices and the truss numbers of all edges. Since a vertex can have many edges with different truss numbers, truss numbers cannot be directly used as vertex features. We introduce the vertex truss-profile, which represents the spectrum of truss numbers of all adjacent edges for a vertex. It is the probability distribution of truss numbers around a vertex. This allows us to measure the similarity between two vertices with respect to the truss numbers. We create truss-profiles of vertices in Lines 5 to 8.

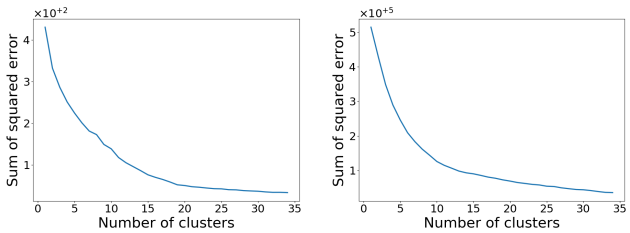
Algorithm 1 Core-TrussDD

Input: $G = (V, E)$
Output: The set A of anomalous vertices

1: $A \leftarrow \emptyset, \mathcal{P} \leftarrow \emptyset$
2: $K \leftarrow$ Core-decomposition(G) ▷ compute core numbers for all vertices
3: $T \leftarrow$ Truss-decomposition(G) ▷ compute truss numbers for all edges
4: $threshold \leftarrow sort(K)[\frac{|V|}{4}]$ ▷ filter out nodes with low core numbers
5: **for all** $v \in V$ **do**
6: **if** $K[v] \geq threshold$ **then**
7: $\vec{P_v} \leftarrow$ truss-profile of v ▷ based on T
8: $\mathcal{P}$.push$(\vec{P_v})$
9: $k \leftarrow$ Elbow-method$(\mathcal{P})$ ▷ optimum number of clusters
10: $[C_1, C_2, \cdots, C_k] \leftarrow$ k-means$(\mathcal{P}, k)$ ▷ vertex clustering
11: **for** $C_i \in [C_1, C_2, \cdots, C_k]$ **do**
12: **for** $v' \in C_i$ **do**
13: $z[v'] \leftarrow$ Z-score of $K[v']$ in cluster C_i
14: **if** $|z[v']| > 2$ **then**
15: A.push(v')
16: $A \leftarrow A \cup$ outliers identified from the histogram
17: **return** A

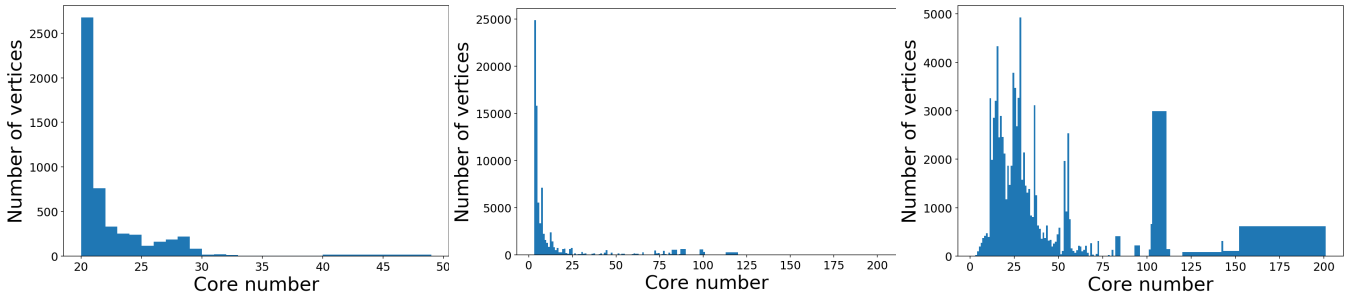
Definition 1 (Vertex Truss-profile): Given a graph $G = (V, E)$ and its truss degeneracy $max(T)$, the truss-profile of vertex $v \in V$ is a vector $P_v = [p_0, p_1, \cdots, p_{max(T)}]$, where $p_i = \frac{n(i)}{\sum_{i=0}^{max(T)} n(i)}$ and $n(i)$ is the number of adjacent edges having truss number i.

Real-world networks often present heterogeneous structures. It is hard to capture a general pattern in truss-profiles of all vertices as they can be highly diverse, making anomaly detection challenging. To address this issue, we group the vertices with similar truss-profiles (Line 10). Here we apply the k-means clustering [31], which minimizes the within-cluster squared distances. One important issue of the k-means



(a) `Email-Eu-core` (b) `BerkStan`

Fig. 7: **Exploring the number of clusters using elbow method. Each figure shows the decrease of within-cluster sum of squared error (SSE), with respect to the number of clusters. For each dataset, we select the number of clusters where the SSE stops decreasing. We choose 15 clusters for both `Email-Eu-core` and `BerkStan` according to Fig. 7a and Fig. 7b.**

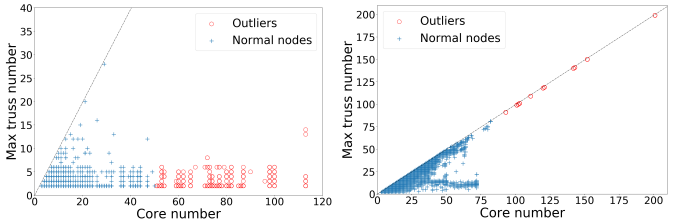


(a) Normal cluster (b) Cluster G (gatekeeper anomalies) (c) Cluster C (clique-like anomalies)

Fig. 8: **Core number distributions in `BerkStan` clusters. Each figure represents the core number distribution within a cluster. Fig. 8a shows a cluster where all vertices have similar core numbers. This pattern is commonly observed (in 13 out of 15 clusters). Fig. 8b shows another cluster where we observe outliers with core numbers deviated from the major population ($K > 50$). These outliers are found to be the gatekeepers in later analysis. We also observe anomalous vertices in Fig. 8c ($K > 150$), which appear to be clique-like structures.**

algorithm is the input parameter k, which determines the number of clusters. We use elbow method [32] to select the optimum value of k for each dataset, regarding the within-cluster sum of squares (Line 9). Fig. 7a and Fig. 7b show how the within-cluster sum of squared error (SSE) changes when the number of clusters increases. We use k-means with 15 clusters for both `Email-Eu-core` and `BerkStan`, since the SSE is hardly reduced by having more clusters. Another issue of clustering is that each vertex will be assigned to a cluster, including those in the periphery with low importance. This will downgrade the clustering performance and lead to false positives in the anomaly detection. Therefore, we ignore the 25% of nodes with the lowest core numbers (Lines 4 and 6).

After clustering the vertices with respect to the truss-profiles, we investigate the patterns and anomalies of core numbers in each cluster. In each cluster, we check the core numbers of the vertices and report the ones that are significantly different than the others. For this purpose, we compute the Z-score of each vertex in order to measure the deviation of their core numbers with respect to the general patterns (Line 13). For a given vertex v with core number K_v the Z-score is calculated as $z_v = \frac{K_v - \mu}{\sigma}$, where μ and σ are the mean and standard deviation of all core numbers in the cluster. In Lines 14 and 15, we identify the outliers which have the absolute values of Z-scores larger than 2. This corresponds to the 2.28% of the population in normal distribution. In addition to the Z-score, we plot the distribution of core numbers and identify anomalies which deviate from the general patterns (Line 16).



(a) Cluster G (gatekeepers) (b) Cluster C (clique-like outliers)

Fig. 9: **Gatekeeper and clique-like anomalies in `BerkStan` clusters. In each scatter plot, the blue crosses denote normal nodes and the red circles represent outliers. The x-axis represents the core number of the vertex, while the y-axis denotes the maximum truss number of the adjacent edges. The dashed line is $y = x - 1$, which indicates the core-truss behavior of cliques. Fig. 9a shows the patterns of gatekeeper outliers in cluster G, where the maximum truss numbers are much less than the core numbers. Fig. 9b gives the clique-like outliers behaviors in cluster C, where the maximum truss numbers are almost equal to the core numbers.**

Remark. Note that the k-means algorithm only serves as a subroutine in our anomaly detection algorithm. The purpose of clustering is to partition the vertices into several groups where we can expect common truss-profile behaviors. The clustering performance is not our major concern as we only aim to have a scaled-down problem size for the anomaly detection.

Time complexity: CORE-TRUSSDD starts with core and truss decomposition, which takes $O(|E|)$ and $O(\sum_{v \in V} d(v)^2)$ time respectively. The algorithm generates truss-profiles for all vertices in $O(|V|max(T)) = O(|V|)$ time, since the truss degeneracy $max(T)$ is $O(1)$ for real-world networks. The k-means clustering can be implemented by the Lloyd's algorithm [33], which takes $O(k|V|max(T)i)$ time, where k is the number of clusters and i is the number of iterations until convergence. In the worst case the Lloyd's algorithm needs $2^{\sqrt{|V|}}$ iterations. Therefore, the overall time complexity is $O(\sum_{v \in V} d(v)^2 + k|V|2^{\sqrt{|V|}})$.

Space complexity: We first construct K and T to store the core numbers of vertices and truss numbers of edges, which take $O(|E|)$ space. In addition, we need $O(|V|max(T)) = O(|V|)$ space for the truss profiles $\mathcal{P}$. The k-means algorithm need $O((|V| + k)max(T)) = O(|V|)$ space. Arrays A and $[C_1, C_2, \cdots, C_k]$ take $O(|V|)$ space, so the total space complexity is $O(|E|)$.

B. Anomalies in real-world networks

In general, we observe similar core numbers within the clusters. Fig. 8a gives the core number distribution in one of the clusters in `BerkStan`. Within the cluster the major population of core numbers lies in a small interval $[20, 30]$. Since the vertices are clustered based on their truss-profiles, it

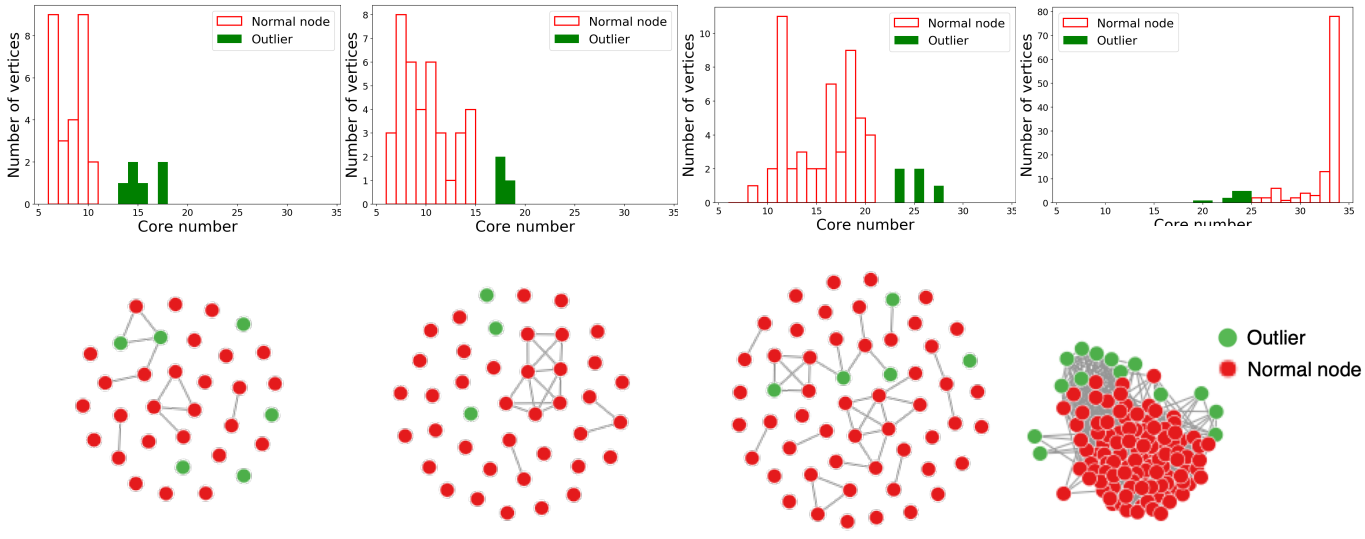
indicates that the core and truss decompositions are consistent. We observed similar consistent patterns in 13 out of 15 clusters in `BerkStan`, as well as in the most clusters in `Email-Eu-core`. In the remaining two clusters (clusters G and C) of `BerkStan`, we identify significant amounts of outliers (more than 5%). Fig. 8b shows the core number distribution in cluster G. The majority of vertices have core numbers ranging from 0 to 25, while some outliers have much larger core numbers (note that the core degeneracy of `BerkStan` is 201). Fig. 9a shows that the core numbers of these outliers are much greater than the maximum truss numbers of their adjacent edges. This indicates that the anomalies discovered in this cluster are the gatekeepers illustrated in Fig. 3b. In Fig. 8c we observe that most vertices in cluster C have their core numbers lie between 0 and 55. We identify 5352 outlier vertices with core number larger than 92. As shown in Fig. 9b, the core numbers are extremely close to the maximum truss numbers of the adjacent edges. For all of the 5352 outliers, the difference between their core numbers and the maximum truss numbers is equal to 2. According to our discussion in Section IV, these are the clique-like structures (Fig. 3a) in the network.

Next, we investigate the patterns and anomalies in the `Email-Eu-core` clusters. Among the 15 clusters four of them present anomalous behaviors. Fig. 10 shows the core number distributions of the four clusters, along with the subgraphs induced by the vertices in the clusters. In cluster A we observe the majority of vertices have core numbers from 6 to 10, while there are six outliers with core numbers larger than 12. Similar anomalies are observed in clusters B and C, where the outliers have core numbers significantly larger than the majority of the population in the cluster. Cluster D presents a different picture, where most of the vertices have core numbers of 33 and the outliers have core numbers less than 25. In order to further analyze these anomalous clusters, we visualize the subgraphs based on the edges within the clusters and display them in Fig. 10. Note that we are not doing graph clustering but cluster the vertices with respect to the truss-profiles, so they may not be connected at all. We noticed that subgraphs induced by clusters A, B, and C are structurally sparse. Vertices with similar truss behavior are often not connected to each other in real-world networks. Moreover, outliers are less likely to be connected with each other than the normal vertices. The subgraph of cluster D shows a very different picture, where the normal vertices form a connected component with relatively high density. This is due to the overall high core numbers in the cluster. Note that the outliers lie in the periphery of the subgraph and few connections exist between them.

Summary. CORE-TRUSSDD algorithm creates the truss-profile to present the truss number spectrum for a given vertex, which allows us to cluster the vertices with similar truss behavior. We then analyze the core number distribution and the Z-scores in the cluster to identify the anomalies. These outliers reveal the clique-like and gatekeeper structures in the networks. By further investigating the subgraphs induced from the clusters, we discover distinct interaction patterns between the outliers.

VII. Related Work

The core and truss decompositions, proposed by [9] and [12], is commonly used in dense subgraph discovery. One of the works that addresses the relation between core and truss decompositions is [15]. Sariyuce et al. proposed a network



(a) Cluster A (b) Cluster B (c) Cluster C (d) Cluster D

Fig. 10: **Core number distributions and the corresponding subgraphs of `Email-Eu-core` clusters. The histograms present the core number distributions within the clusters, while the subgraphs at the bottom are induced by the vertices in the clusters. Normal nodes are colored in red and the green circles denote outliers. Fig. 10a, Fig. 10b, and Fig. 10c show the pattern of three clusters, where vertices are sparsely connected. Fig. 10d presents another cluster which is densely connected. When compared to the normal nodes, the outliers are less likely to be connected with each other, as observed in all clusters.**

decomposition framework which addresses both core and truss decompositions, and explored the hierarchical and overlap relations among the subgraphs. Another related work [16] defined a subgraph structure that combines the definitions of k-core and k-truss, and proposed the corresponding decomposition algorithm.

The most related work to our study is introduced by Shin et al [34]. They explored the relations between core (truss) degeneracy and the number of triangles. Note that, our research directly addresses the relationship between the core and truss decompositions based on local perspectives. We do not limit our analysis to investigate each decomposition on its own. Instead we focus on the interplay of core and truss numbers and introduce the VI and EI plots for network analysis. We also propose the CORE-TRUSSDD algorithm to detect the outlier nodes.

VIII. CONCLUSION

In this paper, we analyzed the interplay between core and truss decompositions in real-world networks. We introduced VI and EI plots to analyze the network structure by using the core and truss decompositions. The VI plot investigates the core-truss interplay from a vertex perspective, by examining the spectrum of edges around vertices with particular core numbers. The EI plot explores the interplay from an edge perspective by checking the truss number of an edge and the core numbers of the two endpoints. We applied our analysis on real-world networks from various domains, and then validate our findings by evaluating the random graphs generated by the BTER model. The VI and EI plots reveal consistent pattern in social networks and autonomous systems, and identify some interesting behaviors in collaboration networks, citation networks, and web networks. Inspired by our observation in VI and EI plots, we proposed the CORE-TRUSSDD algorithm to identify anomalies in the networks by utilizing the core-turss interplay . We analyzed the characteristics of the outliers indentified by our algorithm, and the results support our findings in the VI plots. We believe our study would be handy for domain practitioners to analyze the dense subgraph structure of networks, and provide important insights for anomaly detection.

REFERENCES

[1] X. Du, R. Jin, L. Ding, V. E. Lee, and J. H. T. Jr., "Migration motif: a spatial - temporal pattern mining approach for financial markets." in *KDD*, 2009, pp. 1135–1144.

[2] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Trawling the web for emerging cyber-communities," in *WWW*, 1999, pp. 1481–1493.

[3] D. Gibson, R. Kumar, and A. Tomkins, "Discovering large dense subgraphs in massive graphs," in *VLDB*, 2005, pp. 721–732.

[4] Y. Dourisboure, F. Geraci, and M. Pellegrini, "Extraction and classification of dense communities in the web," in *WWW*, 2007, pp. 461–470.

[5] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou, "Motifcut: regulatory motifs finding with maximum density subgraphs." in *ISMB*, 2006, pp. 156–157.

[6] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava, "Dense subgraph maintenance under streaming edge weight updates for real-time story identification," *PVLDB*, vol. 5, no. 6, pp. 574–585, Feb. 2012.

[7] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry, "3-hop: a high-compression indexing scheme for reachability query." in *SIGMOD Conf.*, 2009, pp. 813–826.

[8] A. Gionis, F. Junqueira, V. Leroy, M. Serafini, and I. Weber, "Piggybacking on social networks," *PVLDB*, vol. 6, no. 6, pp. 409–420, 2013.

[9] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983.

[10] D. Matula and L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *JACM*, vol. 30, no. 3, pp. 417–427, 1983.

[11] K. Saito and T. Yamada, "Extracting communities from complex networks by the k-dense method," in *ICDMW*, 2006.

[12] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," vol. National Security Agency Technical Report, 2008.

[13] A. Verma and S. Butenko, "Network clustering via clique relaxations: A community based approach," in *DIMACS Graph Part. and Clustering*, 2012, pp. 129–140.

[14] Y. Zhang and S. Parthasarathy, "Extracting analyzing and visualizing triangle k-core motifs within networks," in *ICDE*, 2012, pp. 1049–1060.

[15] A. E. Sarıyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek, "Nucleus decompositions for identifying hierarchy of dense subgraphs," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 3, p. 16, 2017.

[16] Z. Li, Y. Lu, W.-P. Zhang, R.-H. Li, J. Guo, X. Huang, and R. Mao, "Discovering hierarchical subgraphs of k-core-truss," *Data Science and Engineering*, vol. 3, no. 2, pp. 136–149, Jun 2018.

[17] P. Erdős and A. Hajnal, "On chromatic number of graphs and set-systems," *Acta Mathematica Hungarica*, vol. 17, pp. 61–99, 1966.

[18] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," Arxiv, Tech. Rep. cs/0310049, 2003.

[19] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[20] C. Tsourakakis, "The k-clique densest subgraph problem," in *Proc. of the 24th International Conf. on World Wide Web*, ser. WWW '15, 2015, pp. 1122–1132.

[21] M. Jha, C. Seshadhri, and A. Pinar, "Path sampling: A fast and provable method for estimating 4-vertex subgraph counts," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15, 2015, pp. 495–505.

[22] A. Pinar, C. Seshadhri, and V. Vishal, "ESCAPE: efficiently counting all 5-vertex subgraphs," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1431–1440.

[23] N. K. Ahmed, J. Neville, R. A. Rossi, and N. G. Duffield, "Efficient graphlet counting for large networks," in *2015 IEEE International Conference on Data Mining, ICDM 2015*, 2015, pp. 1–10.

[24] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Estimation of graphlet statistics," *CoRR*, vol. abs/1701.01772, 2017. [Online]. Available: http://arxiv.org/abs/1701.01772

[25] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of erdos-renyi graphs," *Phys. Rev. E*, vol. 85, p. 056109, May 2012.

[26] J. Leskovec and A. Krevl, "SNAP Datasets," Jun. 2014.

[27] M. Ley, "The dblp computer science bibliography: Evolution, research issues, perspectives," in *International symposium on string processing and information retrieval*. Springer, 2002, pp. 1–10.

[28] J. Kunegis, "Konect: the koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 1343–1350.

[29] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Social Networks*, vol. 21, no. 4, pp. 375 – 395, 2000.

[30] R. S. Burt, *Structural holes: The social structure of competition*. Harvard university press, 2009.

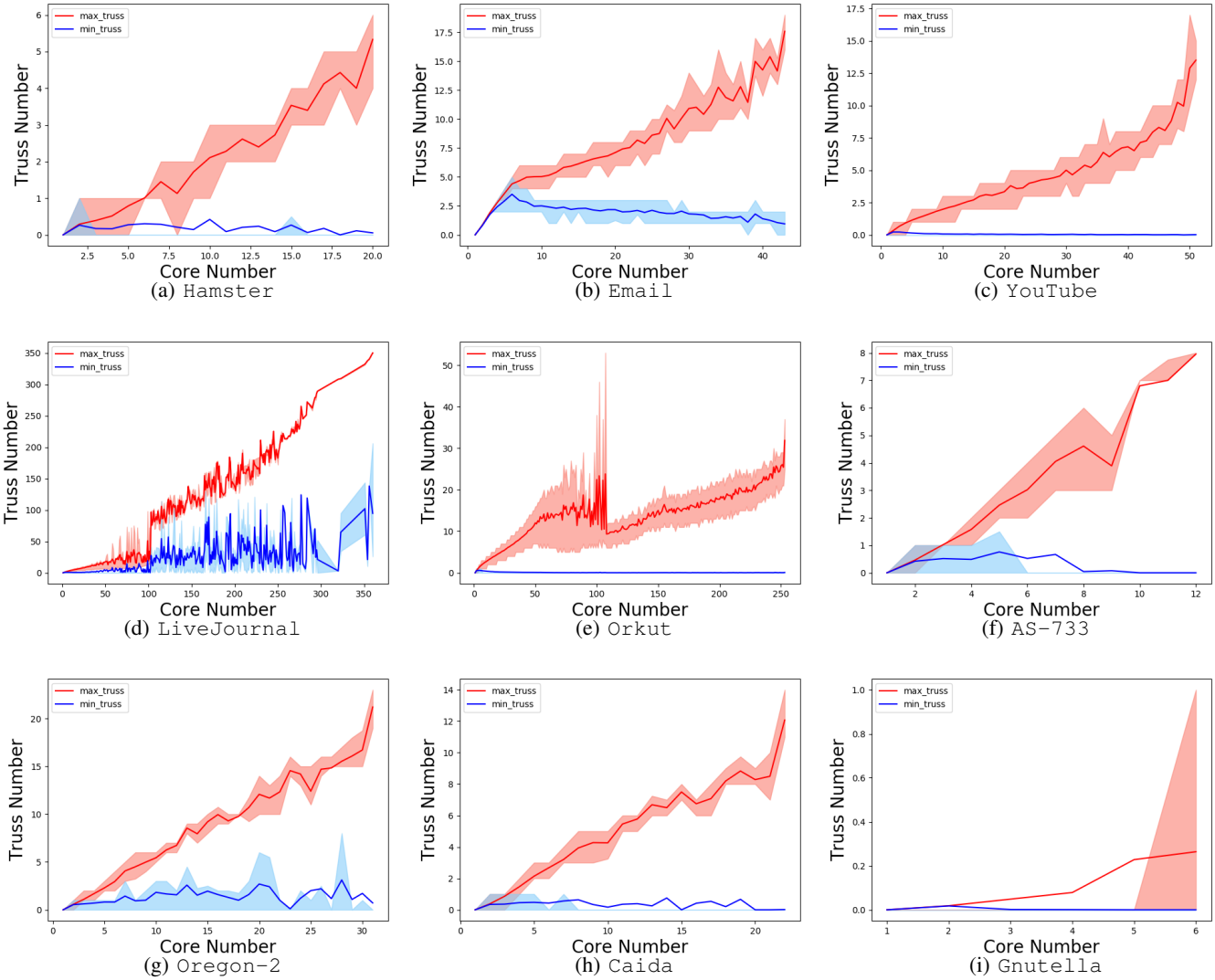
[31] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[32] M. Syakur, B. Khotimah, E. Rochman, and B. Satoto, "Integration k-means clustering method and elbow method for identification of the best customer profile cluster," in *IOP Conference Series: Materials Science and Engineering*, vol. 336, no. 1. IOP Publishing, 2018, p. 012017.

[33] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[34] K. Shin, T. Eliassi-Rad, and C. Faloutsos, "Patterns and anomalies in k-cores of real-world graphs with applications," *Knowledge and Information Systems*, vol. 54, no. 3, pp. 677–710, 2018.

APPENDIX A
OTHER VI AND EI PLOTS



(a) Hamster (b) Email (c) YouTube

(d) LiveJournal (e) Orkut (f) AS-733

(g) Oregon-2 (h) Caida (i) Gnutella

Fig. 11: **Vertex interplay (VI) plots for other real-world graphs (listed but not shown in the paper)(part I). For each vertex with a particular core number, the maximum and minimum of the truss numbers of surrounding edges are shown. Average and interquartile ranges are computed over the vertices with same core number.**

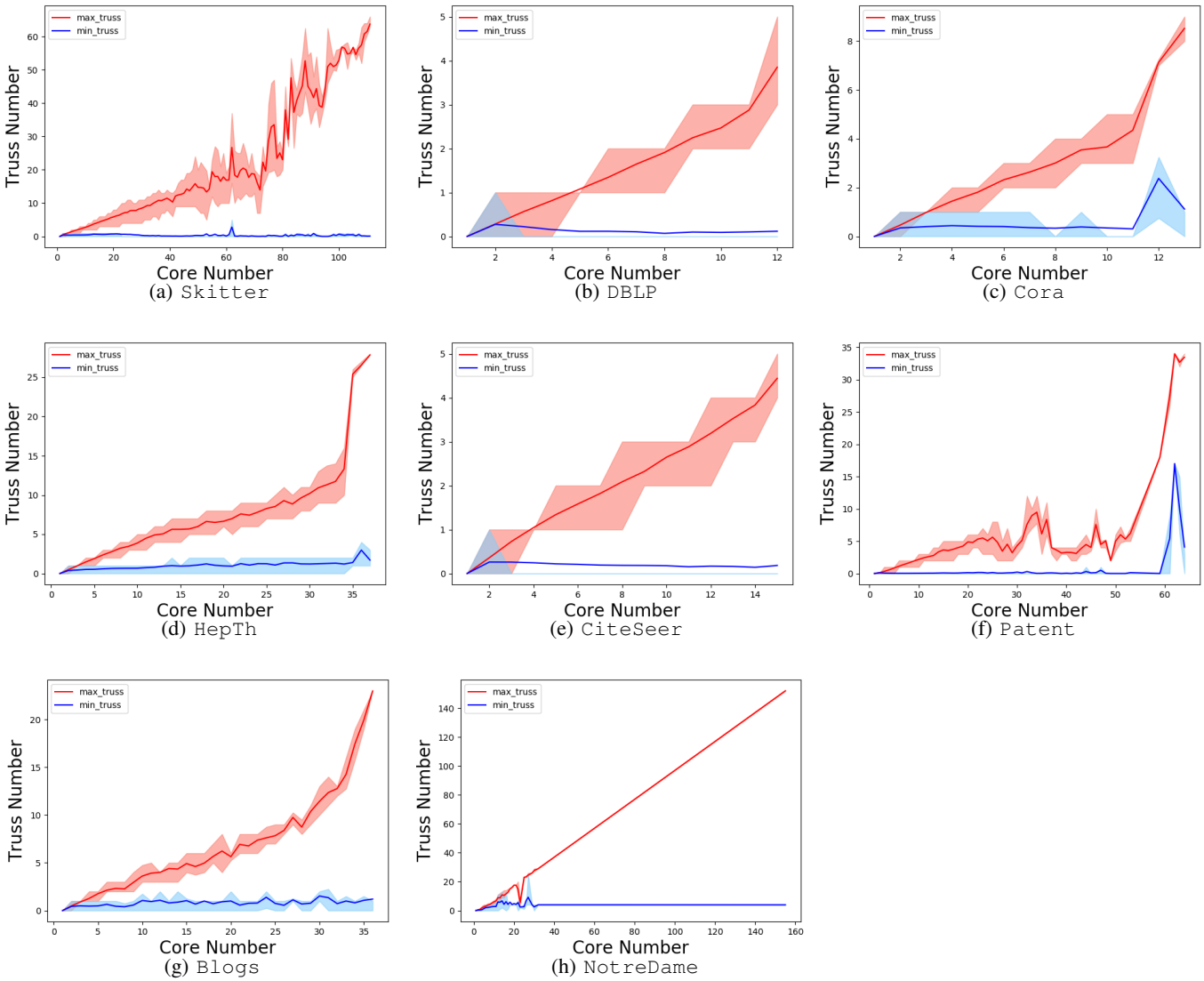
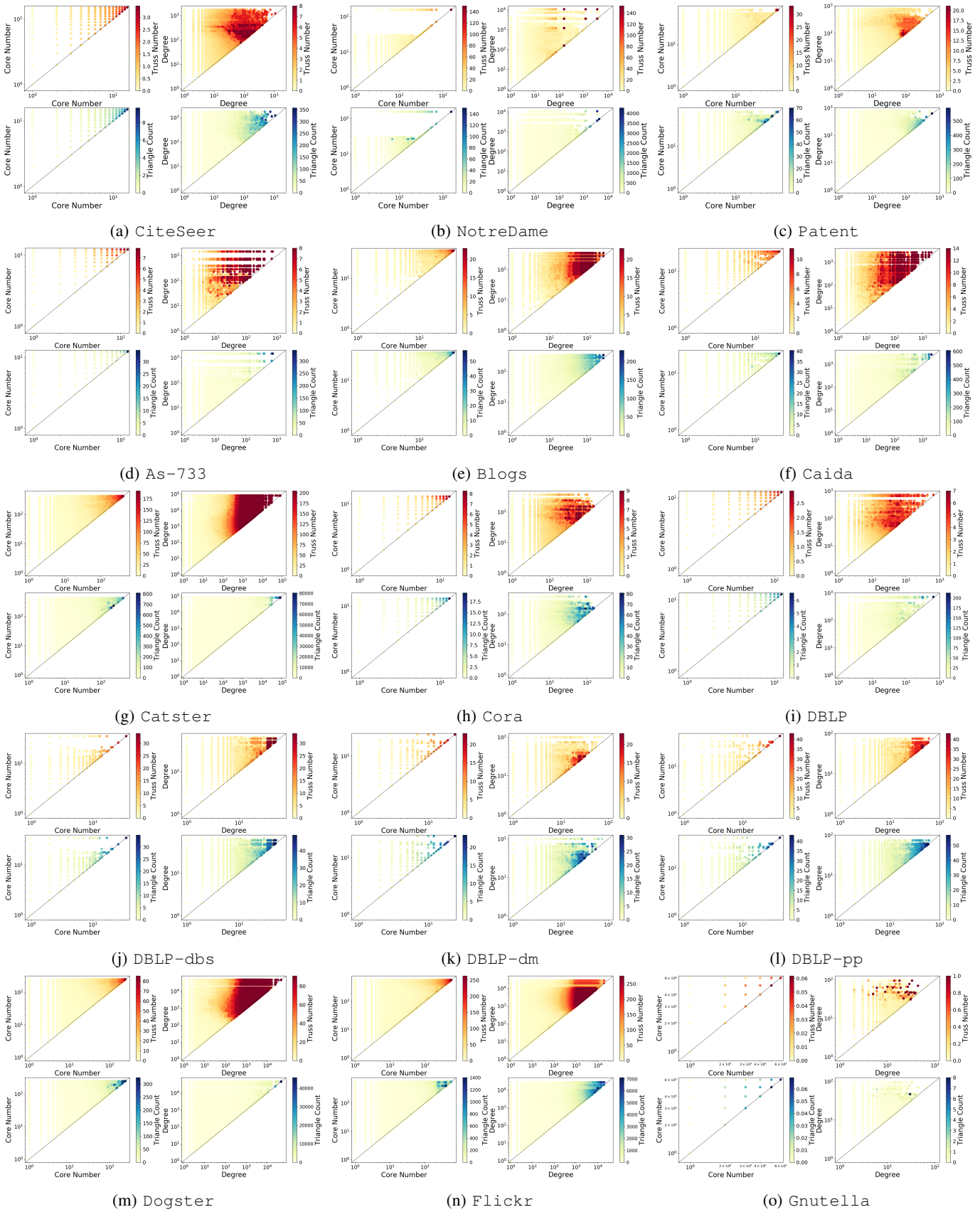


Fig. 12: **Vertex interplay (VI) plots for other real-world graphs (listed but not shown in the paper) (Part II). For each vertex with a particular core number, the maximum and minimum of the truss numbers of surrounding edges are shown. Average and interquartile ranges are computed over the vertices with same core number.**



(a) CiteSeer (b) NotreDame (c) Patent

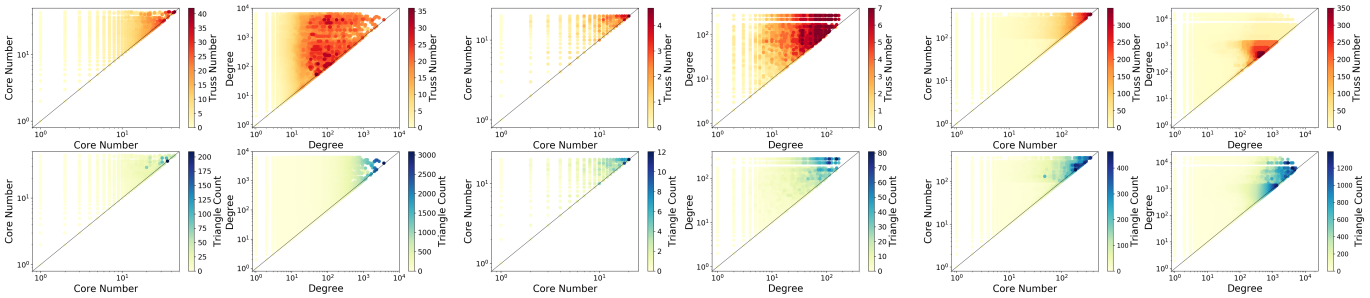
(d) As-733 (e) Blogs (f) Caida

(g) Catster (h) Cora (i) DBLP

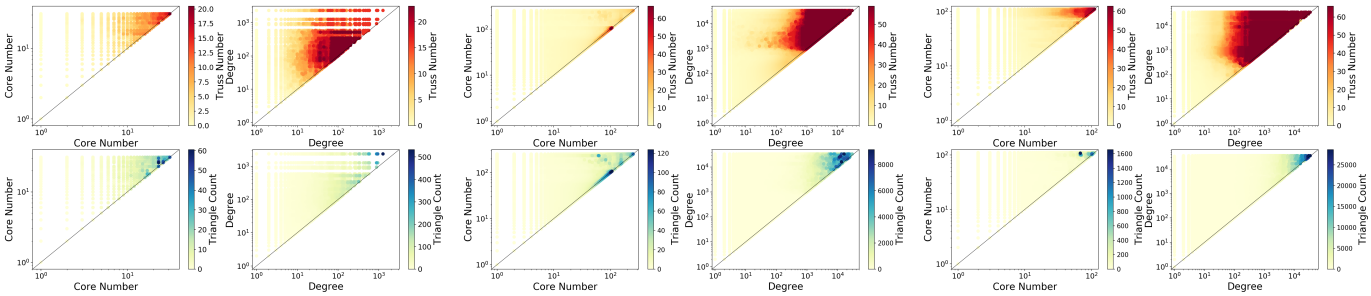
(j) DBLP-dbs (k) DBLP-dm (l) DBLP-pp

(m) Dogster (n) Flickr (o) Gnutella

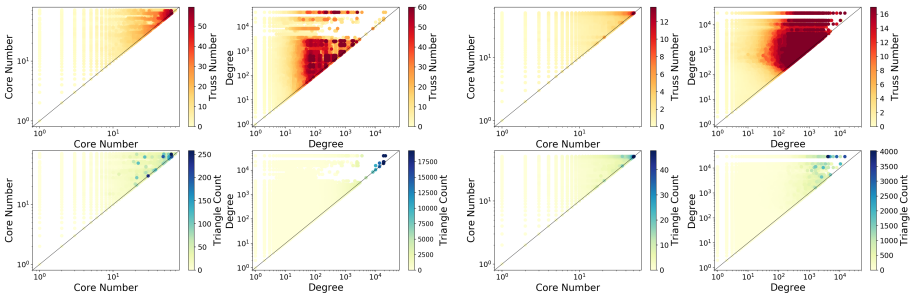
Fig. 13: **Edge interplay (EI) plots for other real-world graphs (Part I). For each pair of endpoints with particular core numbers/degrees, the average of the truss numbers/triangle counts of the edges are shown.**



(a) Google (b) Hamster (c) LiveJournal

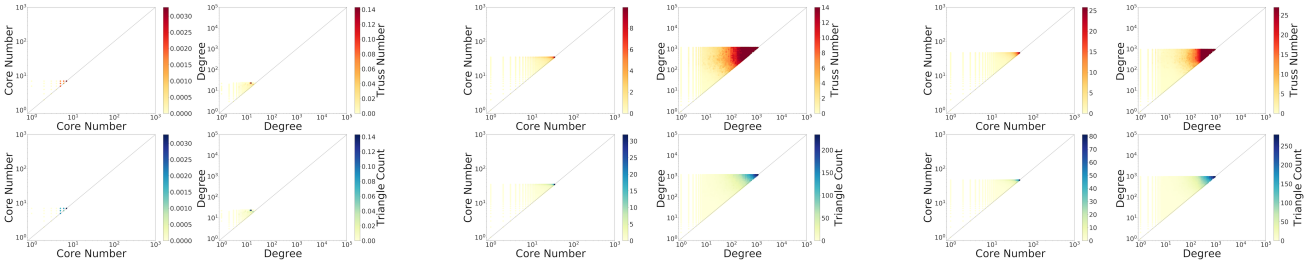


(d) Oregon-2 (e) Orkut (f) Skitter

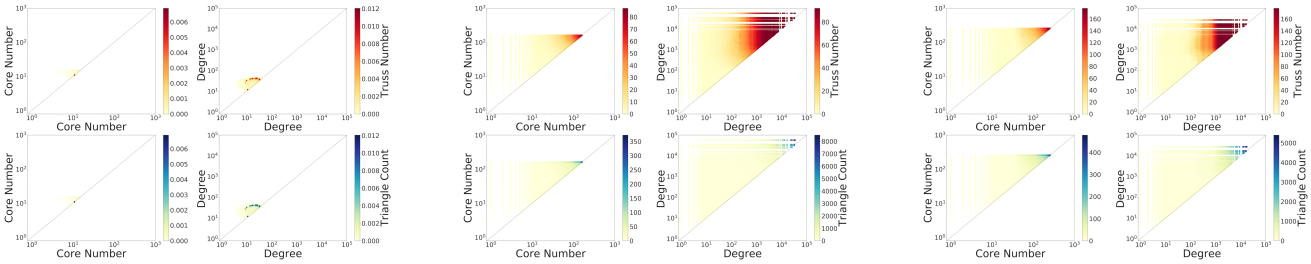


(g) Stanford (h) YouTube

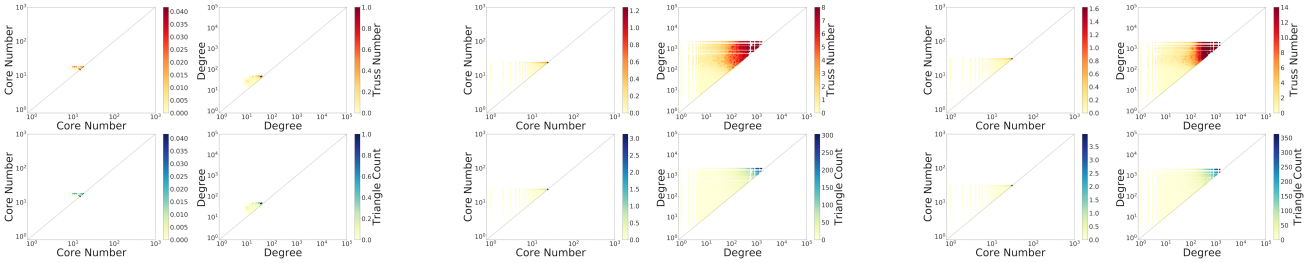
Fig. 14: **Edge interplay (EI) plots for other real-world graphs (Part II). For each pair of endpoints with particular core numbers/degrees, the average of the truss numbers/triangle counts of the edges are shown.**



(a) `Email.` by ER (b) `Email.` by configuration (c) `Email.` by BTER



(d) `BerkStan` by ER (e) `BerkStan` by configuration (f) `BerkStan` by BTER



(g) `HepTh` by ER (h) `HepTh` by configuration (i) `HepTh` by BTER

Fig. 15: **Edge interplay (EI) plots for random graphs with respect to three real-world graphs. For each pair of endpoints with particular core numbers/degrees, the average of the truss numbers/triangle counts of the edges are shown. EI plots of ER, configuration, and BTER models for `Email` (Figures 15a, 15b, 15c), `BerkStan` (Figures 15d, 15e, 15f), and `HepTh` (Figures 15g, 15h, 15i) are given. ER model cannot preserve the edge interplay in any network. Configuration model yields a similar behavior in all three networks, thus cannot capture the behaviors observed for collaboration and web networks. BTER gives the closest estimates for core and truss degeneracies but again exhibits a similar behavior in all cases.**